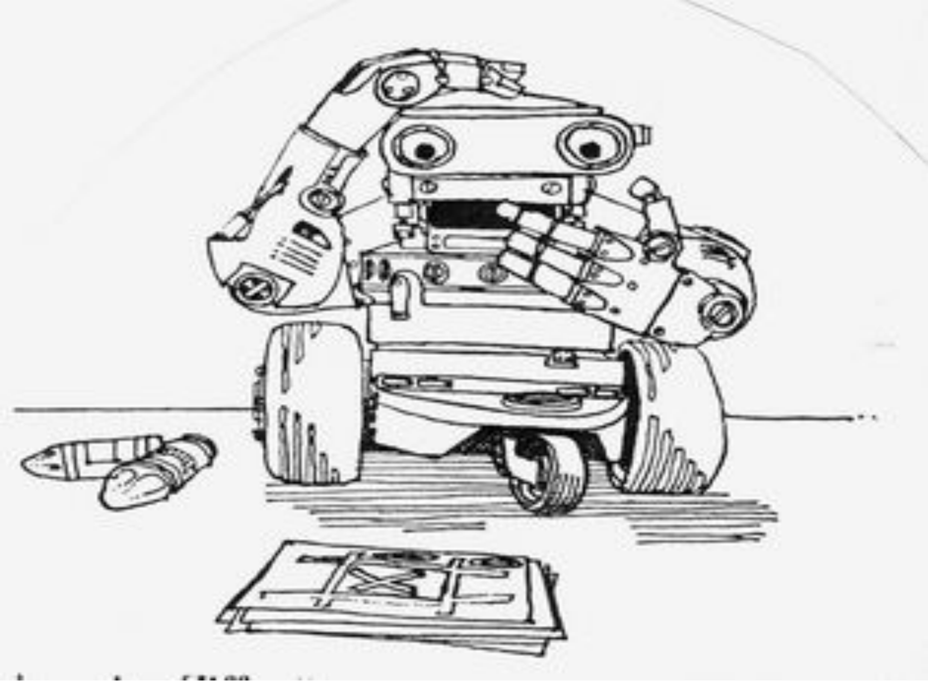
Arno Puder

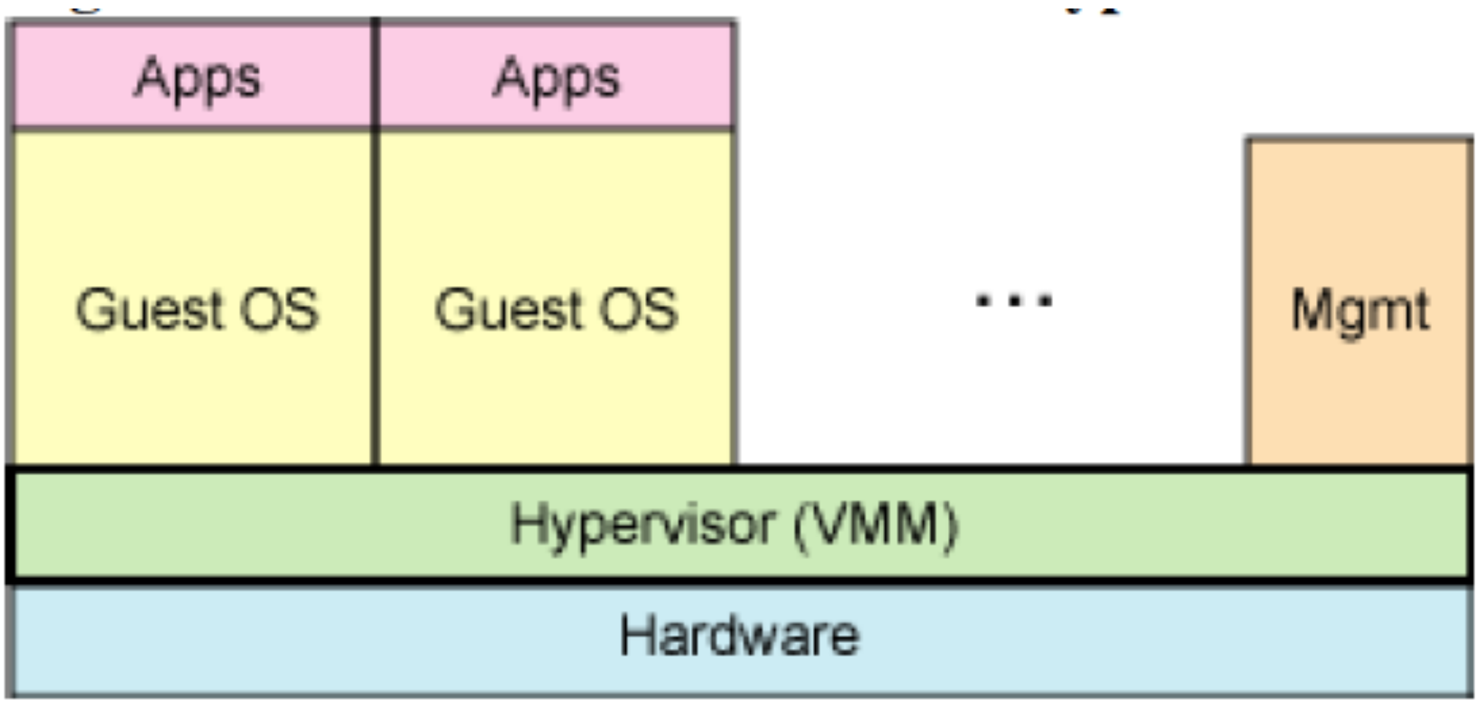


Virtualization

Overview

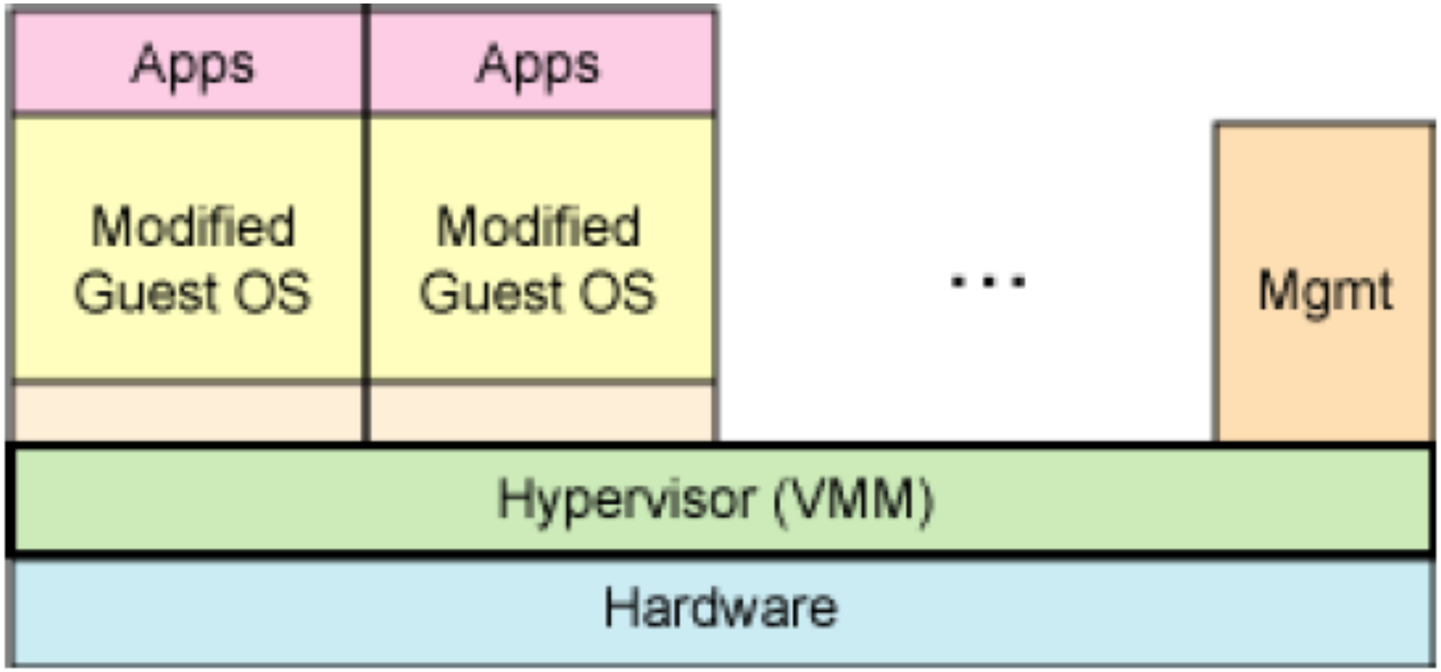
- Virtualization: take something of one form and make it appear to be another form.
- Virtualizing a computer means to make it appear to be a different computer.
- Virtualization dates back to early days of computation.
- Types of virtualization:
  - Full virtualization
  - Paravirtualization
  - Operating system-level virtualization

