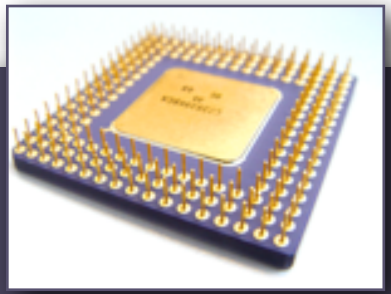
Bitwise Operations (Part 1)

§6.2

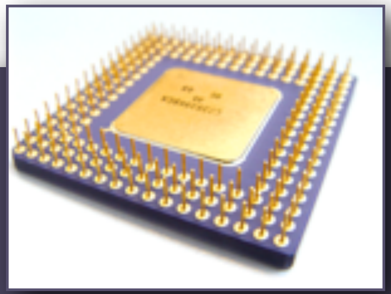
Motivation: Flags



- The EFLAGS register holds 32 bits, like other registers
- However, each bit corresponds to a different flag:

Bit 0:	Carry flag	Status flag
Bit 2:	Parity flag	Status flag
Bit 4:	Auxiliary carry flag	Status flag
Bit 6:	Zero flag	Status flag
Bit 7:	Sign flag	Status flag
Bit 9:	Interrupt enable flag	Control flag
Bit 10:	Direction flag	Control flag
Bit 11:	Overflow flag	Status flag

Motivation: Flags

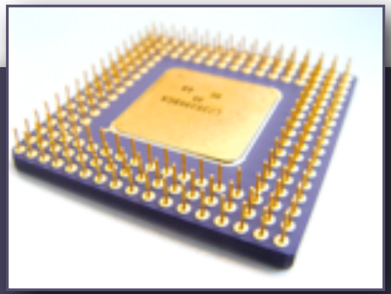


- *Example:*

EFLAGS = 00000A92h
= 00000000 00000000 00001010 10010010b

- Bit 0: Carry flag = 0
- Bit 2: Parity flag = 0
- Bit 4: Auxiliary carry flag = 1
- Bit 6: Zero flag = 0
- Bit 7: Sign flag = 1
- Bit 9: Interrupt enable flag = 1
- Bit 10: Direction flag = 0
- Bit 11: Overflow flag = 1

Motivation: Flags

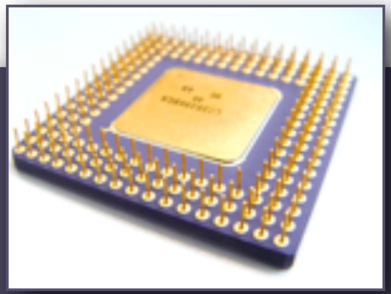


- One way to copy the value of EFLAGS into EAX:

  -
    ```
    pushfd
    pop eax
    ```
  - *But how to determine if a particular bit in EAX is set?*

- ...and then set the value of EFLAGS from EAX:
  - *(put desired value in EAX)*
    ```
    push eax
    popfd
    ```
  - *Copy current value into EAX, then set/clear desired bits*
  - *But how to set/clear individual bits in EAX?*

Review from ELEC 2200/2210



Activity 13 #1

- Recall the basic Boolean/logical operations:

AND

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

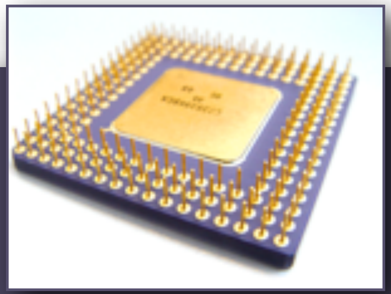
XOR

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

NOT

X	$\neg X$
F	T
T	F

Bitwise Operations



- Boolean operations (AND, OR, XOR, NOT) can be applied *bitwise,* i.e., applied to every bit:

Activity 13 #2

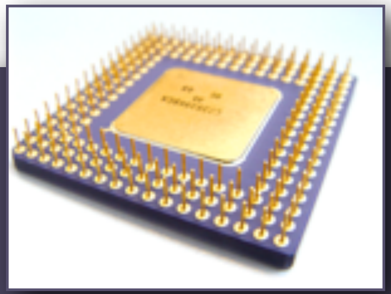
```
     0 0 1 1 1 0 1 1
AND  0 0 0 0 1 1 1 1
     ---------------
     0 0 0 0 1 0 1 1
```

