Experiment No. 03

3.1 Experiment Name

Logical operation and Loop using Assembly Language Programming

3.2 Objectives

- To understand the structure of Assembly Language programming
- To learn about the microprocessor emulator "Emu 8086" and its operation
- To get acquainted with the logical operation using ALU
- To learn how to implement program in “MDA 8086” Trainer Board and interconnect it with “Emu 8086”

3.3 Theory

In assembly language programming, various logical operation is executed through exclusive instruction sets. In this experiment we will be using following operations and loops for study purpose.

3.3.1 Shift Operation: The shift operation simply changes the provided data bit in the register to the left or right. It comes in two varieties.

- **Shift Left (SHL):** The SHL instruction stands for “shift left”. This instruction moves the given register bits one at a time to the left for a positive integer.
- **Shift Right (SHR):** SHR is an abbreviation for "shift right." This instruction moves the given register bits one at a time to the right.
- **Shift Arithmetic Left (SAL):** The SAL is frequently used when numeric multiplication is required. In other words, shift binary one bit to the left while leaving the MSB untouched.
- **Shift Arithmetic Right (SAR):** SAR is quite similar to SAL with a slight change. It shifts binary one bit to the right while leaving the MSB untouched.

3.3.2 Rotate Operation: This instruction rotates the bits. There are four rotation instructions:

- **Left Rotation (ROL):** The ROL instruction indicates to rotate the bits one by one to the left. The only distinction between ROL and SHL is, in SHL, the value of D0 is replaced by another value, whereas in ROL, the value of D0 is replaced by the bit from position D7.
- **Right Rotation (ROR):** The ROR instruction means to rotate the bits to the right one at a time. The sole difference between ROR and SHR is that in SHR, the bit of position D7 is replaced with another value.
- **Left Rotation with Carry (RCL):** The bits in the operand are rotated through the carry flag to the left using RCL. The most significant bit, together with the carry flag, is relocated to the least significant bit by the RCL instruction.
- **Right Rotation with Carry (RCR):** The bits in the operand are rotated through the carry flag using RCR. The least significant bit, together with the carry flag, is relocated to the most significant bit by the RCR instruction.

3.3.3 AND

Only if both of its inputs are true does the AND logic function return true. If one of the inputs is false, the output will be false as well.

3.3.4 OR

In OR operation, if one of the inputs is true, function return true. If one of the inputs is false, the output will be false as well.

3.3.5 X- OR

XOR compares two input bits and generates one output bit. The logic is simple. If the bits are the same, the result is true. If the bits are different, the result is false.

We will also implement looping operation in this experiment. Through this instruction, we will execute operation for the number of times defined before the loop

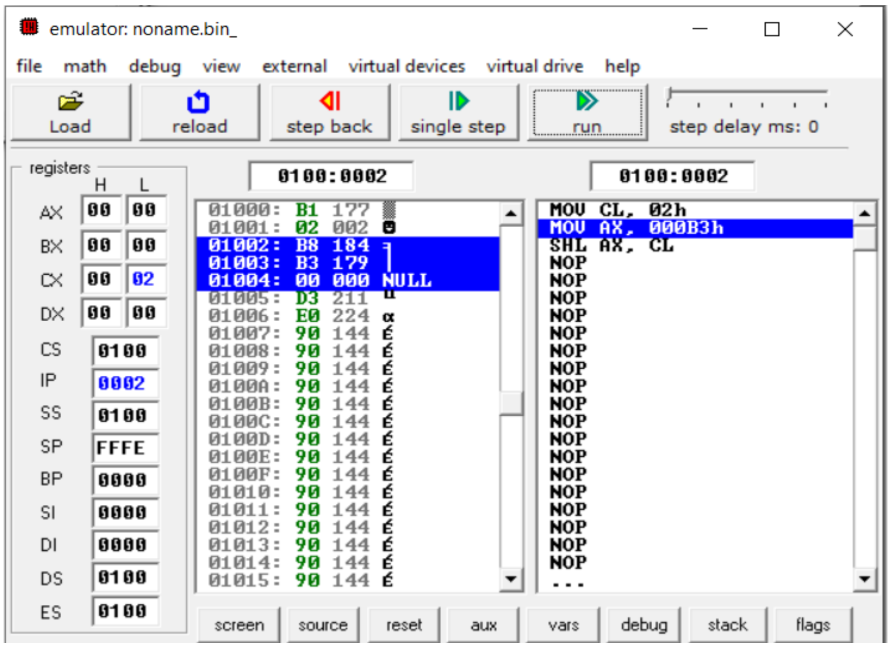
3.4 Apparatus

- Emu 8086 - Microprocessor Emulator

3.5 Emulator Code & output

- **SHL**

```
MOV CL, 2; COUNTER
MOV AX, 0B3H; MOVE 0B3H TO AX
SHL AX, CL; SHIFT LEFT
```

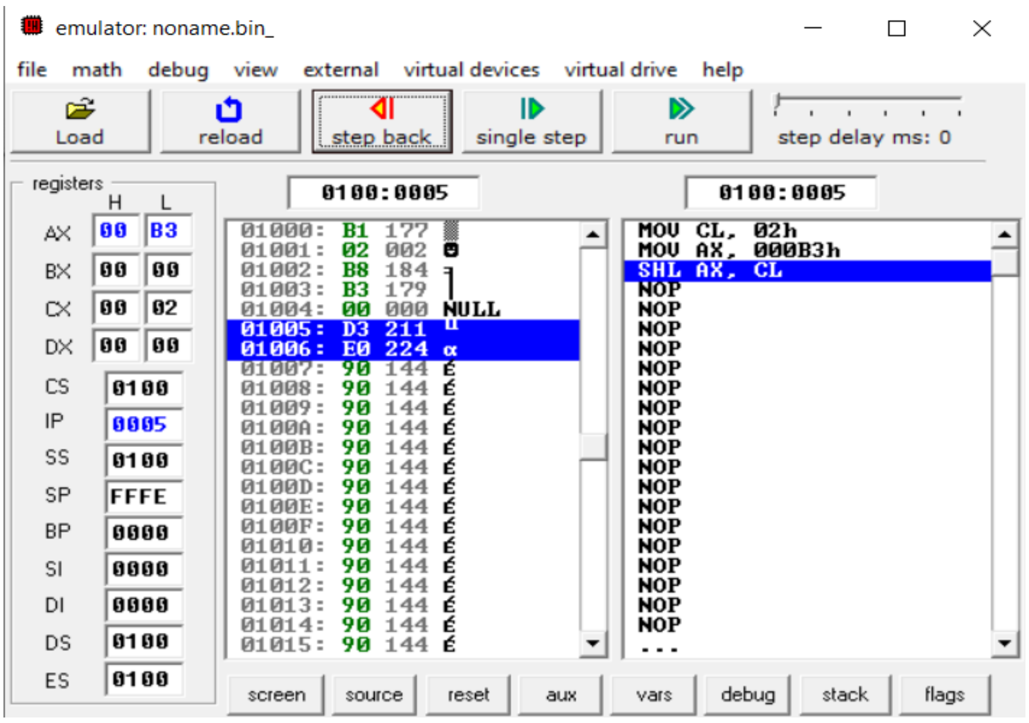


```
registers
     H   L
AX   00  00
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0002
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

```
0100:0002
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: E0 224 α
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0002
MOV  CL, 02h
MOV  AX, 000B3h
SHL  AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.1: Emulator output for SHL (After Step 1)

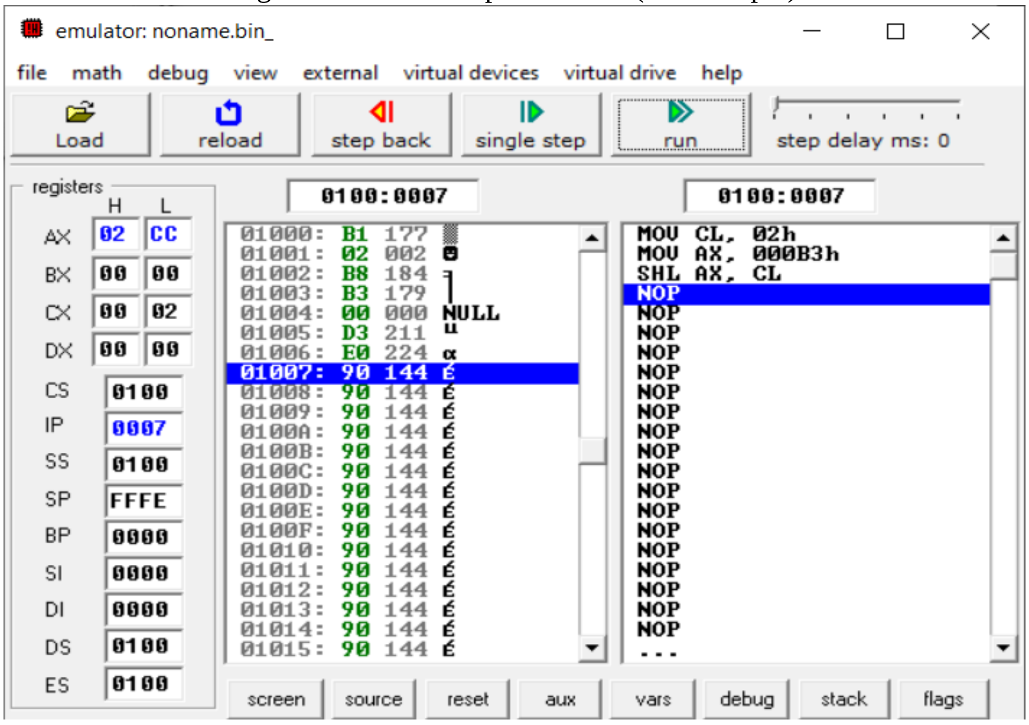


```
registers
      H   L
AX   00  B3
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0005
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

