IA-32 Architecture

Spring, 2018

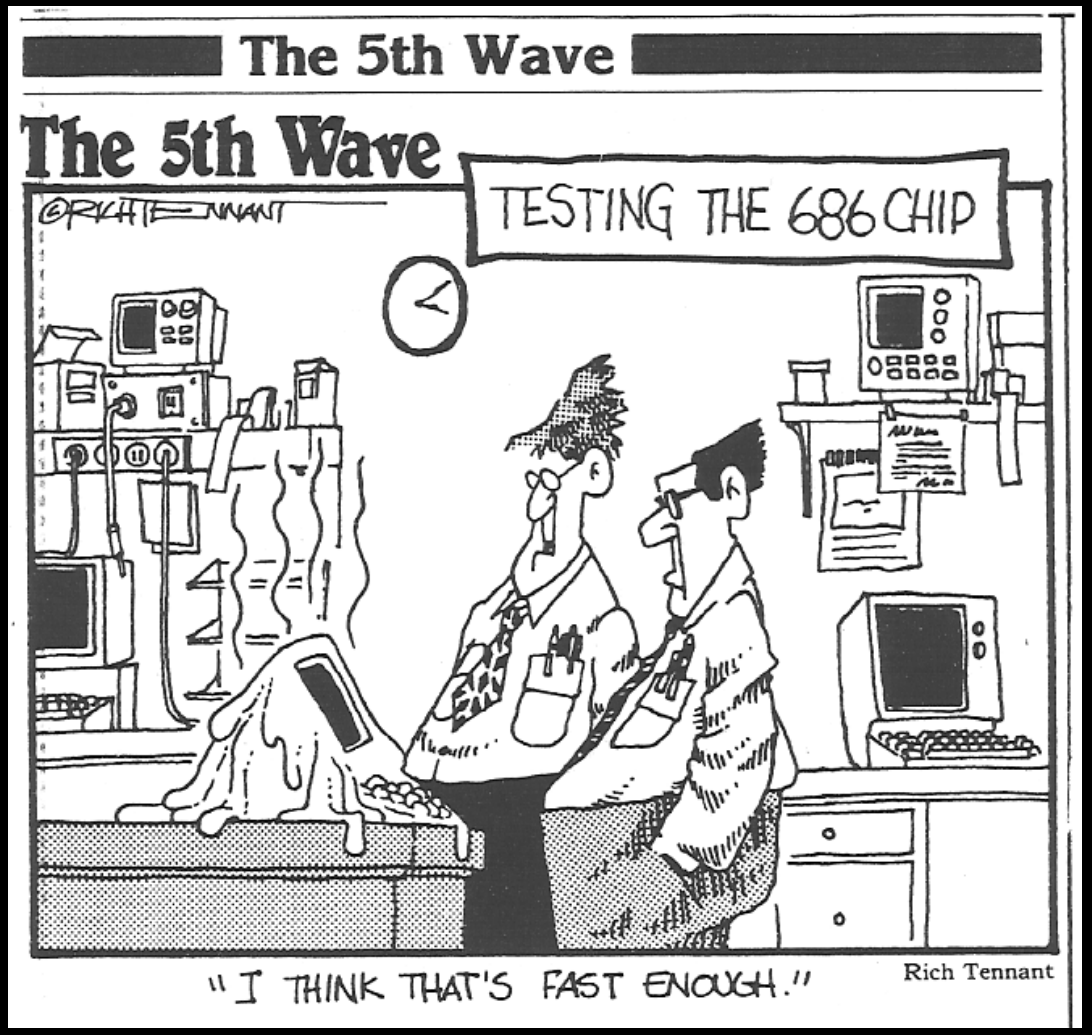
Intel x86 Architecture

- Malware and attacks are often based on assembly language
  - Security professionals constantly analyze assembly language code
  - Source code for applications and malware is not available in most cases
- We cover only the modern 32-bit view of the x86 architecture
- Web page handout has more details

Road map for these series of lectures

- What is IA-32 assembly languages and its basic syntax
  - IA-32 assembly languages and IA-32 architectures
- Where to find your variables and how to make functions call
  - Stacks and IA-32 calling convention
- How to write and analyze IA-32 programs
  - Tools: objdump, gdb, gcc etc
  - Assembly programming
- Attacks and defense
  - Buffer overflow, arc injection and ROP etc.

