x86 architecture

Basic Arithmetic

Assembly language programming
By xorpd

xorpd.net

Objectives

- You will learn about some simple arithmetic instructions:
  - INC, DEC – Increase and decrease.
  - MUL – Multiplication.
  - DIV – Division.
- You will get a feeling of working with x86 instructions. We are not going to run any code yet.

INC, DEC

- INC, DEC instructions allow to increase or decrease numbers (by 1) respectively.
  - INC destination or DEC destination.
  - A wraparound occurs if the number is too large or too small. The wraparound, as usual, is according to the size of the argument.
- Examples:
  - `inc eax`
    - Increases eax by 1. ($eax \leftarrow eax + 1$).
  - `dec si`
    - Decreases si by 1. ($si \leftarrow si - 1$).
  - Invalid example: `inc 1C5h`
    - Where will the result be stored? Invalid opcode.

INC, DEC - Example

Instruction	eax
	FFFFFFFE
inc eax	
inc al	
dec al	
inc ax	
dec ax	
inc eax	
inc eax	

INC, DEC - Example

Instruction	eax
	FFFFFFFE
inc eax	FFFFFFFF
inc al	
dec al	
inc ax	
dec ax	
inc eax	
inc eax	

INC, DEC - Example

Instruction	eax
	FFFFFFFE
inc eax	FFFFFFFF
inc al	FFFFFF00
dec al	
inc ax	
dec ax	
inc eax	
inc eax	

INC, DEC - Example

Instruction	eax
	FFFFFFFE
inc eax	FFFFFFFF
inc al	FFFFFF00
dec al	FFFFFFFF
inc ax	
dec ax	
inc eax	
inc eax	

INC, DEC - Example

Instruction	eax
	FFFFFFFE
inc eax	FFFFFFFF
inc al	FFFFFF00
dec al	FFFFFFFF
inc ax	FFFF0000
dec ax	
inc eax	
inc eax	

INC, DEC - Example

Instruction	eax
	FFFFFFFE
inc eax	FFFFFFFF
inc al	FFFFFF00
dec al	FFFFFFFF
inc ax	FFFF0000
dec ax	FFFFFFFF
inc eax	
inc eax	

INC, DEC - Example

Instruction	eax
	FFFFFFFE
inc eax	FFFFFFFF
inc al	FFFFFF00
dec al	FFFFFFFF
inc ax	FFFF0000
dec ax	FFFFFFFF
inc eax	00000000
inc eax	

INC, DEC - Example

Instruction	eax
	FFFFFFFE
inc eax	FFFFFFFF
inc al	FFFFFF00
dec al	FFFFFFFF
inc ax	FFFF0000
dec ax	FFFFFFFF
inc eax	00000000
inc eax	00000001

MUL

- Allows to multiply numbers. (Unsigned multiplication)
  - MUL argument
  - Some forms:
    - $ax \leftarrow al \cdot argument$; If argument is of size 8 bits.
    - $dx{:}\,ax \leftarrow ax \cdot argument$; If argument is of size 16 bits.
    - $edx{:}\,eax \leftarrow eax \cdot argument$; If argument is of size 32 bits.
- edx:eax means: Bits concatenation of edx and eax.
- The size of the result is larger than the size argument. (Twice the amount of bits).

MUL (Cont.)

- Examples:
  - `mul ecx`
    - Multiplies eax by ecx and stores the result inside edx:eax. $(edx{:}\, eax \leftarrow eax \cdot ecx)$.
  - `mul si`
    - Multiplies ax by si and stores the result inside dx:ax. $(dx{:}\, ax \leftarrow ax \cdot si)$
  - `mul al`
    - Multiplies al by al and stores the result inside ax. $(ax \leftarrow al \cdot al)$.
  - Invalid example: `mul 2Ah`
    - There is no specific reason. There is no such opcode.

MUL - Example

Instruction	edx	eax	ecx
	AB1E2FFF	00000003	00000002
mul ecx			
mul ecx			
mov ax,0EEEEh			
mul ax			
mul cl			

MUL - Example

Instruction	edx	eax	ecx
	AB1E2FFF	00000003	00000002
mul ecx	00000000	00000006	00000002
mul ecx			
mov ax,0EEEEh			
mul ax			
mul cl			

MUL - Example

Instruction	edx	eax	ecx
	AB1E2FFF	00000003	00000002
mul ecx	00000000	00000006	00000002
mul ecx			
mov ax,0EEEEh			
mul ax			
mul cl			

MUL - Example

Instruction	edx	eax	ecx
	AB1E2FFF	00000003	00000002
mul ecx	00000000	00000006	00000002
mul ecx	00000000	0000000C	00000002
mov ax,0EEEEh			
mul ax			
mul cl			

MUL - Example

Instruction	edx	eax	ecx
	AB1E2FFF	00000003	00000002
mul ecx	00000000	00000006	00000002
mul ecx	00000000	0000000C	00000002
mov ax,0EEEEh	00000000	0000EEEE	00000002
mul ax			
mul cl			

MUL - Example

Instruction	edx	eax	ecx
	AB1E2FFF	00000003	00000002
mul ecx	00000000	00000006	00000002
mul ecx	00000000	0000000C	00000002
mov ax,0EEEEh	00000000	0000EEEE	00000002
mul ax	0000DEFF	00006544	00000002
mul cl			

