# Sign Extensions
# Comparing Signed Numbers

Ned Nedialkov

McMaster University
Canada

SE 3F03
February 2014

# Outline

# Signed and unsigned integers

## Examples

- FFh as an unsigned integer is 255; as a signed integer it is $-1$
- $0011|1000_2 = 2^5 + 2^4 + 2^3 = 32 + 16 + 8 = 56$
- Two's complement representation: flip the bits and add 1

$$\begin{array}{rr} & 0011|1000 = 56 \\ & 1100|0111 \\ + & \qquad\quad 1 \\ \hline & 1100|1000 = -56 \end{array}$$

$$\begin{array}{rr} & 0011|0111 \\ + & \qquad\quad 1 \\ \hline & 0011|1000 = 56 \end{array}$$

# Two's complement representation

- Given a number in two's complement representation, if we flip all the bits and add 1, we get the two's-complement representation of the negative of that number.
- The largest positive number in a byte is $0111|1111 = 127$
- The smallest negative number is

$$\begin{array}{rr} & 1000|0000 \\ & 0111|1111 \\ + & \qquad\quad 1 \\ \hline & 1000|0000 \end{array} \begin{array}{l} = 128 \text{ as unsigned} \\ \\ \\ = -128 \text{ as signed} \end{array}$$

# Addition

| | |
|---|---|
| 11111\|111 | carry row |
| 0000\|1111 | $= 15$ |
| 1111\|1011 | $= -5$ |
| 10000\|1010 | $= 10$ |

- The carry bit is ignored
- Overflow: if the first two bits in the carry row are different

| | |
|---|---|
| 01111\|111 | carry row |
| 0111\|1111 | $= 127$ |
| 0000\|0101 | $= 5$ |
| 1000\|0100 | |
| 0111\|1011 | |

# Subtraction

- $a - b = a + (-b)$
- Done trough addition
- No need to examine signs

# Sign extension

## Decreasing size: remove the more significant bits

- **mov ax**, 0034h *;AH=00, AL = 34h*
  **mov cl**, **al**
- **mov ax**, FFFFh
  **mov cl**, **al**
  **cl** contains FFh, which is 255 as unsigned and -1 as signed
- Unsigned numbers: all removed bits must be 0's
- Signed numbers:
  - removed bits must be all 1's or all 0's
  - first bit not removed must be the same as removed bits

# Examples

- Assume FFFFh is reduced to FFh
  - as a signed number, `FFFFh = -1` becomes `FFh=-1`
  - as an unsigned number `FFFFh != FFh`
- Assume 0FFFh is reduced to FFh
  - as a signed number, `0FFFh` is positive but becomes `FFh=-1`
    - the removed bits are not the same
  - as an unsigned number `0FFFh != FFh`
    - then removed bits are different

## Increasing size

- Unsigned numbers: add 0's to the left
- Signed numbers:
  - extend the sign bit to the left
  - e.g. `FFh` becomes `FFFFh`
  - e.g. `1001` becomes `1111|1001`
    - `1001` is `0110+1=0111 = -7`
    - `1111|1001` is `0000|0110+1= 0000|0111=-7`
- To extend 8 bits to 16 bits, use e.g. **mov ah,** `0`
- To extend 16 bits to 32: we cannot access the upper part of **EAX**

## Unsigned integers

```
movz eax, ax      ; extends ax into eax
movz eax, al      ; extends al into eax
movz ax, al       ; extends al into ax
```

## Signed integers

```
cbw               ; extends al into ax
cwd               ; extends ax into dx:ax
cwde              ; extends ax into eax
cdq               ; extends eax into edx:eax
```

# Comparing sign numbers

- Consider comparing $a$ and $b$, where $a < b$
- We work in 8 bits
- Assume $a = -87$ and $b = 42$
- A compare instruction would compute $-87 + (-42)$

$$\begin{array}{rr} & 1010|1001 = -87 \text{ since} \\ & 0101|0110 \\ + & \qquad\quad 1 \\ \hline & 0101|0111 = 87 \end{array}$$

$$1101|0110 = -42$$

- $$\begin{array}{rr} & 1010|1001 = -87 \\ + & 1101|0110 = -42 \\ \hline & 10111|1111 \end{array}$$
- `SF=0`, `OF=1`
- If the result does not overlow, `SF=1`, `OF=0`

- Consider comparing $a > b$
- `SF=OF`
- How can we have an overlow? E.g. $a = 87$ and $b = -47$:

$$\begin{array}{rr} & 0101|0111 = 87 \\ + & 0010|1010 = 42 \\ \hline & 1000|0001 \end{array}$$

- `SF=1`
- Overflow: if the sign of the result is different when adding numbers of the same sign