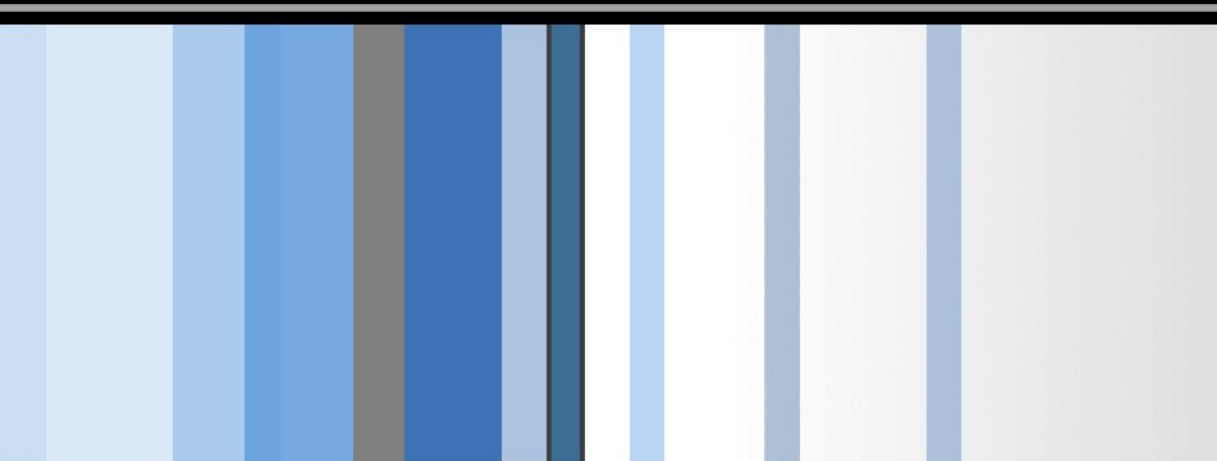# CSC 471 Modern Malware Analysis
X86 ASM

Si Chen (schen@wcupa.edu)

X86 ASM

MOV

- Move **reg/mem** value to **reg/mem**
  - mov A, B is "Move B to A" (A=B)
  - Same data size
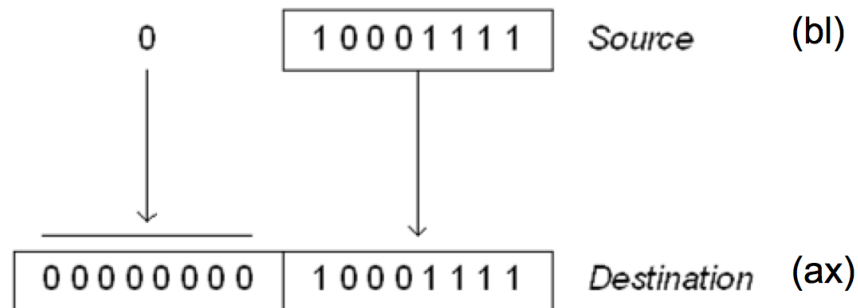
<div align="center">

mov eax, 0x1337
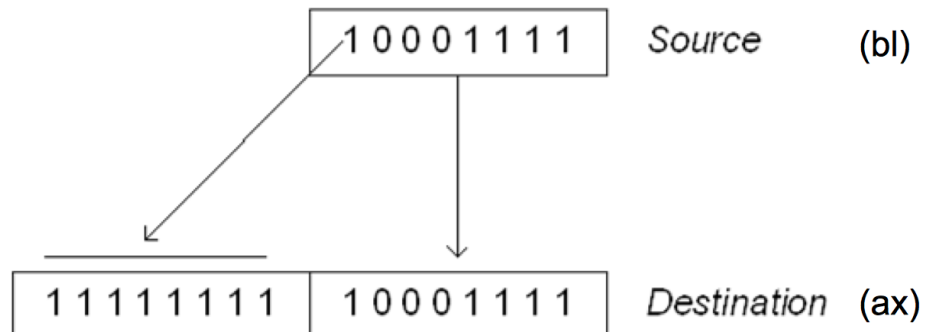
mov bx, ax

mov [esp+4], bl

</div>

MOVZX / MOVSX

- From small register to large register

- Zero-extend (MOVZX) / sign-extend (MOVSX)

- Example: movzx ebx, al

When copy a smaller value into a larger destination, MOVZX instruction fills (extends) the upper half of the destination with zeros

```
0          1 0 0 0 1 1 1 1    Source      (bl)


00000000 | 1 0 0 0 1 1 1 1    Destination  (ax)
```