Intel Engineers



Road Map

- IA-32 assembly language syntax
- IA-32 architecture – registers
- Assembly language addressing modes
- From language to instructions
- Examples
- Where to find more information
- Summary

x86 Primer

- CISC architecture
  - Lots of instructions and addressing modes
  - Operands can be taken from memory
  - Instructions are variable length
    - Depends on operation
    - Depends on addressing modes
- Architecture manuals at:

  http://www.intel.com/products/processor/manuals/index.htm

X86 Assembly Language Syntax

- An assembly statement
  - 3 essentials: opcode, operands (destination, source)
  - add a,b,c -> c = a + b

Intel Syntax	AT&T Syntax
opcode dest, src	opcode src, dest
No suffix for opcode	Has suffix for opcode
Immediate value is number	Immediate value has $
Plain register name	Register name has %
[ ] for memory operand	( ) for memory operand

- Examples
  - mov eax,5 — movl $5,%eax
  - mov eax,[ebx+4] — mov 4(%ebx),%eax

Road Map

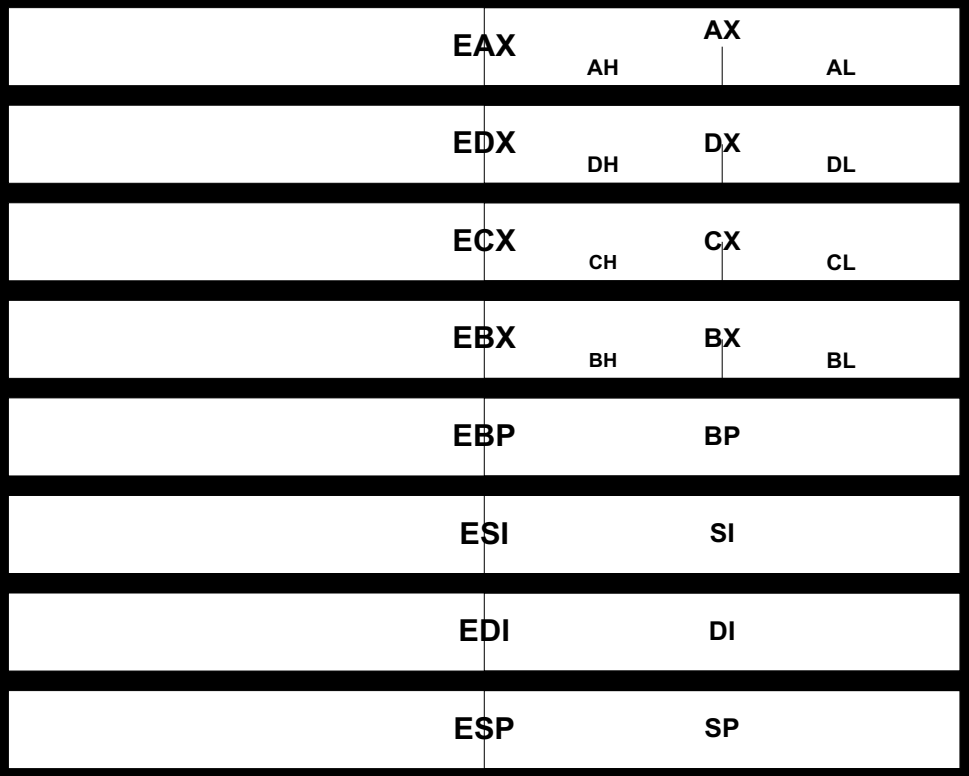
- IA-32 assembly language syntax
- IA-32 architecture – registers
- Assembly language addressing modes
- From Assembly language to instructions
- Examples
- Where to find more information
- Summary