```
0100:0005
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: E0 224 α
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0005
MOV CL, 02h
MOV AX, 000B3h
SHL AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.2: Emulator output for SHL (After Step 2)



```
registers
      H   L
AX   02  CC
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0007
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

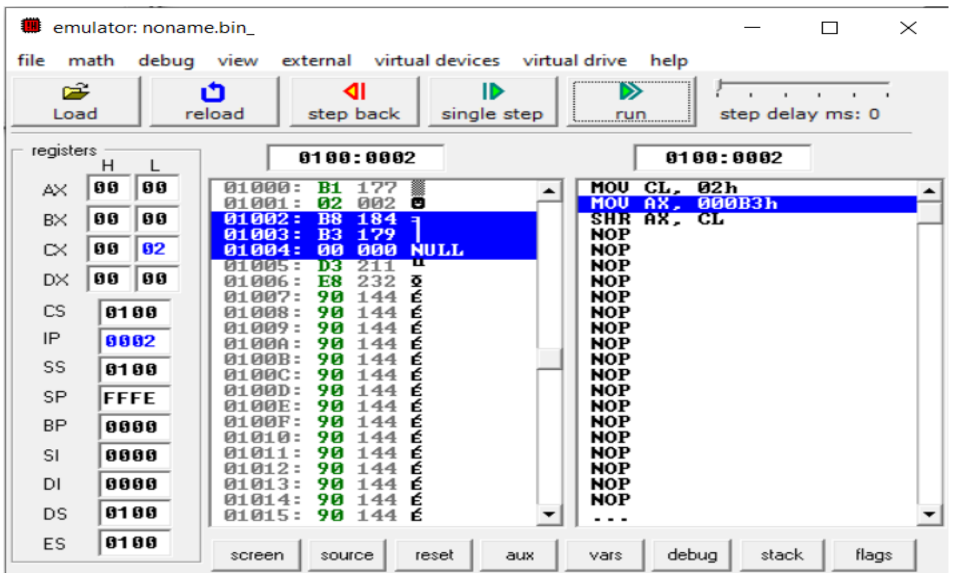
```
0100:0007
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: E0 224 α
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0007
MOV CL, 02h
MOV AX, 000B3h
SHL AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.3: Emulator output for SHL (After Step 3)

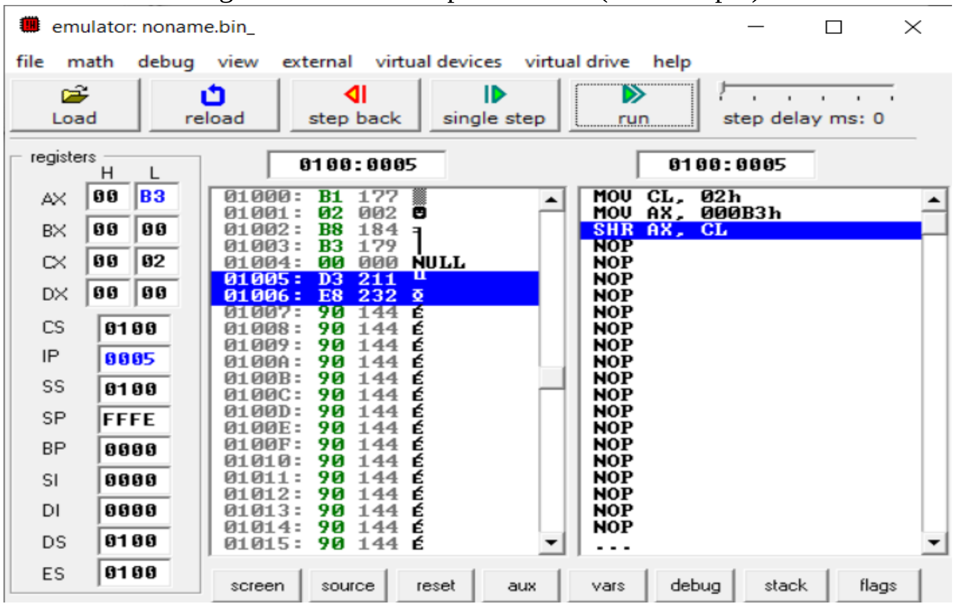
- **SHR**

MOV CL, 2; COUNTER
MOV AX, 0B3H; MOVE 0B3H TO AX
SHR AX, CL; SHIFT LEFT


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
screen source reset aux vars debug stack flags

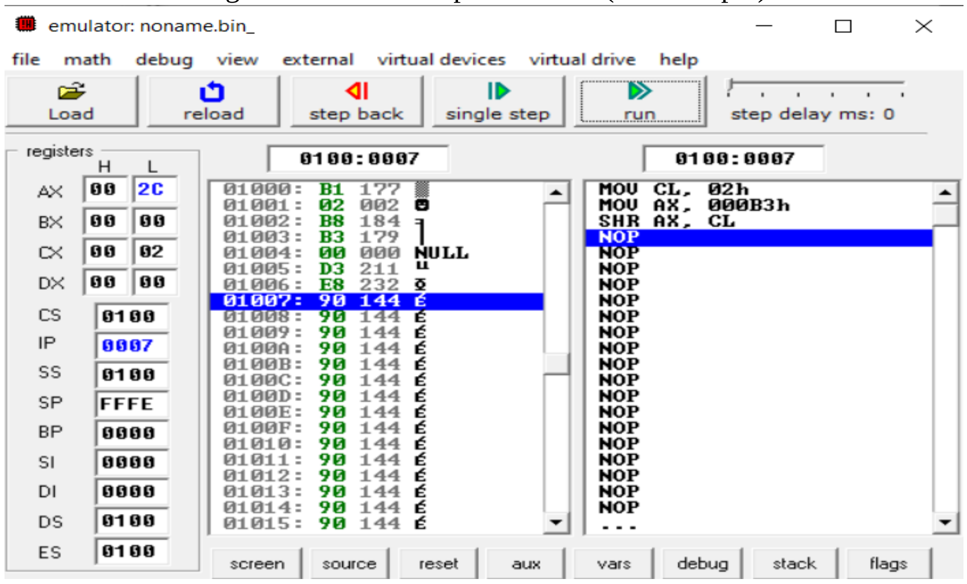
```
registers  H   L
AX        00  00
BX        00  00
CX        00  02
DX        00  00
CS   0100
IP   0002
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100

0100:0002                    0100:0002

01000:  B1  177  ▒           MOV  CL, 02h
01001:  02  002  ☻           MOV  AX, 000B3h
01002:  B8  184  ╕           SHR  AX, CL
01003:  B3  179  │           NOP
01004:  00  000  NULL        NOP
01005:  D3  211  ╙           NOP
01006:  E8  232  Φ           NOP
01007:  90  144  É           NOP
01008:  90  144  É           NOP
01009:  90  144  É           NOP
0100A:  90  144  É           NOP
0100B:  90  144  É           NOP
0100C:  90  144  É           NOP
0100D:  90  144  É           NOP
0100E:  90  144  É           NOP
0100F:  90  144  É           NOP
01010:  90  144  É           NOP
01011:  90  144  É           NOP
01012:  90  144  É           NOP
01013:  90  144  É           NOP
01014:  90  144  É           NOP
01015:  90  144  É           ...
```

Fig 3.4: Emulator output for SHR (After Step 1)


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
screen source reset aux vars debug stack flags

```
registers  H   L
AX        00  B3
BX        00  00
CX        00  02
DX        00  00
CS   0100
IP   0005
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100

0100:0005                    0100:0005

01000:  B1  177  ▒           MOV  CL, 02h
01001:  02  002  ☻           MOV  AX, 000B3h
01002:  B8  184  ╕           SHR  AX, CL
01003:  B3  179  │           NOP
01004:  00  000  NULL        NOP
01005:  D3  211  ╙           NOP
01006:  E8  232  Φ           NOP
01007:  90  144  É           NOP
01008:  90  144  É           NOP
01009:  90  144  É           NOP
0100A:  90  144  É           NOP
0100B:  90  144  É           NOP
0100C:  90  144  É           NOP
0100D:  90  144  É           NOP
0100E:  90  144  É           NOP
0100F:  90  144  É           NOP
01010:  90  144  É           NOP
01011:  90  144  É           NOP
01012:  90  144  É           NOP
01013:  90  144  É           NOP
01014:  90  144  É           NOP
01015:  90  144  É           ...
```

Fig 3.5: Emulator output for SHR (After Step 2)

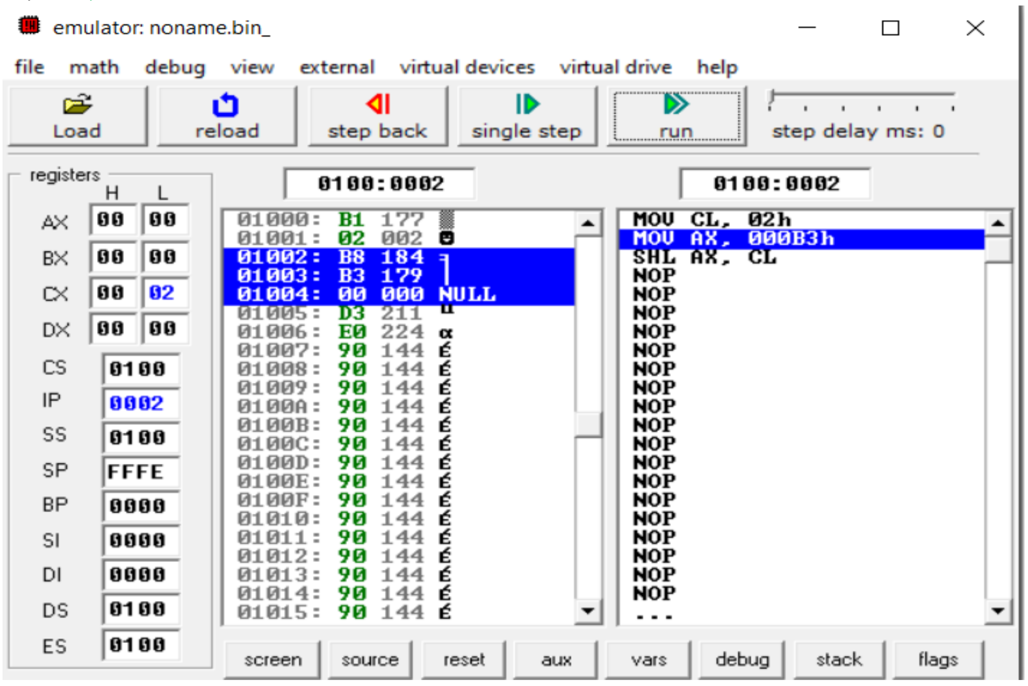

emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
screen source reset aux vars debug stack flags

```
registers  H   L
AX        00  2C
BX        00  00
CX        00  02
DX        00  00
CS   0100
IP   0007
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100

0100:0007                    0100:0007

01000:  B1  177  ▒           MOV  CL, 02h
01001:  02  002  ☻           MOV  AX, 000B3h
01002:  B8  184  ╕           SHR  AX, CL
01003:  B3  179  │           NOP
01004:  00  000  NULL        NOP
01005:  D3  211  ╙           NOP
01006:  E8  232  Φ           NOP
01007:  90  144  É           NOP
01008:  90  144  É           NOP
01009:  90  144  É           NOP
0100A:  90  144  É           NOP
0100B:  90  144  É           NOP
0100C:  90  144  É           NOP
0100D:  90  144  É           NOP
0100E:  90  144  É           NOP
0100F:  90  144  É           NOP
01010:  90  144  É           NOP
01011:  90  144  É           NOP
01012:  90  144  É           NOP
01013:  90  144  É           NOP
01014:  90  144  É           NOP
01015:  90  144  É           ...
```

