Logical and Shift Practice

eax	**ebx**	**ecx**	**edx**
`0x01234567`	`0x89ABCDEF`	`0xFEDCBA98`	`0x76543210`

Fill in the following table showing the effects of the indicated instructions. Also, show the values of the **CF** and **OF** flag *after* the instruction executes. Assume each instruction is independent of the others.

	Instruction	New Value (in hex)	CF	OF
1	`andl  $-16, %eax`			
2	`andl  $-16, %edx`			
3	`testl $-16, %edx`			
4	`orl   $-16, %ecx`			
5	`xorw  $-1, %ax`			
6	`notw  %ax`			
7	`negw  %ax`			
8	`shrw  $1, %bx`			
9	`shlw  $1, %bx`			
A	`sall  $4, %bx`			
B	`sarl  $4, %bx`			
C	`salxl $4, %bx`			

Logical and Shift Instructions

`ands ors xors`

The **OF** and **CF** flags are set to `0`; the **SF**, **ZF**, and **PF** flags are set according to the result.

`nots`

Bits that are `1` become `0`, bits that are `0` become `1`.
One's complement. Flags unaffected.

`negs`

Replaces operand with its two's complement.
The **CF** flag set to `0` if the source operand is `0`; is otherwise set to `1`.
The **OF** flag set to `1` if the source operand is `INT_MIN`; is otherwise set to `0`. The **SF**, **ZF**, **AF**, and **PF** flags are set according to the result.

`shrs sars shls sals`

Shifts the bits in the first destination operand to the left or right by the number of bits specified in the source operand. Spaces are filled with `0` *except* in the case of `sars`, which fills spaces with the sign bit of the original operand. The **CF** is set to the last bit that was shifted out. The **SF**, **ZF**, **AF**, and **PF** flags are set according to the result. **OF** is unaffected for all but `1`-bit shifts, in which case:

- `shrs`
  - **OF** Set to sign of original operand
- `sars`
  - **OF** Set to `0`
- `shls sals`
  - **OF** Set to `0` if most significant two bits of operand were the same, otherwise set to `1`

`shrxs sarxs shlxs salxs`

Same as above, but flags are unaffected.

`rcrs rcls rors rols`

Expert mode***!!!***