x86 Registers

- Eight 32-bit general registers: EAX, EBX, ECX, EDX, ESI, EDI, ESP (stack pointer), EBP (base pointer, a.k.a. frame pointer)
- Names are not case-sensitive and are usually lower-case in assembly code (e.g. eax, ecx)
- More registers on the x86-64

x86 Registers

- 8 general-purpose 32-bit registers
- Athough general, they have common usages
  - ESP is the stack pointer
  - EBP is the frame pointer
- Not all registers can be used for all operations
  - Multiplication, division, shifting use specific registers



x86 Floating-point Registers

- Floating-point unit uses a stack, st0, st1 …, st7
- Each register is 80-bits wide (doesn't use IEEE FP standard)
- Example
  - fld qword[a]
  - fadd

SIGN	EXPONENT	SIGNIFICAND

Road Map

- IA-32 assembly language syntax
- IA-32 architecture – registers
- Assembly language addressing modes
- From Assembly language to instructions
- Examples
- Where to find more information
- Summary

X86 Operand Addressing Modes

- Addressing mode is about where the operands are.
- Immediate Operands – operand values are part of the instruction
  - movl $5, %eax
- Register Operands – operand values are in registers
  - add %ebx, %eax
- Memory Operands – operand values are in memory
  - movl $5,0x4[%ebp]

X86 Operand addressing modes

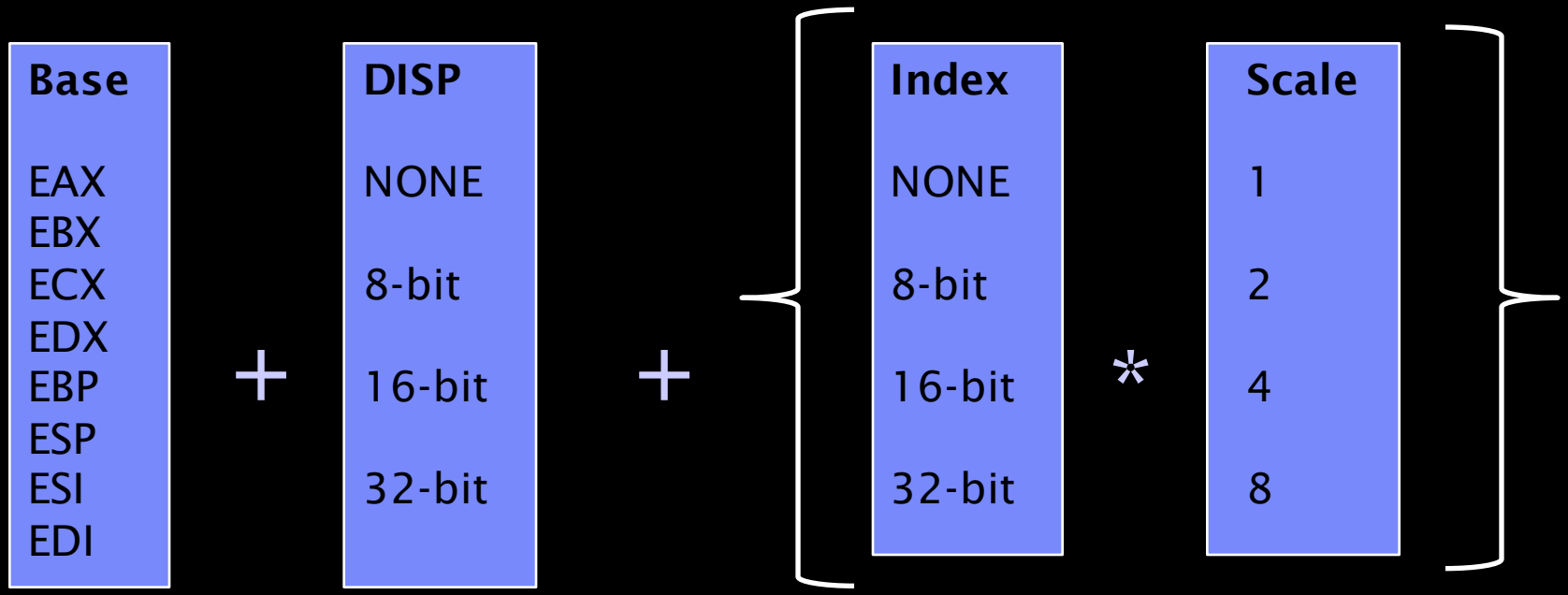
- Memory Operands – operand values are in memory
  - Addressed with

