

Data transfer instructions of 8086 microprocessor

General purpose byte or word transfer instructions:

- **MOV:** copy byte or word from specified source to specified destination
- **PUSH:** copy specified word to top of stack.
- **POP:** copy word from top of stack to specified location
- **PUSHA:** copy all registers to stack
- **POPA:** copy words from stack to all registers.
- **XCHG:** Exchange bytes or exchange words

These are I/O port transfer instructions:

- **IN:** copy a byte or word from specific port to accumulator
- **OUT:** copy a byte or word from accumulator to specific port

Special address transfer Instructions:

- **LEA:** load effective address of operand into specified register
- **LDS:** load DS register and other specified register from memory
- **LES:** load ES register and other specified register from memory

Flag transfer instructions:

- **LAHF:** load AH with the low byte of flag register
- **SAHF:** Stores AH register to low byte of flag register
- **PUSHF:** copy flag register to top of stack
- **POPF:** copy top of stack word to flag register

<p>PUSH</p>	<p>REG SREG memory immediate</p>	<p>Store 16 bit value in the stack.</p> <p>Note: PUSH immediate works only on 80186 CPU and later!</p> <p>Algorithm:</p> <ul style="list-style-type: none"> • $SP = SP - 2$ • $SS:[SP]$ (top of the stack) = operand <p>Example: MOV AX, 1234h PUSH AX POP DX ; DX = 1234h RET</p> <table border="1" data-bbox="491 1861 687 1962"> <tr> <td>C</td> <td>Z</td> <td>S</td> <td>O</td> <td>P</td> <td>A</td> </tr> <tr> <td colspan="6">unchanged</td> </tr> </table>	C	Z	S	O	P	A	unchanged					
C	Z	S	O	P	A									
unchanged														

POP

REG
SREG
memory

Get 16 bit value from the stack.

Algorithm:

- operand = SS:[SP] (top of the stack)
- SP = SP + 2

Example:

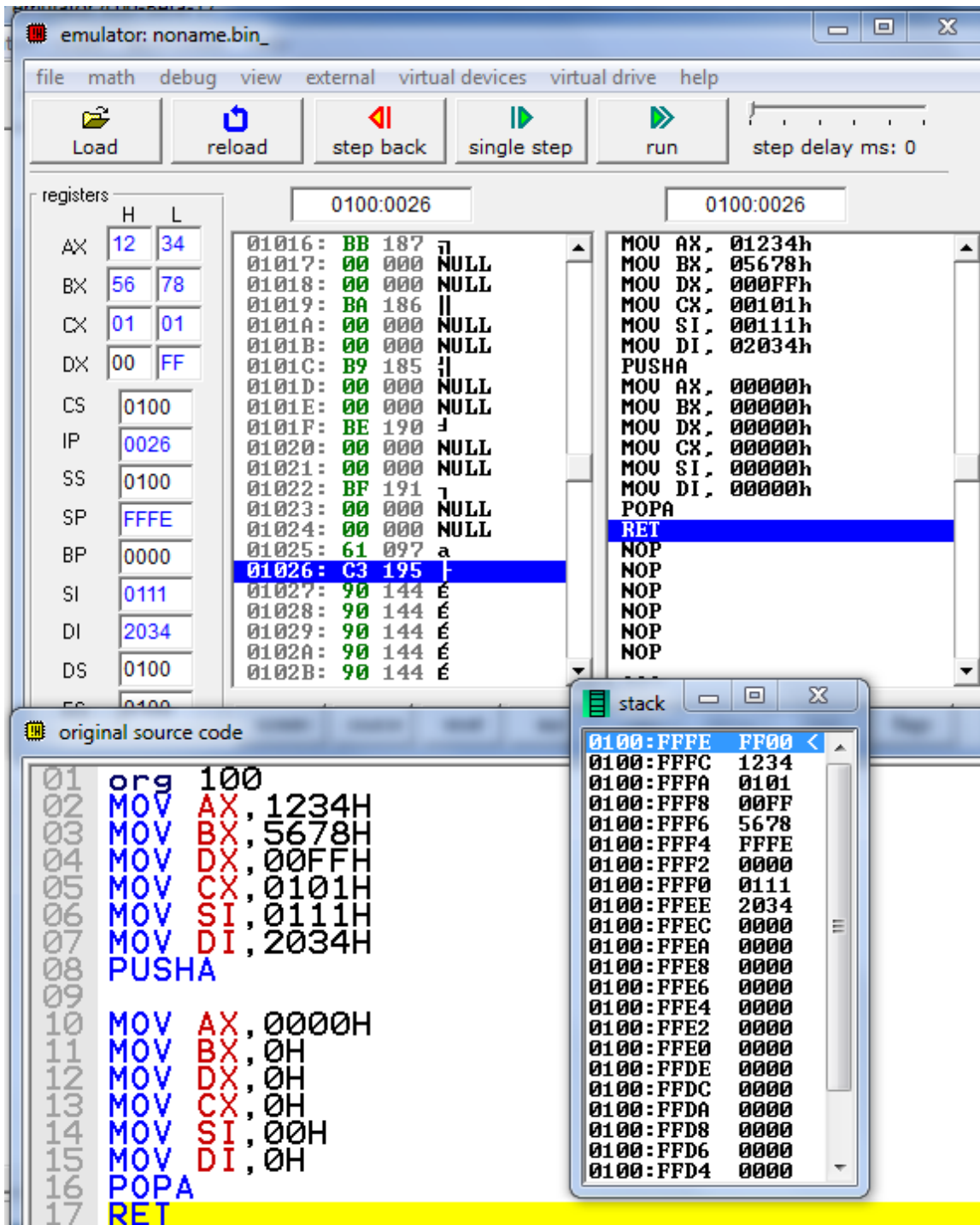
```
MOV AX, 1234h
PUSH AX
POP DX ; DX = 1234h
RET
```