Full Virtualization



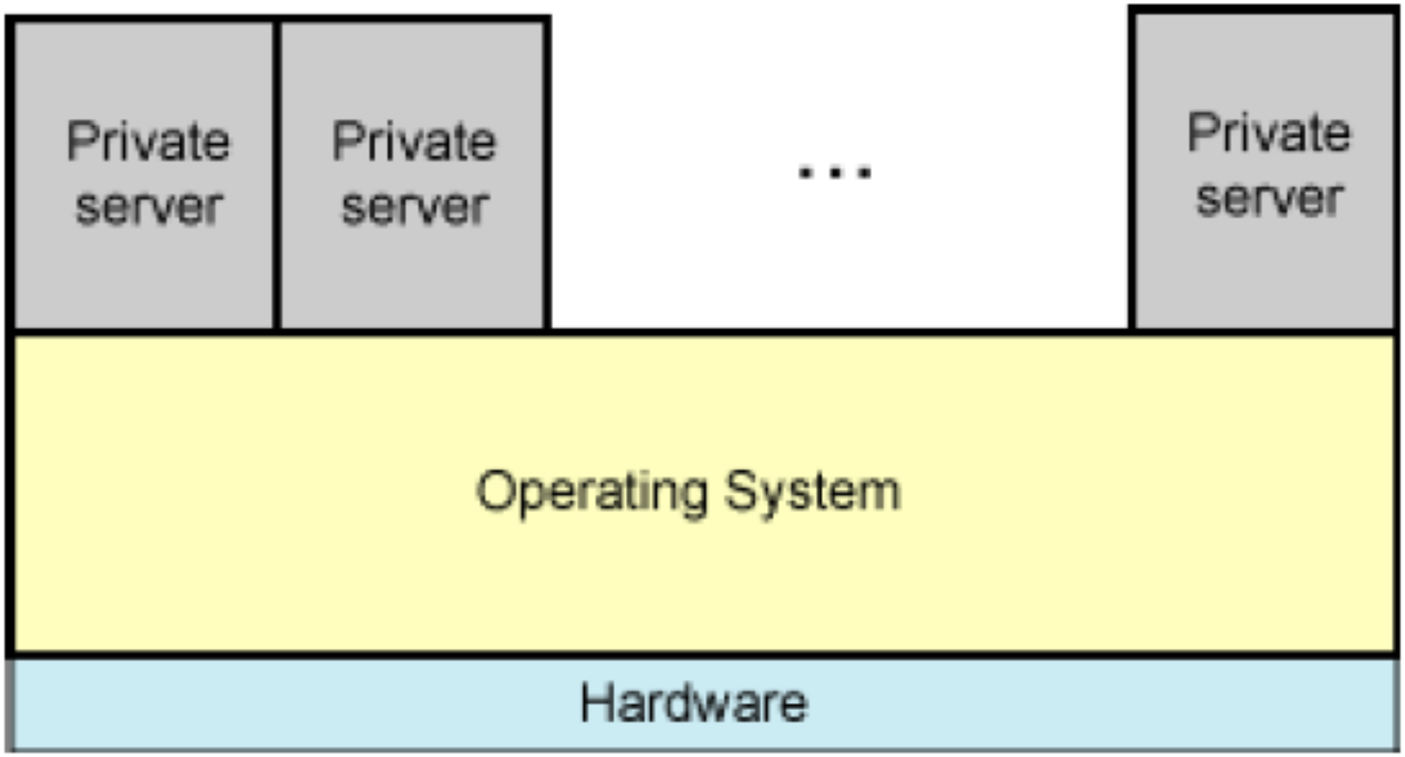
- A program called *hypervisor* emulates the complete hardware.
- Pro: can run unmodified OSes (called Guest OS).
- Cons: slow.
- Examples: Bochs, VirtualBox.