```
     0 0 1 1 1 0 1 1
XOR  0 0 0 0 1 1 1 1
     ---------------
     0 0 1 1 0 1 0 0
```

```
     0 0 1 1 1 0 1 1
OR   0 0 0 0 1 1 1 1
     ---------------
     0 0 1 1 1 1 1 1
```

```
NOT  0 0 1 1 1 0 1 1
     ---------------
     1 1 0 0 0 1 0 0
```

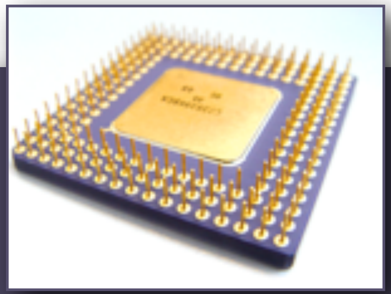
Topics Covered in Notes:



- AND, OR, XOR instruction
- NOT instruction

Activity 13 #3–4

Bit Masks & Testing Bits

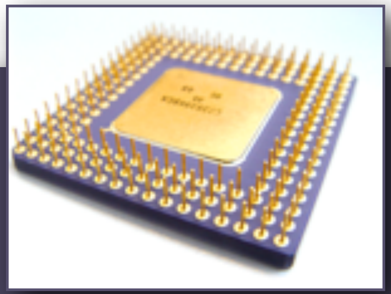


- *bit mask:* a binary integer value (usually a constant) that is combined with another value using a bitwise operation in order to extract, set, or clear particular bits
  - Like using masking tape when painting

- E.g., how to tell if a number has bit 3 set?

  -
    ```
       10011110        01110011        00000000
     & 00001000      & 00001000      & 00001000
       00001000        00000000        00000000
    ```
  - Bitwise AND the number with the bit mask 00001000b, then check whether the result is nonzero

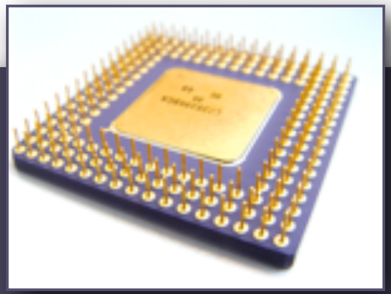
Bit Masks & Testing Bits



- E.g., how to tell if a number has either bit 0 or 3 set?
  - Activity 13 #5

  -
    ```
      10011110      01110011      00000000
    & 00001001    & 00001001    & 00001001
      00001000      00000001      00000000
    ```
  - Bitwise AND the number with the bit mask 00001001b, then check whether the result is nonzero
- Examples of bit masks with OR and XOR later...

TEST Instruction



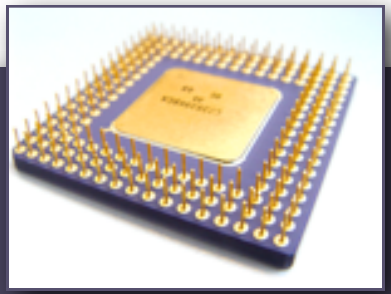
- Performs a nondestructive AND operation between each pair of matching bits in two operands
- No operands are modified, but the Zero flag is affected.
- Example: jump to a label if either bit 0 or bit 1 in AL is set.

  -
    ```
    test al, 00000011b
    jnz  ValueFound
    ```
- Example: jump to a label if neither bit 0 nor bit 1 in AL is set.

  -
    ```
    test al, 00000011b
    jz   ValueNotFound
    ```

Activity 13 #6

Topics Covered in Notes:



- TEST instruction