Fig 3.6: Emulator output for SHR (After Step 3)

- **SAL**

MOV CL, 2; COUNTER
MOV AX, 0B3H; MOVE 0B3H TO AX
SAL AX, CL; SHIFT ARITHAMTIC LEFT

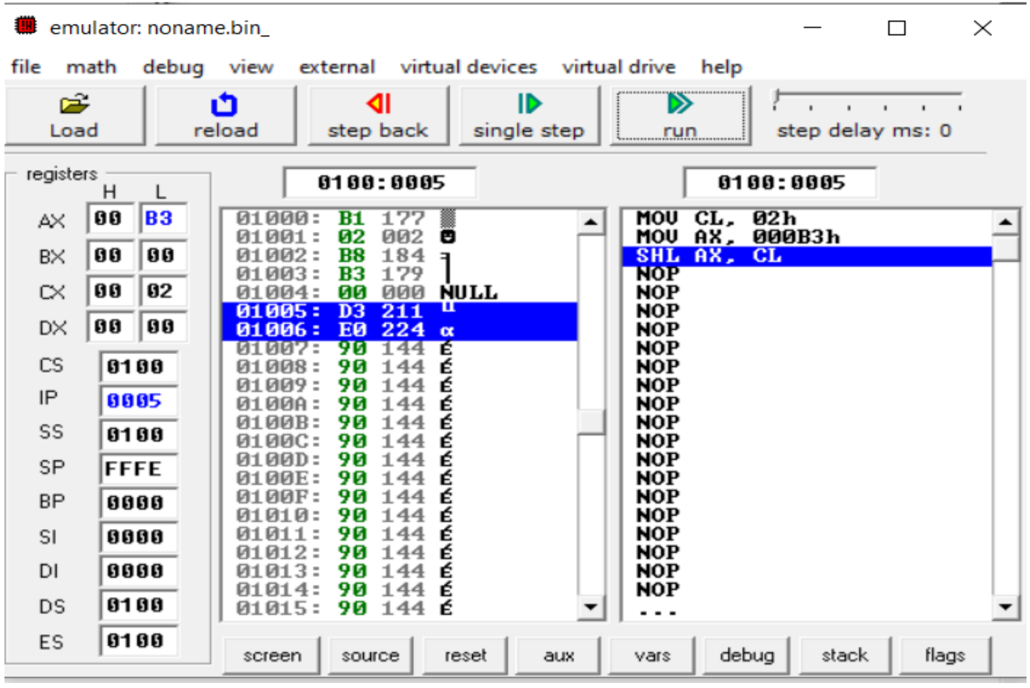


```
     H   L
AX  00  00
BX  00  00
CX  00  02
DX  00  00
CS  0100
IP  0002
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:0002
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: E0 224 α
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0002
MOV  CL, 02h
MOV  AX, 000B3h
SHL  AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.7: Emulator output for SAL (After Step 1)

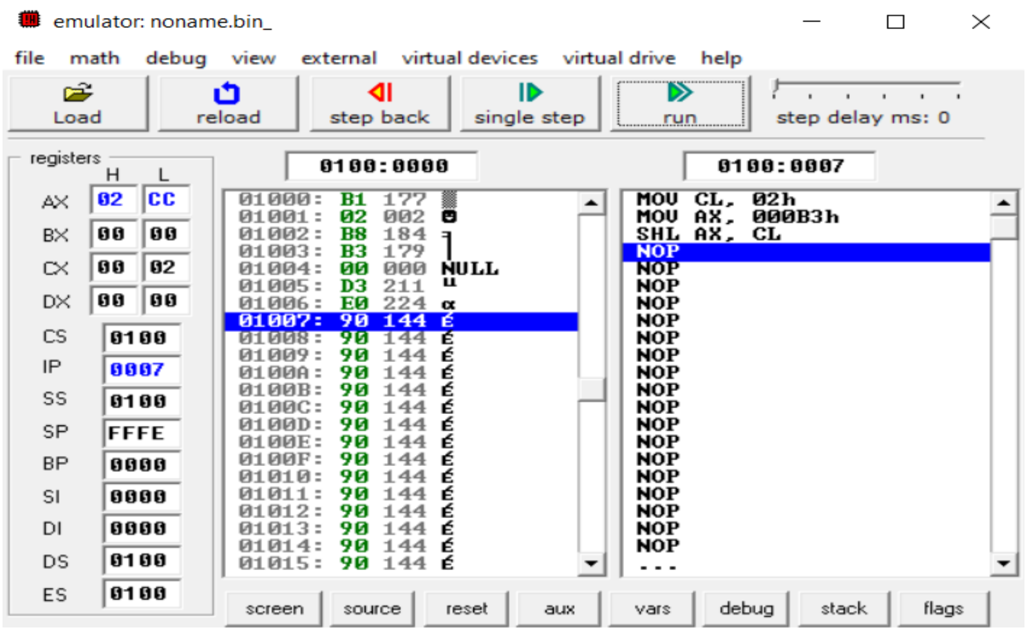


```
     H   L
AX  00  B3
BX  00  00
CX  00  02
DX  00  00
CS  0100
IP  0005
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:0005
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: E0 224 α
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0005
MOV  CL, 02h
MOV  AX, 000B3h
SHL  AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.8: Emulator output for SAL (After Step 2)


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
screen source reset aux vars debug stack flags

```
     H   L
AX   02  CC
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0007
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

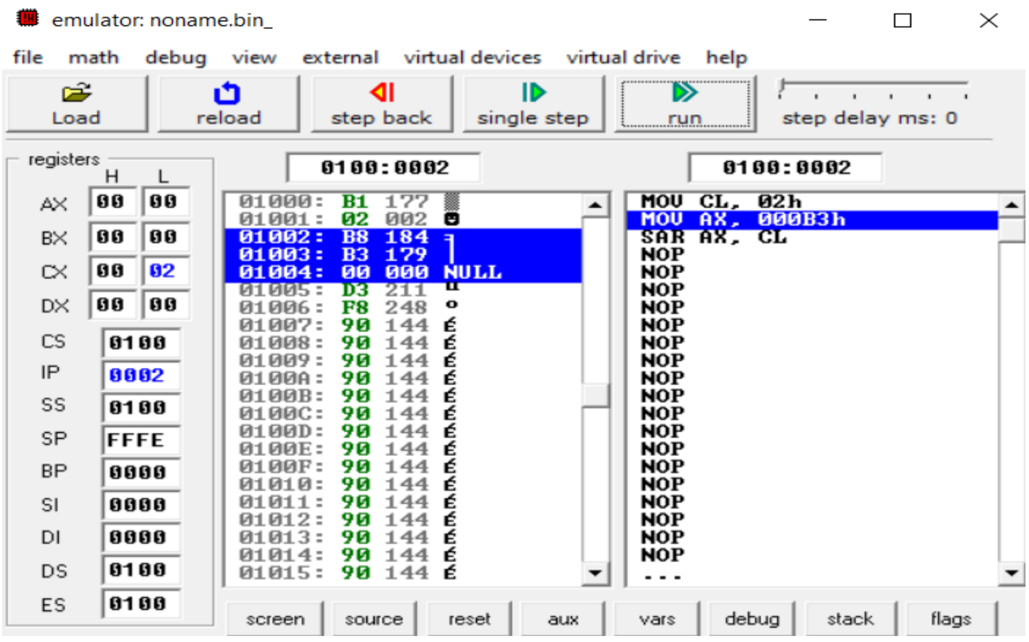
```
0100:0000
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: E0 224 α
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0007
MOV  CL, 02h
MOV  AX, 000B3h
SHL  AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.9: Emulator output for SAL (After Step 3)

- **SAR**

MOV CL, 2; COUNTER
MOV AX, 0B3H; MOVE 0B3H TO AX
SAR AX, CL; SHIFT ARITHAMTIC LEFT

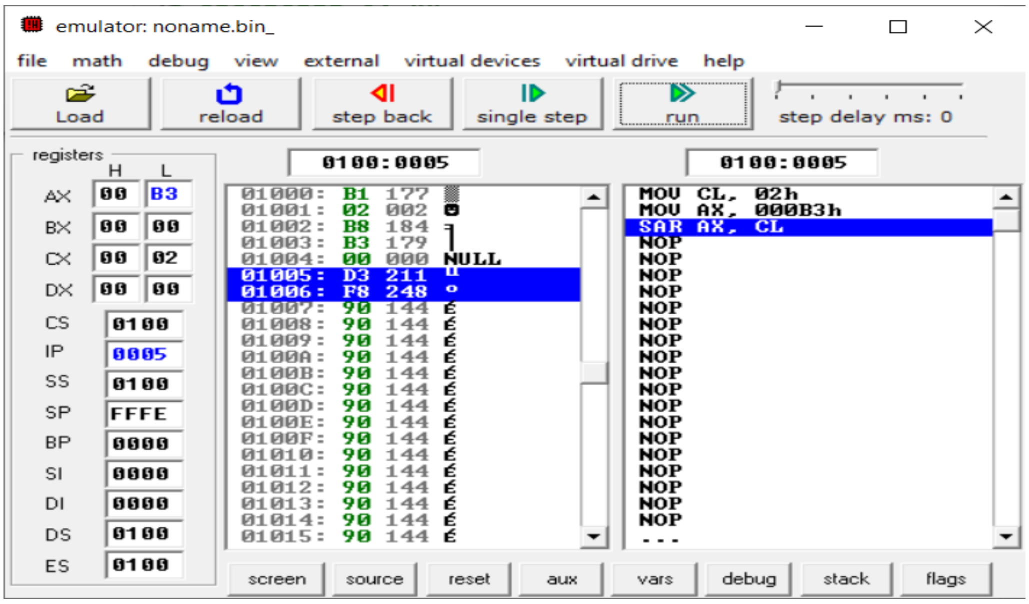

emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
screen source reset aux vars debug stack flags

```
     H   L
AX   00  00
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0002
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