Paravirtualization



- Guest OS is modified to interact with host hardware more efficiently.
- Pro: higher performance.
- Cons: guest OS needs to be modified.
- Example: VirtualBox Guest Additions.

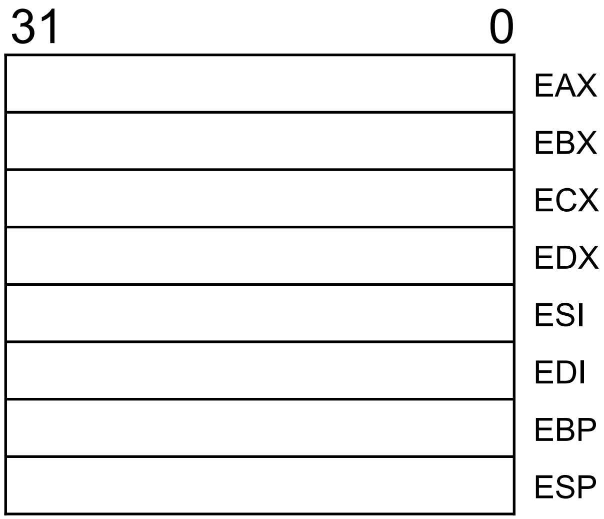
OS-Level Virtualization



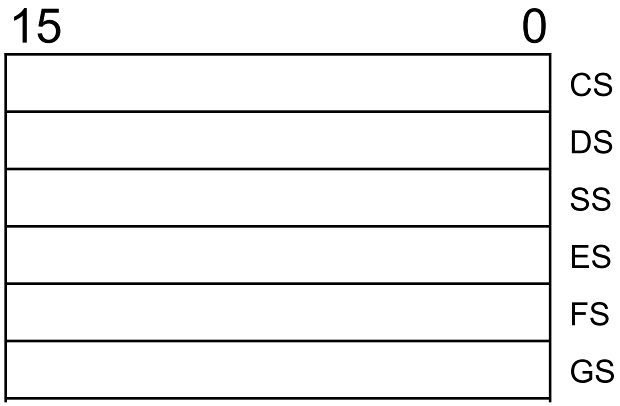
- Host OS offers special API to support virtualization (e.g., Linux namespaces). Private server instances are isolated from each other.
- Pros: very efficient execution. No need for Guest OS.
- Cons: not full virtualization. Private server need to be adapted.
- Example: Docker.