C	Z	S	O	P	A
unchanged					

The screenshot shows an emulator window with the following components:

- Registers:** A table showing the state of registers. The stack pointer (SP) is at address 0000 with value FFFE. The instruction pointer (IP) is at 0017. The current instruction being executed is POP DX, with the comment "; DX = 1234h".
- Original Source Code:** A list of assembly instructions. Line 09 is highlighted in yellow: POP DX ; DX = 1234h. Line 14 is also highlighted in yellow: RET.
- Stack:** A window showing the stack contents. The top of the stack is at address 0100:FFFE with value FF00. The stack grows downwards.
- Random Access Memory:** A window showing memory contents. The address 0100:2000 is selected, and the value is 12 018. The memory contains several NULL values.

<p>PUSHA</p>	<p>No operands</p>	<p>Push all general purpose registers AX, CX, DX, BX, SP, BP, SI, DI in the stack. Original value of SP register (before PUSHA) is used.</p> <p>Note: this instruction works only on 80186 CPU and later!</p> <p>Algorithm:</p> <ul style="list-style-type: none"> • PUSH AX • PUSH CX • PUSH DX • PUSH BX • PUSH SP • PUSH BP • PUSH SI • PUSH DI <table border="1" data-bbox="464 887 660 987"> <tr> <td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td> </tr> <tr> <td colspan="6">unchanged</td> </tr> </table>	C	Z	S	O	P	A	unchanged					
C	Z	S	O	P	A									
unchanged														
<p>POPA</p>	<p>No operands</p>	<p>Pop all general purpose registers DI, SI, BP, SP, BX, DX, CX, AX from the stack. SP value is ignored, it is Popped but not set to SP register).</p> <p>Note: this instruction works only on 80186 CPU and later!</p> <p>Algorithm:</p> <ul style="list-style-type: none"> • POP DI • POP SI • POP BP • POP xx (SP value ignored) • POP BX • POP DX • POP CX • POP AX <table border="1" data-bbox="464 1664 660 1765"> <tr> <td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td> </tr> <tr> <td colspan="6">unchanged</td> </tr> </table>	C	Z	S	O	P	A	unchanged					
C	Z	S	O	P	A									
unchanged														



XCHG

REG, memory
memory, REG
REG, REG

Exchange values of two operands.

Algorithm:

operand1 < - > operand2

Example:

```
MOV AL, 5
MOV AH, 2
XCHG AL, AH ; AL = 2, AH = 5
XCHG AL, AH ; AL = 5, AH = 2
RET
```

C Z S O P A

unchanged

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

AX	H	L
	02	05
BX	55	66
CX	00	00
DX	11	77
CS	0100	
IP	FF00	
SS	0100	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