```
0100:0002
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: F8 248 °
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0002
MOV  CL, 02h
MOV  AX, 000B3h
SAR  AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.10: Emulator output for SAR (After Step 1)

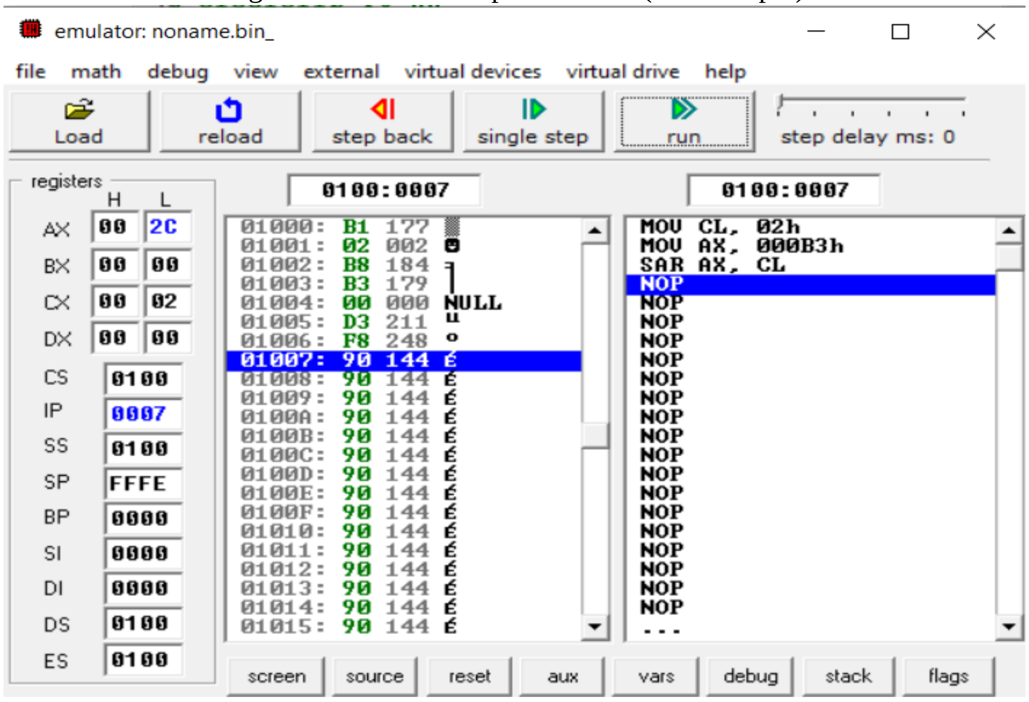

emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
registers
     H   L
AX   00  B3
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0005
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

```
0100:0005
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: F8 248 °
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0005
MOV CL, 02h
MOV AX, 000B3h
SAR AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.11: Emulator output for SAR (After Step 2)


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
registers
     H   L
AX   00  2C
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0007
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

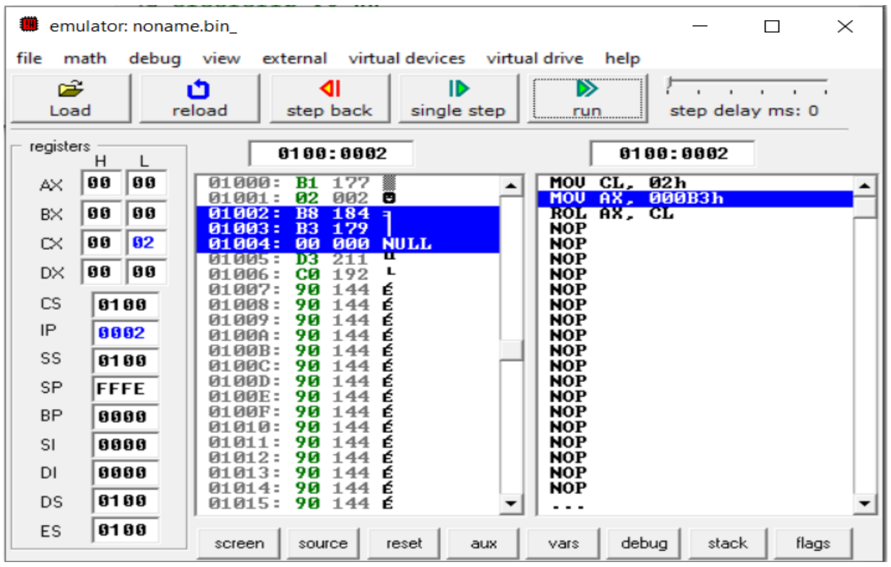
```
0100:0007
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: F8 248 °
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0007
MOV CL, 02h
MOV AX, 000B3h
SAR AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.12: Emulator output for SAR (After Step 3)

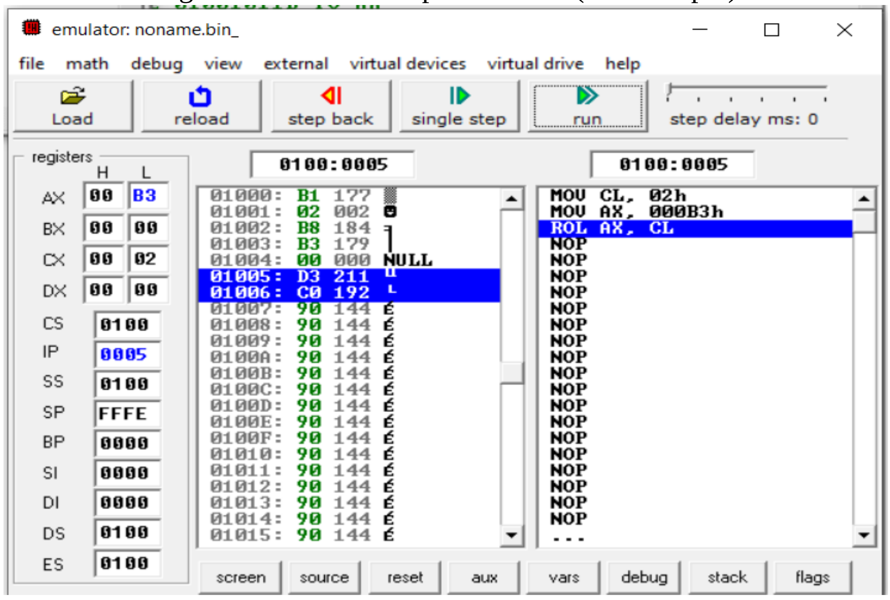
- **ROL**

```
MOV CL, 2; COUNTER
MOV AX, 0B3H; MOVE 0B3H TO AX
ROL AX, CL; ROTATE LEFT
```


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

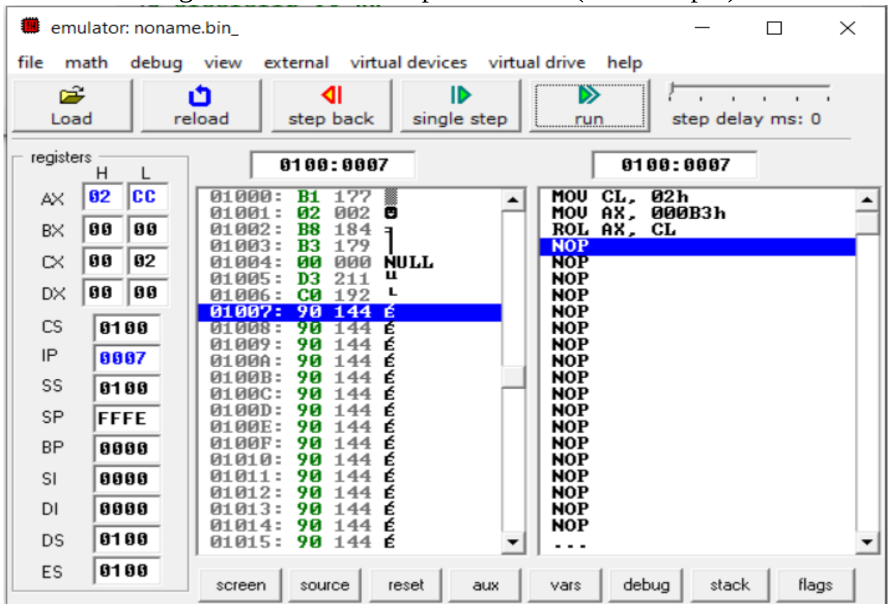
```
registers  H   L
AX        00  00
BX        00  00
CX        00  02
DX        00  00
CS        0100
IP        0002
SS        0100
SP        FFFE
BP        0000
SI        0000
DI        0000
DS        0100
ES        0100

0100:0002                    0100:0002
01000: B1 177 ▒              MOV  CL, 02h
01001: 02 002 ☻              MOV  AX, 000B3h
01002: B8 184 ╕              ROL  AX, CL
01003: B3 179 │              NOP
01004: 00 000 NULL           NOP
01005: D3 211 ╙              NOP
01006: C0 192 └              NOP
01007: 90 144 É              NOP
01008: 90 144 É              NOP
01009: 90 144 É              NOP
0100A: 90 144 É              NOP
0100B: 90 144 É              NOP
0100C: 90 144 É              NOP
0100D: 90 144 É              NOP
0100E: 90 144 É              NOP
0100F: 90 144 É              NOP
01010: 90 144 É              NOP
01011: 90 144 É              NOP
01012: 90 144 É              NOP
01013: 90 144 É              NOP
01014: 90 144 É              NOP
01015: 90 144 É              ...
```

Fig 3.13: Emulator output for ROL (After Step 1)


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
registers  H   L
AX        00  B3
BX        00  00
CX        00  02
DX        00  00
CS        0100
IP        0005
SS        0100
SP        FFFE
BP        0000
SI        0000
DI        0000
DS        0100
ES        0100