MOVSX fills the upper half of the destination with a copy of the source operand's sign bit

```
           1 0 0 0 1 1 1 1    Source      (bl)


1 1 1 1 1 1 1 1 | 1 0 0 0 1 1 1 1    Destination  (ax)
```

West
Chester
University

More About Memory Access

- mov ebx, [esp + eax * 4] **Intel**

- mov (%esp, %eax, 4), %ebx **AT&T**

- mov BYTE [eax], 0x0f
 You must indicate the data size: BYTE/WORD/DWORD

ADD / SUB

- ADD / SUB

- Normallly "reg += reg" or "reg += imm"

- Data size should be equal

  - ADD eax, ebx

  - sub eax, 123

  - sub eax, BL ; Illegal

INC / DEC

- **inc, dec** — Increment, Decrement

- The **inc** instruction increments the contents of its operand by one. The **dec** instruction decrements the contents of its operand by one.

- *Syntax*
  inc <reg>
  inc <mem>
  dec <reg>
  dec <mem>

- *Examples*
  DEC EAX — subtract one from the contents of EAX.
  INC DWORD PTR [var] — add one to the 32-bit integer stored at location *var*

SHL / SHR / SAR

- Shift logical left / right

- Shift arithmetic right

- Common usage: **SHL eax, 2** (when calculate memory address)

mov eax, 0xA	0	0	0	0	1	0	1	0
shl eax, 2	0	0	1	0	1	0	0	0
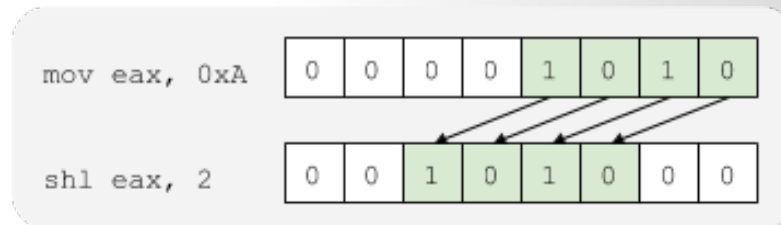
Jump

- Unconditional jump: jmp

- Conditional jump: je/jne
  and ja/jae/jb/jbe/jg/jge/jl/jle ...

- Sometime with "cmp A, B" -- compare these two values and set eflags

- Conditional jump is decided by some of the eflags bits.

ref

The JMP Instruction

- JMP (jump) instruction causes an **unconditional** jump
- Syntax is:

JMP	destination/target_label

- JMP can be used to get around the **range** restriction [126/127 byte]
- Flags – no change

```
TOP:

; body of the loop, say 2 instructions
DEC       CX         ; decrement counter
JNZ       TOP        ; keep looping if CX > 0
MOV       AX, BX
```

```
TOP:

; the loop body contains so many instructions
; that label TOP is out of range for JNZ. Solution is-
          DEC       CX
          JNZ       BOTTOM
          JMP       EXIT
BOTTOM:
          JMP       TOP
EXIT:
          MOV       AX, BX
```
Section 6-3: Assembly Language Programming

Unsigned and Signed Jumps.

Condition	Unsigned	Signed
source < dst	JB	JL
source <= dst	JBE	JLE
source ≠ dst	JNE(JNZ)	JNE(JNZ)
source = dst	JE(JZ)	JE(JZ)
source >= dst	JAE	JGE
source > dst	JA	JG

6

Jump

- ja/jae/jb/jbe are unsigned comparison
- jg/jge/jl/jle are signed comparison

Unsigned and Signed Jumps.

Condition	Unsigned	Signed
source < dest	JB	JL
source <= dest	JBE	JLE
source ≠ dest	JNE(JNZ)	JNE(JNZ)
source = dest	JE(JZ)	JE(JZ)
source >= dest	JAE	JGE
source > dest	JA	JG

6

CMP

- **cmp** — Compare

- Compare the values of the two specified operands, setting the condition codes in the machine status word appropriately. This instruction is equivalent to the sub instruction, except the result of the subtraction is discarded instead of replacing the first operand. *Syntax*
cmp <reg>,<reg>
cmp <reg>,<mem>
cmp <mem>,<reg>
cmp <reg>,<con>

- *Example*
cmp DWORD PTR [var], 10
jeq loop

- If the 4 bytes stored at location *var* are equal to the 4-byte integer constant 10, jump to the location labeled *loop*.

Q & A