Multiplication and Division

ICS312

Machine-Level and Systems Programming

Henri Casanova (henric@hawaii.edu)

Multiplication and Division

- On x86 the multiplication and division instructions are a bit cumbersome
- The point here is not for you to learn them by heart at all
  - In an exam situation the specification would be provided to you
- The point is to learn how to read their specification (this lectures will be very quick)

Multiplication

- There are two instructions to do multiplications
- Multiplying unsigned numbers: **`mul`**
- Multiplying signed numbers: **`imul`**
- Why do we need two different instructions?
- Consider the multiplication of FF by FF
  - If we assume unsigned quantities, this is 255*255 = 65035 = FE0Bh
  - If we assume signed quantities, this is -1 * -1 = 1 = 0001h
- So clearly we need two different instructions because we need to get two different results

The mul Instruction (unsigned)

- The size of the result of the multiplication can be much larger than the size of the operands (in number of bits)
  - Multiplications just leads to much bigger numbers than additions
  - At most the result will be twice the size of the operands (255 * 255 = 65,025, which is needs to be encoded with 2 bytes)
- So there is a larger risk of overflowing
- For this reason, the **`mul`** instructions impose that the result is encoded on more bits than the operands
- Consequently, it will **never overflow**
  - Although the carry bit may be set
- The **`mul`** instruction imposes that the result be encoded on twice as many bits as its operands

The mul Instruction (unsigned)

- It takes a single operand (that's not an immediate constant):

```
mul    src (register or memory reference)
```

operand	action
reg/mem8	AX = AL * operand
reg/mem16	DX:AX = AX * operand
reg/mem32	EDX:EAX = EAX * operand

- This table uses the typical way to specify operands
  - **reg16**: a 16-bit register
  - **immed8**: an 8-bit immediate operand (i.e., a number)
  - **mem16**: 16-bits of memory content
- Note the XXX:YYY notation: the n-bit value will be split over two n/2-bit registers

- **WARNING**: Multiplication can **overwrite** DX or EDX

The imul Instruction (signed)

- **`imul`** is fancier than **`mul`** and has three formats:

```
imul  src
imul  dst, src1
imul  dst, src1, src2
```

- The different combinations are shown in Table 2.2 in the textbook
- Let's look at it…

The imul instruction

Will not overflow (although the overflow bit may be set)

dst	src1	src2	action
	reg/mem8		AX = AL * src1
	reg/mem16		DX:AX = AX * src1
	reg/mem32		EDX:EAX = EAX*src1
reg16	reg/mem16		dst *= src1
reg32	reg/mem32		dst *= src1
reg16	immed8		dst *= immed8
reg32	immed8		dst *= immed8
reg16	immed16		dst *= immed16
reg32	immed32		dst *= immed32
reg16	reg/mem16	immed8	dst = src1*src2
reg32	reg/mem32	immed8	dst = src1*src2
reg16	reg/mem16	immed16	dst = src1*src2
reg32	reg/mem32	immed32	dst = src1*src2

Two Instructions for Division

- **`div`** for unsigned quantities
- **`idiv`** for signed quantities
- They perform integer division
  - e.g.,: 19 / 4 produces quotient = 4 remainder = 3
- Only one format for both:

```
div/idiv    src (register or memory reference)
```

src	action
reg/mem8	AL = AX / src and AH = AX mod src
reg/mem16	AX = DX:AX / src and DX = DX:AX mod src
reg/mem32	EAX = EDX:EAX / src and EDX = EDX:EAX mod src

- Warning: it's very common for programmers to forget initializing DX or EDX before the division!

Division Example

- Say I want to divide 2042 by 13 and I want to have quotient and remainder as 4-byte values
- Here is the code

```
mov         eax, 2042
mov         edx, 0      // IMPORTANT
mov         ecx, 13
idiv        ecx
// eax contains 157
// edx contains  1
// 2042 = 157 * 13 + 1
```

Example Program in Textbook

- Section 2.1.4 shows a sample program that uses all the arithmetic operations we just saw
- There is nothing particularly difficult about it, especially because overflows are not handled (so the numbers entered had better be "small")
- Make sure you go through that example and understand how it works
  - You may want to run it as well
- We'll ignore the content of Section 2.1.5 in the textbook

Conclusion

- Your job: make sure that if given both tables provided in these lecture notes you can program a multiplication or a division

- We will have an in-class quiz on the last 3 modules next week:
  - **This module**
  - **Casting module**
  - **Overflow module**