0100:0005                    0100:0005
01000: B1 177 ▒              MOV  CL, 02h
01001: 02 002 ☻              MOV  AX, 000B3h
01002: B8 184 ╕              ROL  AX, CL
01003: B3 179 │              NOP
01004: 00 000 NULL           NOP
01005: D3 211 ╙              NOP
01006: C0 192 └              NOP
01007: 90 144 É              NOP
01008: 90 144 É              NOP
01009: 90 144 É              NOP
0100A: 90 144 É              NOP
0100B: 90 144 É              NOP
0100C: 90 144 É              NOP
0100D: 90 144 É              NOP
0100E: 90 144 É              NOP
0100F: 90 144 É              NOP
01010: 90 144 É              NOP
01011: 90 144 É              NOP
01012: 90 144 É              NOP
01013: 90 144 É              NOP
01014: 90 144 É              NOP
01015: 90 144 É              ...
```

Fig 3.14: Emulator output for ROL (After Step 2)

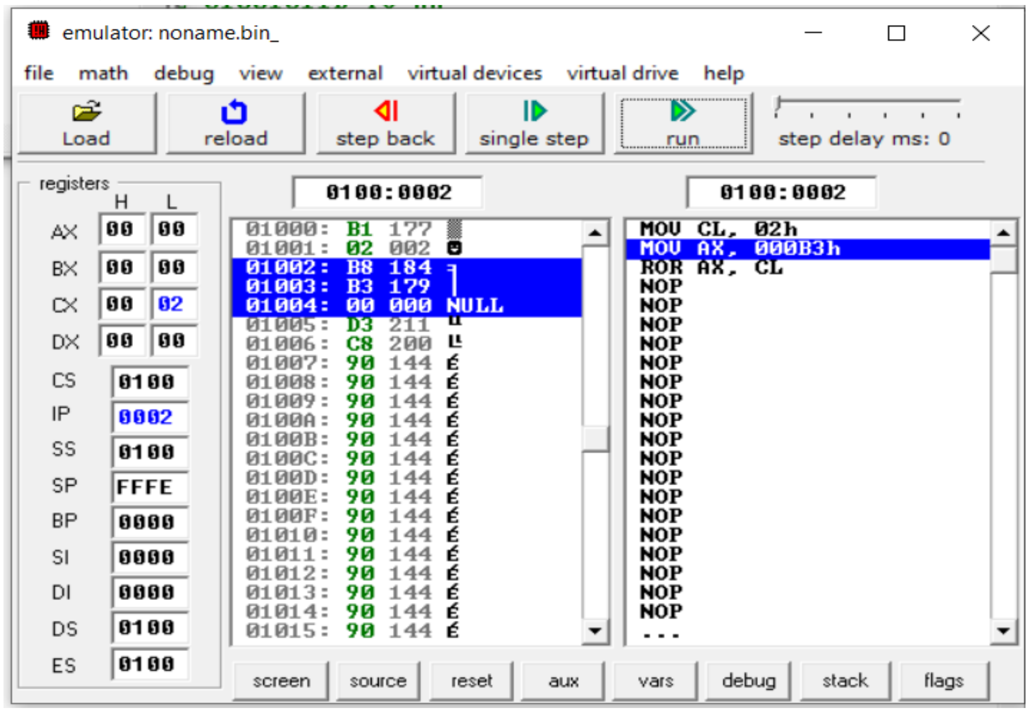

emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
registers  H   L
AX        02  CC
BX        00  00
CX        00  02
DX        00  00
CS        0100
IP        0007
SS        0100
SP        FFFE
BP        0000
SI        0000
DI        0000
DS        0100
ES        0100

0100:0007                    0100:0007
01000: B1 177 ▒              MOV  CL, 02h
01001: 02 002 ☻              MOV  AX, 000B3h
01002: B8 184 ╕              ROL  AX, CL
01003: B3 179 │              NOP
01004: 00 000 NULL           NOP
01005: D3 211 ╙              NOP
01006: C0 192 └              NOP
01007: 90 144 É              NOP
01008: 90 144 É              NOP
01009: 90 144 É              NOP
0100A: 90 144 É              NOP
0100B: 90 144 É              NOP
0100C: 90 144 É              NOP
0100D: 90 144 É              NOP
0100E: 90 144 É              NOP
0100F: 90 144 É              NOP
01010: 90 144 É              NOP
01011: 90 144 É              NOP
01012: 90 144 É              NOP
01013: 90 144 É              NOP
01014: 90 144 É              NOP
01015: 90 144 É              ...
```

Fig 3.15: Emulator output for ROL (After Step 3)

- **ROR**

MOV CL, 2; COUNTER
MOV AX, 0B3H; MOVE 0B3H TO AX
ROR AX, CL; ROTATE RIGHT

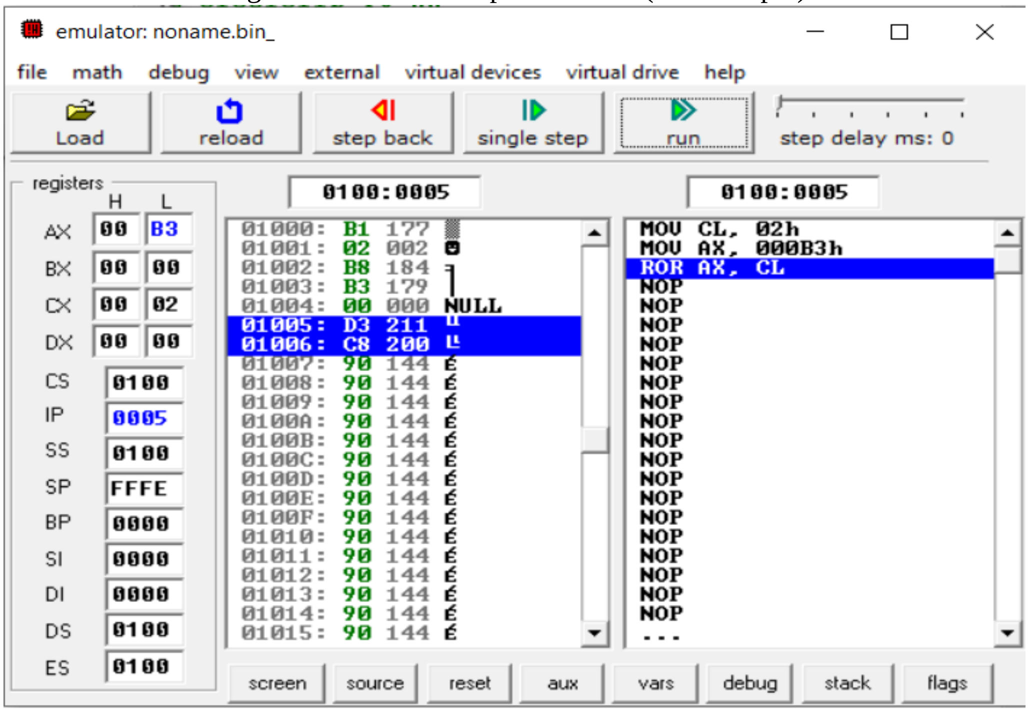


```
registers
     H   L
AX  00  00
BX  00  00
CX  00  02
DX  00  00
CS  0100
IP  0002
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:0002
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: C8 200 ╚
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0002
MOV  CL, 02h
MOV  AX, 000B3h
ROR  AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.16: Emulator output for ROR (After Step 1)

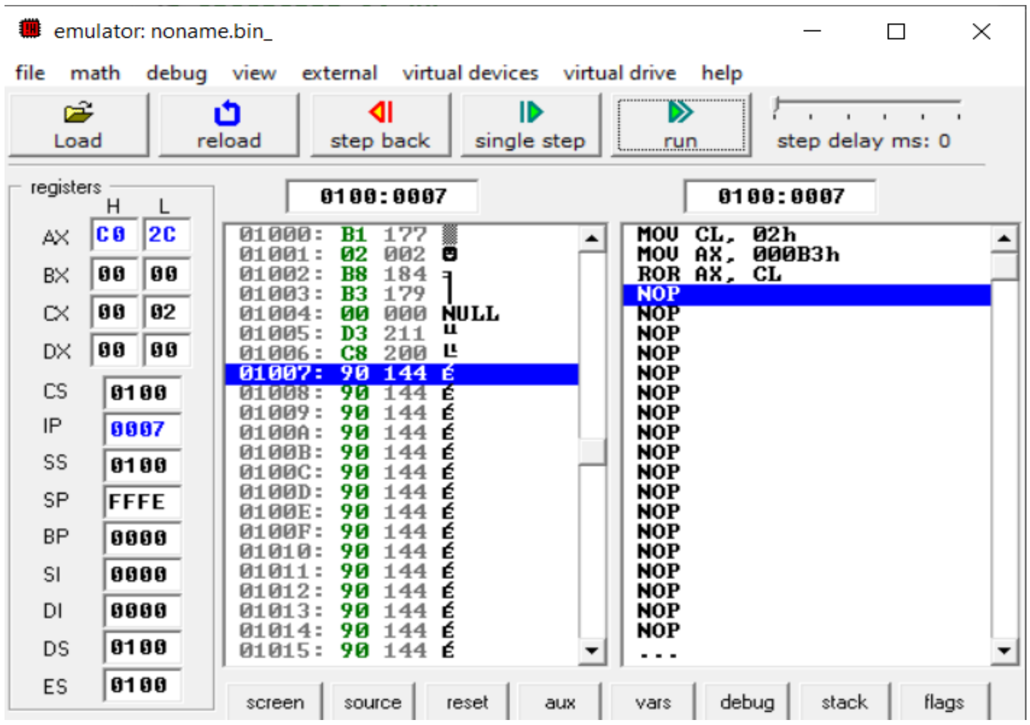