- Specifying the segment
  - Explicitly specify segments: movl %eax, %es:(%ebx)
  - Implicitly specify segments

Reference Type	Register Used	Segment Type	Default Use
Instructions	CS	Code Segment	Instruction fetch
Stack	SS	Stack Segment	Anything stack (push etc.)
Local Data	DS	Data Segment	All data ref (non stack)
Strings	ES	Extra Segment	Dest of string instructions

X86 Operand Addressing Modes

- Specifying effective address
  - Offset = Base + Disp + (Index * Scale)



X86 Addressing Modes

- Writing the memory operand
  - Intel syntax: segreg:[base+index*scale+disp]
  - AT&T syntax: %segreg:disp(base, index, scale)
- Example:
  - int a[2][10]; and move a[1][2] into EAX
  - Intel syntax:

```
mov ebx, a;
mov ecx, 2;
mov eax, ds:[ebx + ecx * 4h + 40]
```

  - AT&T syntax:

```
movl &a,%ebx;
movl $2,%ecx;
movl %ds:40(%ebx, %ecx, 0x4),%eax
```

Road Map

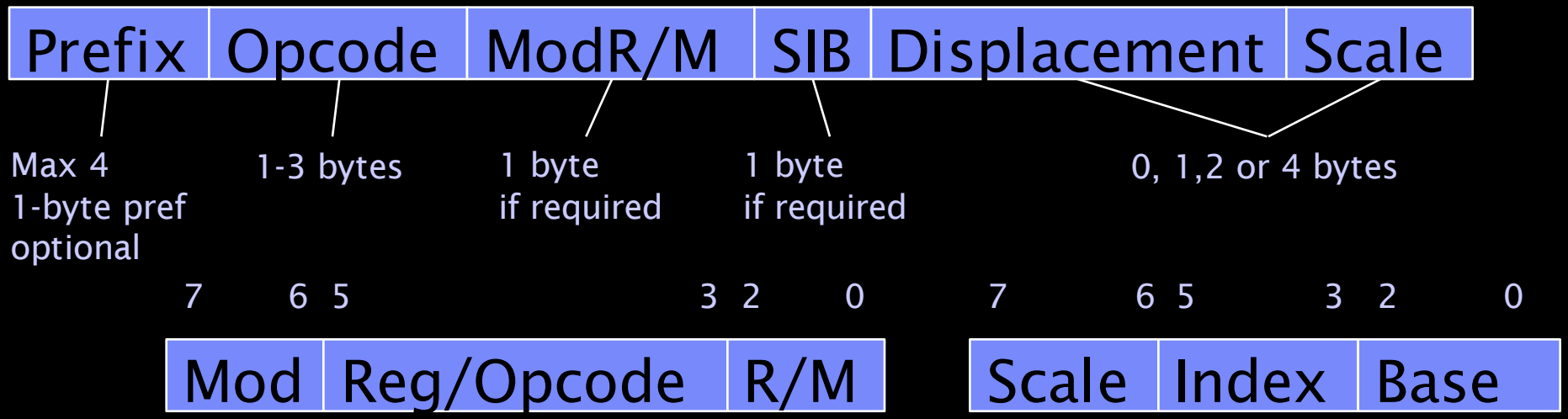
- IA-32 assembly language syntax
- IA-32 architecture – registers
- Assembly language addressing mode
- From Assembly language to instructions
- Examples
- Where to find more information
- Summary