MUL - Example

Instruction	edx	eax	ecx
	AB1E2FFF	00000003	00000002
mul ecx	00000000	00000006	00000002
mul ecx	00000000	0000000C	00000002
mov ax,0EEEEh	00000000	0000EEEE	00000002
mul ax	0000DEFF	00006544	00000002
mul cl			

MUL - Example

Instruction	edx	eax	ecx
	AB1E2FFF	00000003	00000002
mul ecx	00000000	00000006	00000002
mul ecx	00000000	0000000C	00000002
mov ax,0EEEEh	00000000	0000EEEE	00000002
mul ax	0000DEFF	00006544	00000002
mul cl	0000DEFF	00000088	00000002

MUL - Example

Instruction	edx	eax	ecx
	AB1E2FFF	00000003	00000002
mul ecx	00000000	00000006	00000002
mul ecx	00000000	0000000C	00000002
mov ax,0EEEEh	00000000	0000EEEE	00000002
mul ax	0000DEFF	00006544	00000002
mul cl	0000DEFF	00000088	00000002

Take a brake

- Take a break.
- Come back when you are ready for the DIV instruction.

DIV

- Divides one number by another number. (Unsigned division).
  - DIV arg
  - Opposite of MUL.
  - Some forms:
    - arg of size 8 bits:
      - $al \leftarrow ax \ / \ arg$; Quotient
      - $ah \leftarrow ax \ \% \ arg$; Remainder
    - arg of size 16 bits:
      - $ax \leftarrow dx{:}\, ax \ / \ arg$
      - $dx \leftarrow dx{:}\, ax \ \% \ arg$
    - arg of size 32 bits:
      - $eax \leftarrow edx{:}\, eax \ / \ arg$
      - $edx \leftarrow edx{:}\, eax \ \% \ arg$

DIV (Cont.)

- Examples:
  - `div ch`
    - Divides ax by ch and stores the division result inside al. The remainder is stored inside ah.
  - `div esi`
    - Divides edx:eax by esi. Stores the quotient inside eax. Stores the remainder inside edx.
  - `div di`
    - Divides dx:ax by di. Stores the quotient inside ax. Stores the remainder inside dx.
  - Invalid example: `div 5CAh`
    - There is no such opcode.

DIV (Cont.)

- Exceptions:
  - The processor raises **Exceptions** whenever something wrong happens while running your code.
  - The operation system and the processor work together to handle those exceptions.
- Exceptions raised by the DIV instruction:
  - Division by zero.
    - If we try to divide by zero, an exception is raised.
  - Quotient overflow.
    - If the quotient is too large, an exception is raised.

DIV - Example

Instruction	ecx	edx	eax
	00000002	00000000	00000008
div ecx			
inc ecx			
div ecx			

DIV - Example

Instruction	ecx	edx	eax
	00000002	00000000	00000008
div ecx	00000002	00000000	00000004
inc ecx			
div ecx			

DIV - Example

Instruction	ecx	edx	eax
	00000002	00000000	00000008
div ecx	00000002	00000000	00000004
inc ecx	00000003	00000000	00000004
div ecx			

DIV - Example

Instruction	ecx	edx	eax
	00000002	00000000	00000008
div ecx	00000002	00000000	00000004
inc ecx	00000003	00000000	00000004
div ecx	00000003	00000001	00000001

Remainder

4%3 = 1

Quotient

4 / 3 = 1

DIV – Example (2)

Instruction	ebx	edx	eax
	0000003A	00000020	00000000
div ebx			
div bx			
mov bl,0feh			
div bl			

DIV – Example (2)

Instruction	ebx	edx	eax
	0000003A	00000020	00000000
div ebx	0000003A	00000030	8D3DCB08
div bx			
mov bl,0feh			
div bl			

DIV – Example (2)

Instruction	ebx	edx	eax
	0000003A	00000020	00000000
div ebx	0000003A	00000030	8D3DCB08
div bx	0000003A	00000030	8D3DD75c
mov bl,0feh			
div bl			

DIV – Example (2)

Instruction	ebx	edx	eax
	0000003A	00000020	00000000
div ebx	0000003A	00000030	8D3DCB08
div bx	0000003A	00000030	8D3DD75c
mov bl,0feh	000000FE	00000030	8D3DD75c
div bl			

DIV – Example (2)

Instruction	ebx	edx	eax
	0000003A	00000020	00000000
div ebx	0000003A	00000030	8D3DCB08
div bx	0000003A	00000030	8D3DD75c
mov bl,0feh	000000FE	00000030	8D3DD75c
div bl	000000FE	00000030	8D3D0ED9

Summary

- INC,DEC – Increase and decrease by 1.
- MUL – Multiply numbers.
  - eax, ax, al (Result in [edx,eax], [dx,ax], [ah,al]).
- DIV – Divide numbers.
  - Divides edx:eax, dx:ax or ax

Exercises

- Some code reading.
- Some code writing.
- Do all the exercises, to make sure you grasp the new instructions we have just learned.
  - The bonus ones are not mandatory.

- Have fun :)