```
registers
     H   L
AX  00  B3
BX  00  00
CX  00  02
DX  00  00
CS  0100
IP  0005
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:0005
01000: B1 177 ▒
01001: 02 002 ☻
01002: B8 184 ╕
01003: B3 179 │
01004: 00 000 NULL
01005: D3 211 ╙
01006: C8 200 ╚
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0005
MOV  CL, 02h
MOV  AX, 000B3h
ROR  AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.17: Emulator output for ROR (After Step 2)



```
registers
      H   L
AX   C0  2C
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0007
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

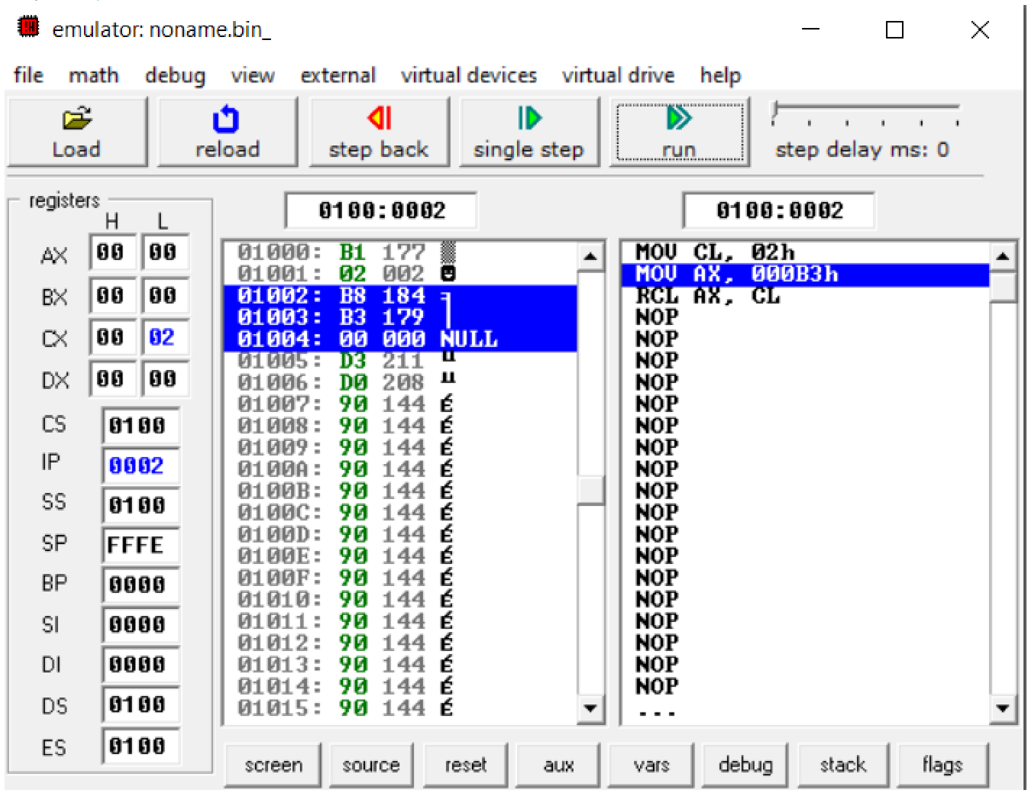
```
0100:0007

01000:  B1  177  ▒
01001:  02  002  ☻
01002:  B8  184  ╕
01003:  B3  179  │
01004:  00  000  NULL
01005:  D3  211  ╙
01006:  C8  200  ╚
01007:  90  144  É
01008:  90  144  É
01009:  90  144  É
0100A:  90  144  É
0100B:  90  144  É
0100C:  90  144  É
0100D:  90  144  É
0100E:  90  144  É
0100F:  90  144  É
01010:  90  144  É
01011:  90  144  É
01012:  90  144  É
01013:  90  144  É
01014:  90  144  É
01015:  90  144  É
```

```
0100:0007

MOV  CL, 02h
MOV  AX, 000B3h
ROR  AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.18: Emulator output for ROR (After Step 3)

- **RCL**

MOV CL, 2; COUNTER
MOV AX, 0B3H; MOVE 0B3H TO AX
RCL AX, CL; ROTATE WITH CARRY TO LEFT

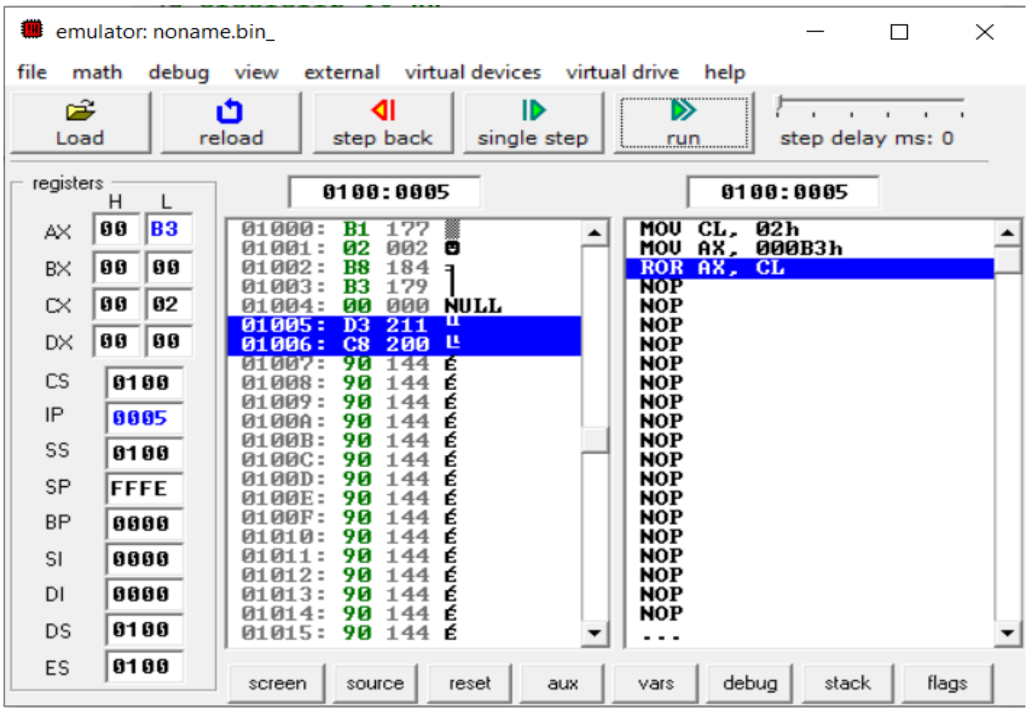


```
registers
      H   L
AX   00  00
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0002
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

```
0100:0002

01000:  B1  177  ▒
01001:  02  002  ☻
01002:  B8  184  ╕
01003:  B3  179  │
01004:  00  000  NULL
01005:  D3  211  ╙
01006:  D0  208  ╨
01007:  90  144  É
01008:  90  144  É
01009:  90  144  É
0100A:  90  144  É
0100B:  90  144  É
0100C:  90  144  É
0100D:  90  144  É
0100E:  90  144  É
0100F:  90  144  É
01010:  90  144  É
01011:  90  144  É
01012:  90  144  É
01013:  90  144  É
01014:  90  144  É
01015:  90  144  É
```

```
0100:0002

MOV  CL, 02h
MOV  AX, 000B3h
RCL  AX, CL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

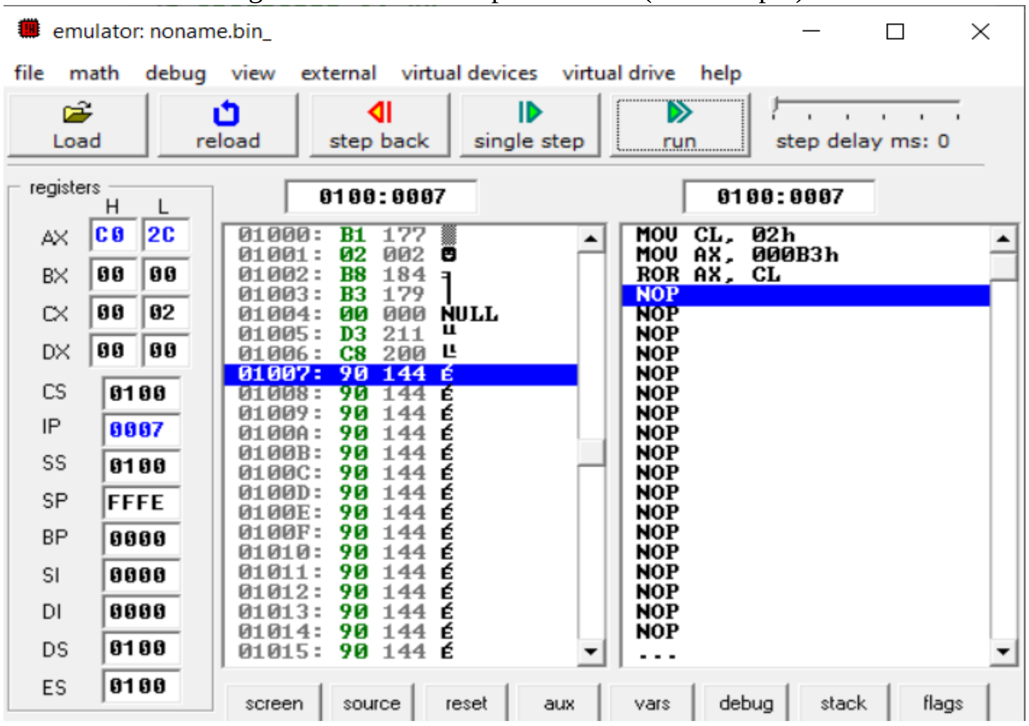
Fig 3.19: Emulator output for RCL (After Step 1)


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
registers
     H   L
AX   00  B3
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0005
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

```
0100:0005                      0100:0005

01000: B1 177 ▒                MOV  CL, 02h
01001: 02 002 ☻                MOV  AX, 000B3h
01002: B8 184 ╕                ROR  AX, CL
01003: B3 179 │                NOP
01004: 00 000 NULL             NOP
01005: D3 211 ╙                NOP
01006: C8 200 ╚                NOP
01007: 90 144 É                NOP
01008: 90 144 É                NOP
01009: 90 144 É                NOP
0100A: 90 144 É                NOP
0100B: 90 144 É                NOP
0100C: 90 144 É                NOP
0100D: 90 144 É                NOP
0100E: 90 144 É                NOP
0100F: 90 144 É                NOP
01010: 90 144 É                NOP
01011: 90 144 É                NOP
01012: 90 144 É                NOP
01013: 90 144 É                NOP
01014: 90 144 É                NOP
01015: 90 144 É                NOP
                               ...
```

Fig 3.20: Emulator output for RCL (After Step 2)