0100:FF00 0100:FF00

10F00: F4 244 ↑ HLT

10F01: 00 000 NULL ADD [BX + SI], AL

10F02: 00 000 NULL ADD [BX + SI], AL

10F03: 00 000 NULL ADD [BX + SI], AL

10F04: 00 000 NULL ADD [BX + SI], AL

10F05: 00 000 NULL ADD [BX + SI], AL

10F06: 00 000 NULL ADD [BX + SI], AL

10F07: 00 000 NULL ADD [BX + SI], AL

10F08: 00 000 NULL ADD [BX + SI], AL

10F09: 00 000 NULL ADD [BX + SI], AL

10F0A: 00 000 NULL ADD [BX + SI], AL

10F0B: 00 000 NULL ADD [BX + SI], AL

10F0C: 00 000 NULL ADD [BX + SI], AL

10F0D: 00 000 NULL ADD [BX + SI], AL

10F0E: 00 000 NULL ADD [BX + SI], AL

10F0F: 00 000 NULL ADD [BX + SI], AL

10F10: 00 000 NULL ADD [BX + SI], AL

10F11: 00 000 NULL ADD [BX + SI], AL

10F12: 00 000 NULL ADD [BX + SI], AL

10F13: 00 000 NULL ADD [BX + SI], AL

10F14: 00 000 NULL ADD [BX + SI], AL

10F15: 00 000 NULL ...

screen source reset aux vars debug stack flags

```
01 org 100
02 MOV AL, 5
03 MOV AH, 2
04 XCHG AL, AH ; AL = 2, AH = 5
05 XCHG AL, AH ; AL = 5, AH = 2
06 MOV [2000H], 45H
07 MOV BX, 1177H
08 MOV DX, 5566H
09 XCHG BX, [2000H]
10 XCHG [2000H], BX
11 XCHG DX, BX
12 RET
13
14
15
16
```

Random Access Memory

0100:2000 update

0100:2000:	45	069	E
0100:2001:	00	000	NULL
0100:2002:	00	000	NULL
0100:2003:	00	000	NULL
0100:2004:	00	000	NULL
0100:2005:	00	000	NULL
0100:2006:	00	000	NULL
0100:2007:	00	000	NULL
0100:2008:	00	000	HLT

IN	AL, im.byte AL, DX AX, im.byte AX, DX	<p>Input from port into AL or AX. Second operand is a port number. If required to access port number over 255 - DX register should be used.</p> <p>Example: IN AX, 4</p> <table border="1" data-bbox="459 506 655 607"> <tr> <td>C</td> <td>Z</td> <td>S</td> <td>O</td> <td>P</td> <td>A</td> </tr> <tr> <td colspan="6">unchanged</td> </tr> </table>	C	Z	S	O	P	A	unchanged					
C	Z	S	O	P	A									
unchanged														
OUT	im.byte, AL im.byte, AX DX, AL DX, AX	<p>Output from AL or AX to port. First operand is a port number. If required to access port number over 255 - DX register should be used.</p> <p>Example: MOV AX, 0FFFh ; Turn on all OUT 4, AX ; traffic lights.</p> <p>MOV AL, 100b ; Turn on the third OUT 7, AL ; magnet of the stepper-motor.</p> <table border="1" data-bbox="459 1028 655 1128"> <tr> <td>C</td> <td>Z</td> <td>S</td> <td>O</td> <td>P</td> <td>A</td> </tr> <tr> <td colspan="6">unchanged</td> </tr> </table>	C	Z	S	O	P	A	unchanged					
C	Z	S	O	P	A									
unchanged														

ENG.YOUSSEF.MH

emulator: noname.bin_

file math debug view external virtual devices

Load reload step back single step

registers

	H	L
AX	00	04
BX	00	00
CX	00	00
DX	00	00
CS	0100	
IP	0019	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

0100:0019

```

01017: E7 231 τ
01018: 04 004 ♦
01019: F4 244 ↑
0101A: 90 144 E
0101B: 90 144 E
0101C: 90 144 E
0101D: 90 144 E
0101E: 90 144 E
0101F: 90 144 E
01020: 90 144 E
01021: 90 144 E
01022: 90 144 E
01023: 90 144 E

```

Traffic Lights

FEDCBA9876543210
0000000000001001

port 4 (word - 16 bits)

original source code

```

01
02
03 mov ax, 1234
04
05
06 out 199, ax
07 MOV AX, 0
08 IN AX, 199
09 mov ax, -5678
10 out 199, ax
11
12 MOV AX, 0FFFh ; Turn on all
13 OUT 4, AX ; traffic ligh
14 MOV AX, 4
15 OUT 4, AX
16
17 hlt
18
19

```

LED d...

-05678

port 199 (2 bytes)