x86 Registers

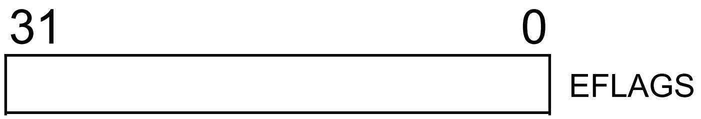
General-Purpose Registers



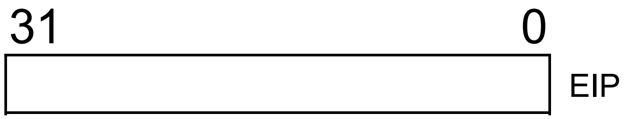
Segment Registers



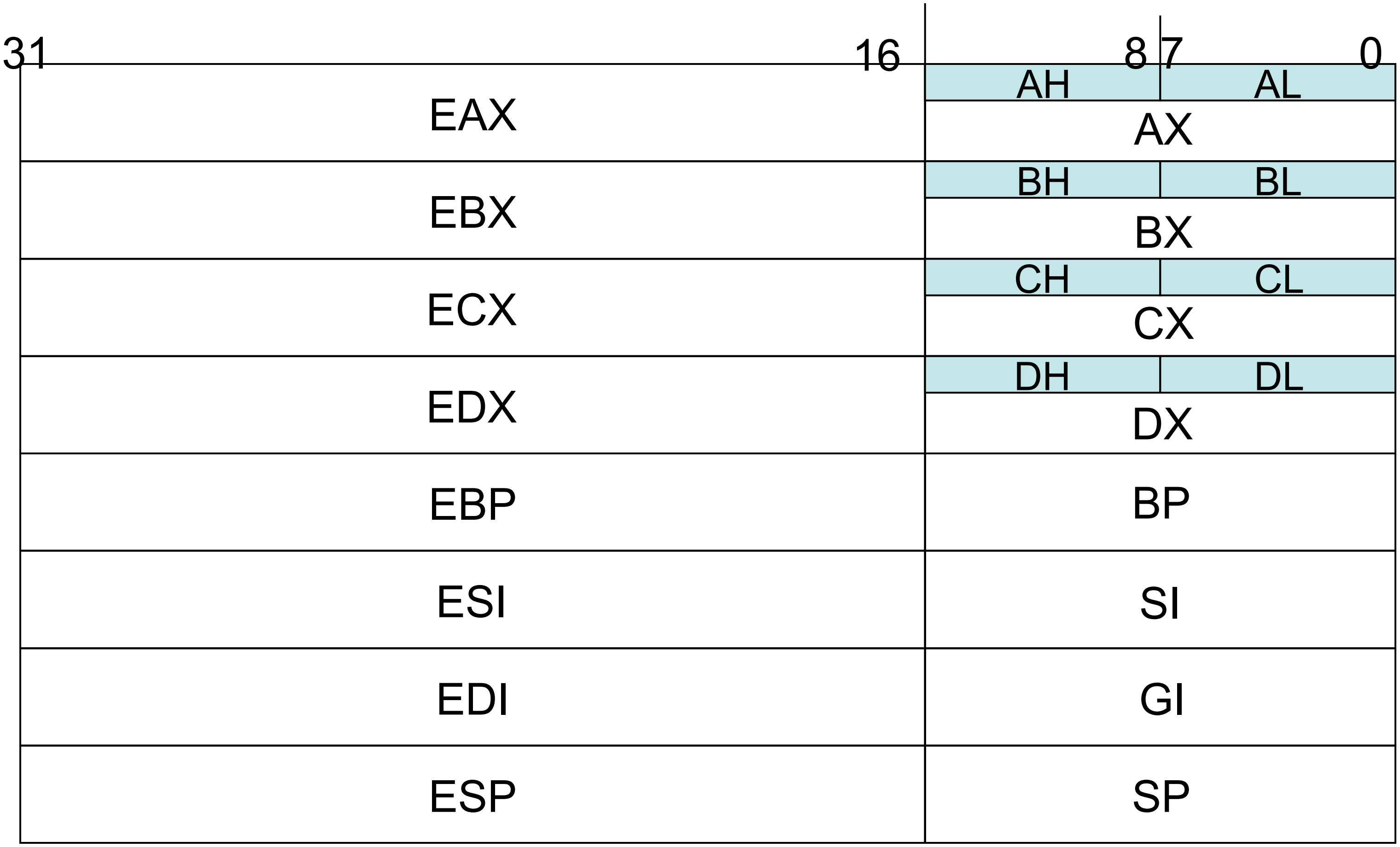
Program status and control Register



Instruction Register



General Purpose Registers



31 16	8	7 0
EAX	AH	AL
	AX	
EBX	BH	BL
	BX	
ECX	CH	CL
	CX	
EDX	DH	DL
	DX	
EBP	BP	
ESI	SI	
EDI	GI	
ESP	SP	

Some registers are only available for certain machine instructions.