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

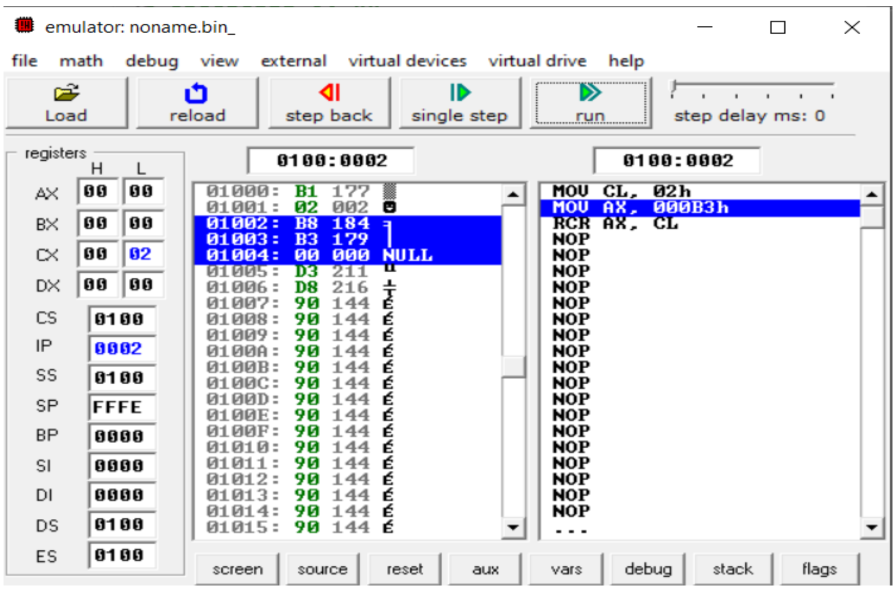
```
registers
     H   L
AX   C0  2C
BX   00  00
CX   00  02
DX   00  00
CS   0100
IP   0007
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

```
0100:0007                      0100:0007

01000: B1 177 ▒                MOV  CL, 02h
01001: 02 002 ☻                MOV  AX, 000B3h
01002: B8 184 ╕                ROR  AX, CL
01003: B3 179 │                NOP
01004: 00 000 NULL             NOP
01005: D3 211 ╙                NOP
01006: C8 200 ╚                NOP
01007: 90 144 É                NOP
01008: 90 144 É                NOP
01009: 90 144 É                NOP
0100A: 90 144 É                NOP
0100B: 90 144 É                NOP
0100C: 90 144 É                NOP
0100D: 90 144 É                NOP
0100E: 90 144 É                NOP
0100F: 90 144 É                NOP
01010: 90 144 É                NOP
01011: 90 144 É                NOP
01012: 90 144 É                NOP
01013: 90 144 É                NOP
01014: 90 144 É                NOP
01015: 90 144 É                NOP
                               ...
```

Fig 3.21: Emulator output for RCL (After Step 3)

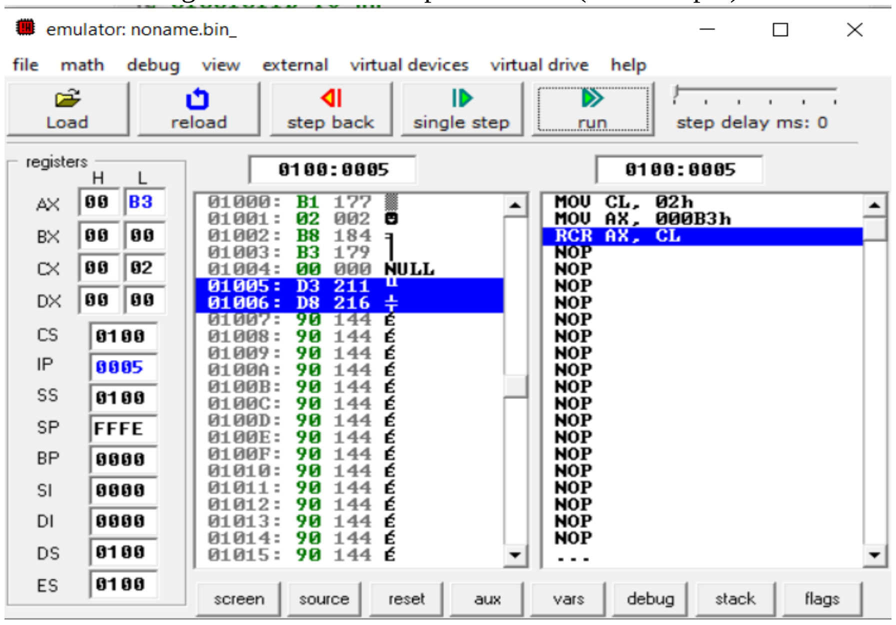
- **RCR**

```
MOV CL, 2; COUNTER
MOV AX, 0B3H; MOVE 0B3H TO AX
RCR AX, CL; ROTATE WITH CARRY TO RIGHT
```


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

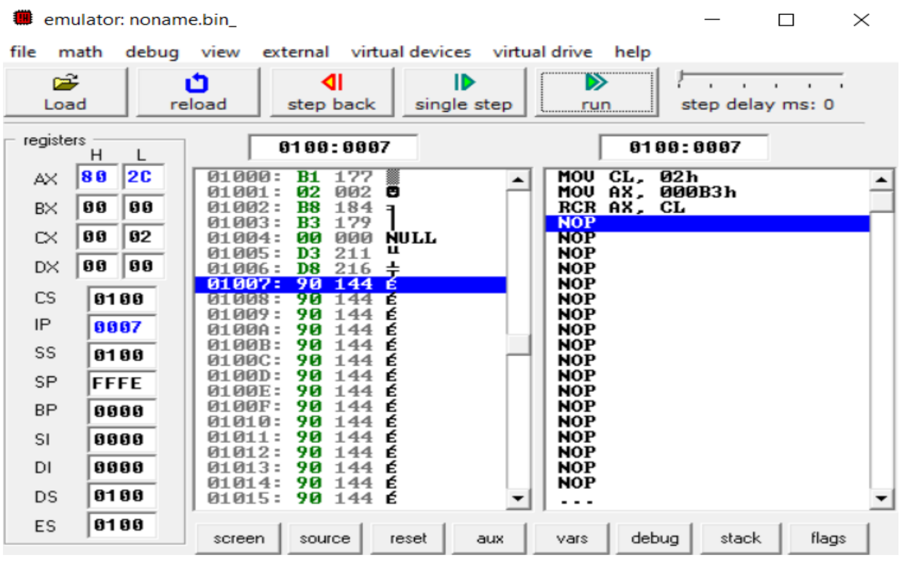
```
registers        0100:0002                      0100:0002
    H  L
AX  00 00        01000: B1 177 ▒                MOV  CL, 02h
BX  00 00        01001: 02 002 ☻                MOV  AX, 000B3h
CX  00 02        01002: B8 184 ╕                RCR  AX, CL
DX  00 00        01003: B3 179 │                NOP
CS  0100         01004: 00 000 NULL             NOP
IP  0002         01005: D3 211 ╙                NOP
SS  0100         01006: D8 216 ╪                NOP
SP  FFFE         01007: 90 144 É                NOP
BP  0000         01008: 90 144 É                NOP
SI  0000         01009: 90 144 É                NOP
DI  0000         0100A: 90 144 É                NOP
DS  0100         0100B: 90 144 É                NOP
ES  0100         0100C: 90 144 É                NOP
                 0100D: 90 144 É                NOP
                 0100E: 90 144 É                NOP
                 0100F: 90 144 É                NOP
                 01010: 90 144 É                NOP
                 01011: 90 144 É                NOP
                 01012: 90 144 É                NOP
                 01013: 90 144 É                NOP
                 01014: 90 144 É                NOP
                 01015: 90 144 É                ...
```

Fig 3.22: Emulator output for RCR (After Step 1)


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
registers        0100:0005                      0100:0005
    H  L
AX  00 B3        01000: B1 177 ▒                MOV  CL, 02h
BX  00 00        01001: 02 002 ☻                MOV  AX, 000B3h
CX  00 02        01002: B8 184 ╕                RCR  AX, CL
DX  00 00        01003: B3 179 │                NOP
CS  0100         01004: 00 000 NULL             NOP
IP  0005         01005: D3 211 ╙                NOP
SS  0100         01006: D8 216 ╪                NOP
SP  FFFE         01007: 90 144 É                NOP
BP  0000         01008: 90 144 É                NOP
SI  0000         01009: 90 144 É                NOP
DI  0000         0100A: 90 144 É                NOP
DS  0100         0100B: 90 144 É                NOP
ES  0100         0100C: 90 144 É                NOP
                 0100D: 90 144 É                NOP
                 0100E: 90 144 É                NOP
                 0100F: 90 144 É                NOP
                 01010: 90 144 É                NOP
                 01011: 90 144 É                NOP
                 01012: 90 144 É                NOP
                 01013: 90 144 É                NOP
                 01014: 90 144 É                NOP
                 01015: 90 144 É                ...
```

Fig 3.23: Emulator output for RCR (After Step 2)

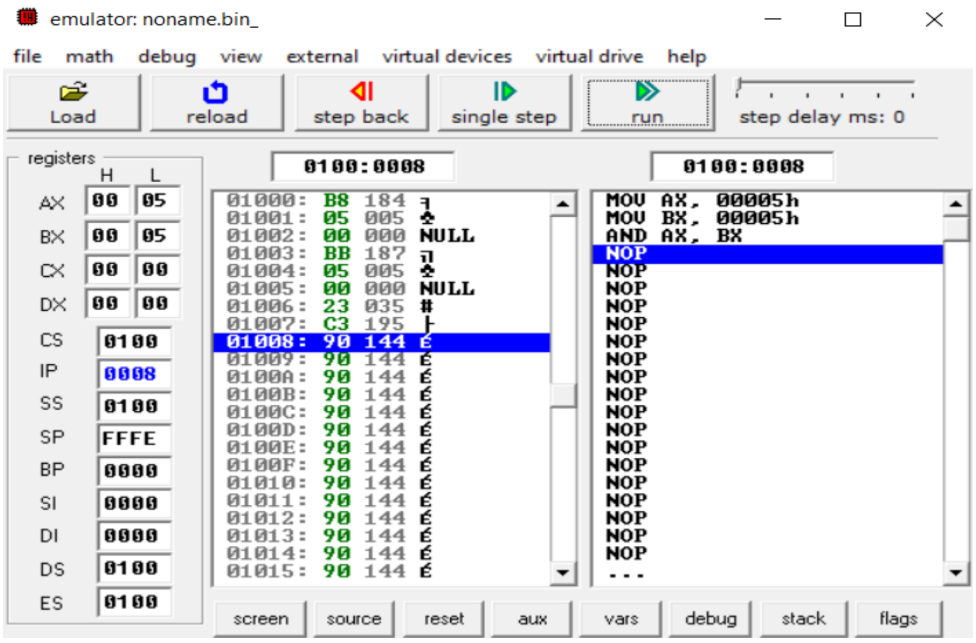

emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
registers        0100:0007                      0100:0007
    H  L
AX  80 2C        01000: B1 177 ▒                MOV  CL, 02h
BX  00 00        01001: 02 002 ☻                MOV  AX, 000B3h
CX  00 02        01002: B8 184 ╕                RCR  AX, CL
DX  00 00        01003: B3 179 │                NOP
CS  0100         01004: 00 000 NULL             NOP
IP  0007         01005: D3 211 ╙                NOP
SS  0100         01006: D8 216 ╪                NOP
SP  FFFE         01007: 90 144 É                NOP
BP  0000         01008: 90 144 É                NOP
SI  0000         01009: 90 144 É                NOP
DI  0000         0100A: 90 144 É                NOP
DS  0100         0100B: 90 144 É                NOP
ES  0100         0100C: 90 144 É                NOP
                 0100D: 90 144 É                NOP
                 0100E: 90 144 É                NOP
                 0100F: 90 144 É                NOP
                 01010: 90 144 É                NOP
                 01011: 90 144 É                NOP
                 01012: 90 144 É                NOP
                 01013: 90 144 É                NOP
                 01014: 90 144 É                NOP
                 01015: 90 144 É                ...
```

Fig 3.24: Emulator output for RCR (After Step 3)

- **AND**

MOV AX, 5
MOV BX, 5
AND AX, BX