From Assembly Language to Instruction

- Assembly language is for humans
- Instructions are for machines
- One assembly language statement translates into one instruction
- Why this translation is important
  - x86 is CISC, i.e., instructions have variable length
  - To properly deliver/inspect malicious code at the correct memory location, attackers/defenders should be aware of the length of the instructions

From Assembly Language to Instruction

- X86 Instruction Format



Prefix is for lock (atomic) instructions, segment, etc.
ModR/M and SIB specify addressing modes (register or memory)
Displacement and immediate value when necessary

From Assembly Language to Instruction

- The x86 instruction format
  - E.g. SUBL r32, r/m32 (ATT format);
  - Meaning: r/m32 = r/m32 –r32
  - r32 => 32-bit register operand
    - Corresponds to the Reg/Opcode field of ModR/M
  - r/m32 => 32-bit register or memory operand
    - Corresponds to the R/M filed of ModR/M
  - An x86 instruction has at most two operands (incl. dest)
  - An x86 instruction has at most one operand in memory

From Assembly Language to Instruction

- Examples (AT&T format):
  - SUBL r32, r/m32; r/m32 = r/m32 -r32
    - subl %eax, %ebx
    - 0x29 0xC3 (0b 11 000 011)
  - ADDL imm32, r/m32; r/m32 = r/m32 + imm32
    - addl $0xff000005, %ecx
    - 0x81 0xC1 (0b 11 000 001) 0x05 0x00 0x00 0xff
  - MULL r/m32; eax = eax * r/m32
    - mull %ss:0x40(%ebx, %ecx, 4)
  - 0x36 0xf7 0x64 (0b 01 100 100) 0x8b (0b 10 001 011) 0x40

Road Map

- IA-32 assembly language syntax
- IA-32 architecture – registers
- Assembly language addressing mode
- From Assembly language to instructions
- Examples
- Where to find more information
- Summary

Examine Real Executables

- Let's try the instructions in a real machine
- Just use gcc to compile
- Command:

  gcc -m32 -c -S array_example.c
- Let's try it

Road Map

- IA-32 assembly language syntax
- IA-32 architecture – registers
- Assembly language addressing mode
- From Assembly language to instructions
- Examples
- Where to find more information
- Summary

Reference Documents

- You won't be able to remember everything after a week
- Assembly language syntax
  - http://asm.sourceforge.net/articles/linasm.html
  - Full Ref: ORACLE, x86 Assembly Language Reference Manual
- A Tiny Guide to Programming in 32-bit x86 Assembly Language on syllabus

Reference Documents

- Intel Software Developer's manual is the best reference
  - IA-32 architecture (registers etc.): Vol. 1 Chapter 3
  - IA-32 instruction format: Vol. 2 Chapter 2
  - IA-32 instruction references: Vol. 2 Chapter 3 and 4
- Still confused? Try it out with gcc and objdump

Road Map

- IA-32 assembly language syntax
- IA-32 architecture – registers
- Assembly language addressing mode
- From Assembly language to instructions
- Examples
- Where to find more information
- Summary

Summary

- IA-32 assembly language syntax:
  - Instructions for human-to-read/write, for compiler to parse
  - Intel and AT&T
- IA-32 Architecture for instruction execution
  - CISC ISA
  - General-purpose registers
- IA-32 instruction addressing modes
  - Locating the operands
  - Immediate, register and memory (seg + offset: SIB & disp)
- IA-32 assembly language to instructions
  - Instruction format
  - ModR/M and SIB tables (know the existence of the tables)