X86 Instruction Overview

Memory Operations

MOV	-	move data
Push	-	push data onto stack
Pop	-	Pop data off the stack

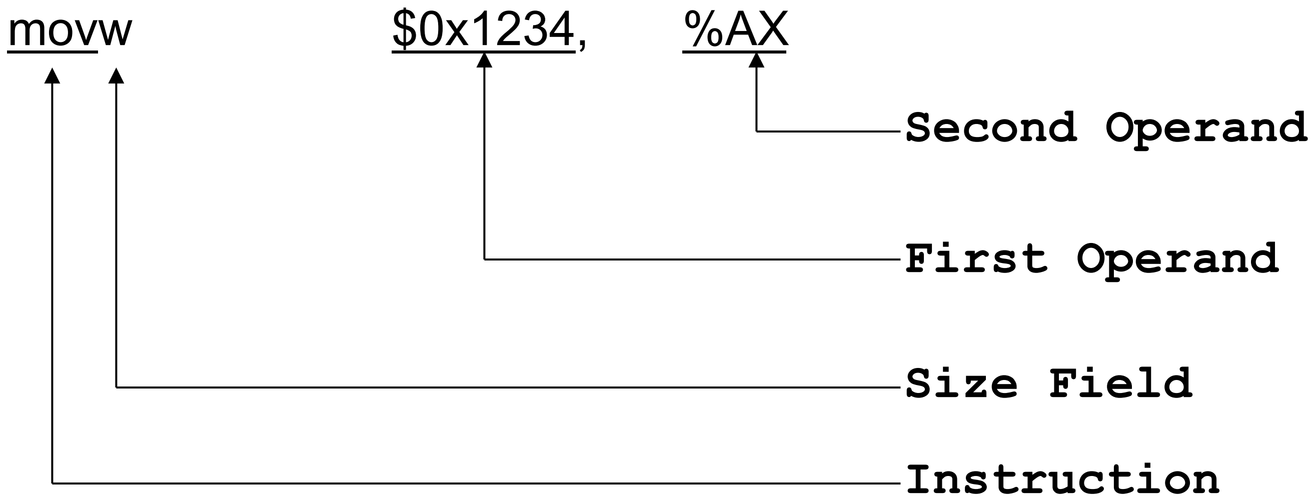
Logical and arithmetic operations

AND	-	Bitwise and
OR	-	Bitwise or
XOR	-	Bitwise exclusive or
ADD	-	Addition
SUB	-	Subtraction

Control flow operatoins

JMP	-	Jump
JZ	-	Jump if Zero
JNZ	-	Jump if NOT Zero
CALL	-	Call Subroutine
RET	-	Return from subroutine

Anatomy of a Move Instruction



movw $0x1234, %AX

Second Operand

First Operand

Size Field

Instruction

- This instruction will move the value 0x1234 into register %AX
- General format of move instruction: mov src, dest
- NOTE: We assume AT&T assembly syntax!

Jump Instructions

- Jump instruction changes %EIP to modify flow of control
  - Used to implement if statements and loops
- Target of a jump is the address of an instruction (like a C pointer-to-function)
- Assembler labels reference an address
- Instructions:
  - JMP (unconditional jump)
  - JZ (Jump if Zero)
  - JNZ (Jump if Not Zero)

Indirect Addressing

- Assembly equivalent of dereferencing a pointer.
- Writes to the memory location designated by the indirect address.
- General format:

  offset(register)

- Examples:

```
movb $’A’,(%ecx)
movw %bx,4(%esp)
```

Sample Program

```
// C program
void boot() {
    char* screen_base = (char *) 0xb8000;
    *screen_base = ‘A’;
    while (1) ;
}
```

```
Addr     Machine code     Assembly
------------------------------------------------
   0:     b8 00 80 0b 00             mov     $0xb8000,%eax
   5:     b3 41                      mov     $0x41,%bl
   7:     67 88 18                   mov     %bl,(%eax)
  10:     eb fe              loop:   jmp     loop
```

Class homepage shows a simple PC Emulator that can run this program.