```
registers
     H   L
AX   00  05
BX   00  05
CX   00  00
DX   00  00
CS   0100
IP   0008
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

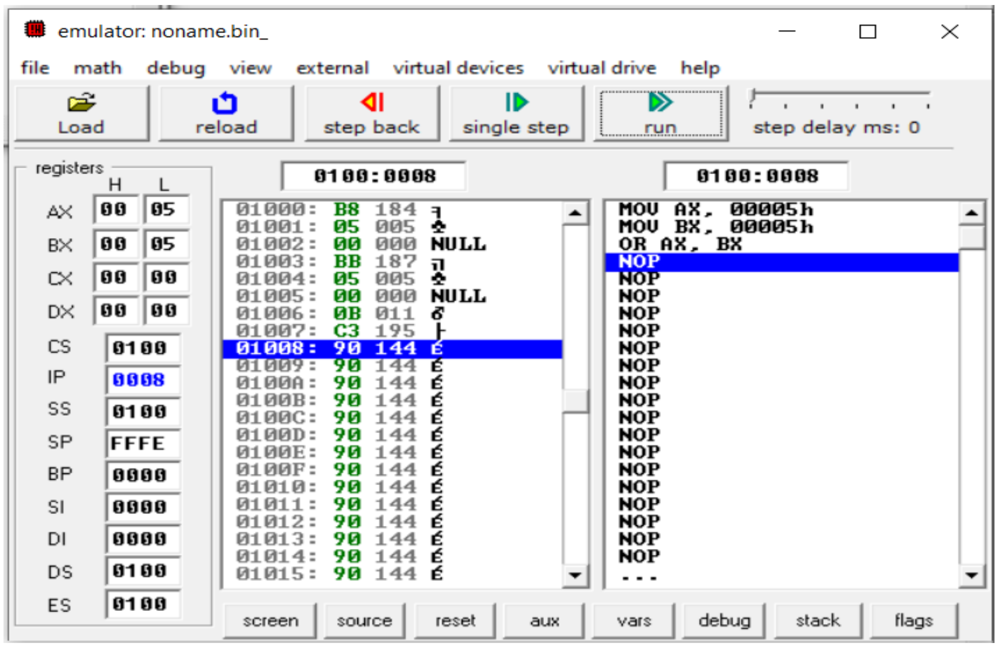
```
0100:0008
01000: B8 184 ╕
01001: 05 005 ♣
01002: 00 000 NULL
01003: BB 187 ╗
01004: 05 005 ♣
01005: 00 000 NULL
01006: 23 035 #
01007: C3 195 ├
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0008
MOV  AX, 00005h
MOV  BX, 00005h
AND  AX, BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.25: Emulator output for AND

- **OR**

MOV AX, 5
MOV BX, 5
OR AX, BX



```
registers
     H   L
AX   00  05
BX   00  05
CX   00  00
DX   00  00
CS   0100
IP   0008
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

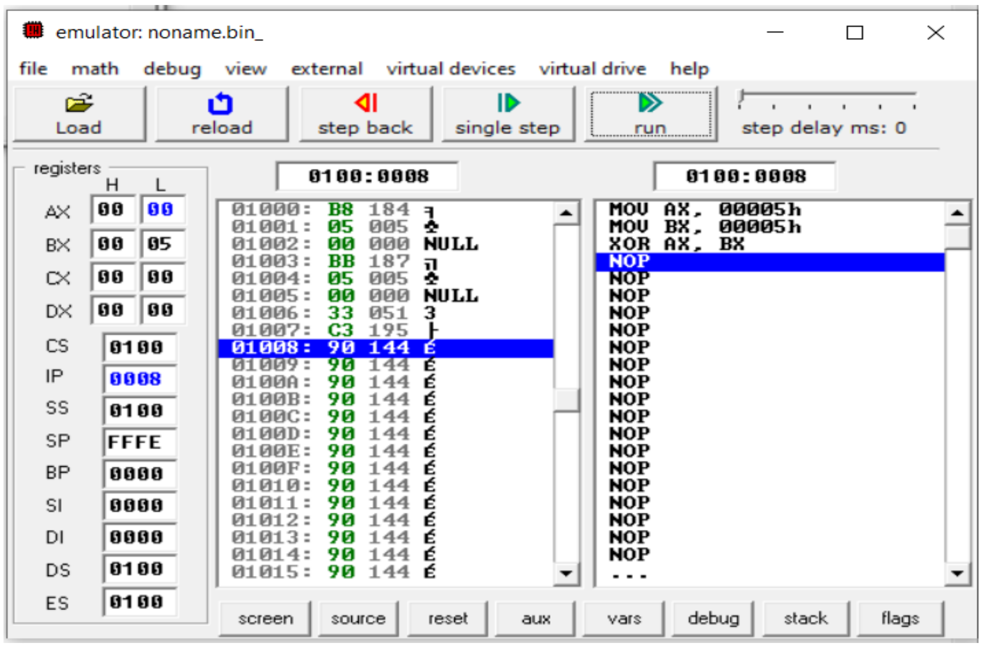
```
0100:0008
01000: B8 184 ╕
01001: 05 005 ♣
01002: 00 000 NULL
01003: BB 187 ╗
01004: 05 005 ♣
01005: 00 000 NULL
01006: 0B 011 ♂
01007: C3 195 ├
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0008
MOV  AX, 00005h
MOV  BX, 00005h
OR  AX, BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.26: Emulator output for OR

- **XOR**

MOV AX, 5
MOV BX, 5
XOR AX, BX


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
screen source reset aux vars debug stack flags

```
    H  L
AX  00 00
BX  00 05
CX  00 00
DX  00 00
CS  0100
IP  0008
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

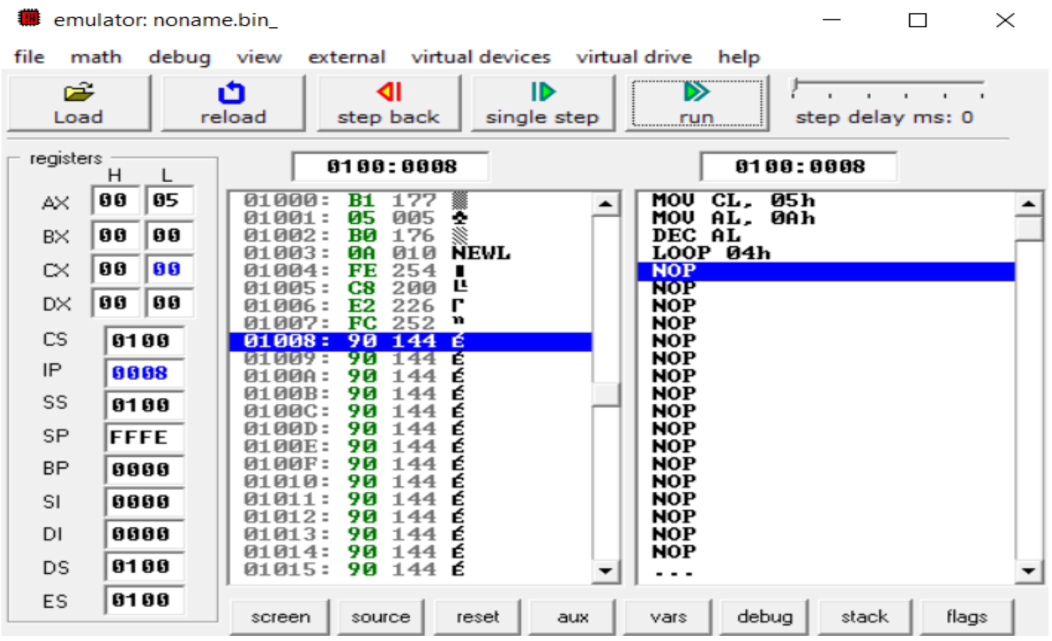
```
0100:0008
01000: B8 184 ╕
01001: 05 005 ♣
01002: 00 000 NULL
01003: BB 187 ╗
01004: 05 005 ♣
01005: 00 000 NULL
01006: 33 051 3
01007: C3 195 ├
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0008
MOV AX, 00005h
MOV BX, 00005h
XOR AX, BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.27: Emulator output for XOR

- **LOOP**

MOV CL, 5
MOV AL, 10
TOP:
DEC AL
LOOP TOP


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
screen source reset aux vars debug stack flags

```
    H  L
AX  00 05
BX  00 00
CX  00 00
DX  00 00
CS  0100
IP  0008
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:0008
01000: B1 177 ▒
01001: 05 005 ♣
01002: B0 176 ░
01003: 0A 010 NEWL
01004: FE 254 ■
01005: C8 200 ╚
01006: E2 226 Γ
01007: FC 252 ⁿ
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0008
MOV CL, 05h
MOV AL, 0Ah
DEC AL
LOOP 04h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig 3.27: Emulator output for LOOP

3.6 Discussion & Conclusion

In this lab, we practiced assembly language programming on the emu8086 for 1-byte or 2-byte data. We also studied about arithmetic and shift-rotate programming languages.

The shift-left command moves the bit to the left, while the shift-right instruction moves it to the right. The only distinction between shift and rotate instructions is that rotate instructions cycle the bits. While shift rotates the parts out, around goes out one side and comes in the other. either side, leaving the region where the rotated bits were either unmodified or zeroes.

We also observed the process of AND, OR, and X-OR operation in the program. Finally, the objective of the experiment was achieved.