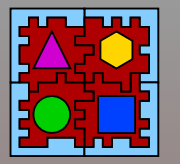


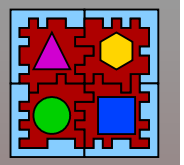
Dedicated Operating System

OS Initialization Review: From the Electron to The Boot

LISHA/UFSC

Prof. Dr. Antônio Augusto Fröhlich
guto@lisha.ufsc.br
http://www.lisha.ufsc.br/Guto

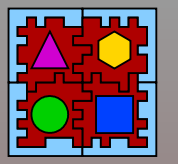
March 2011



What is an OS at all?

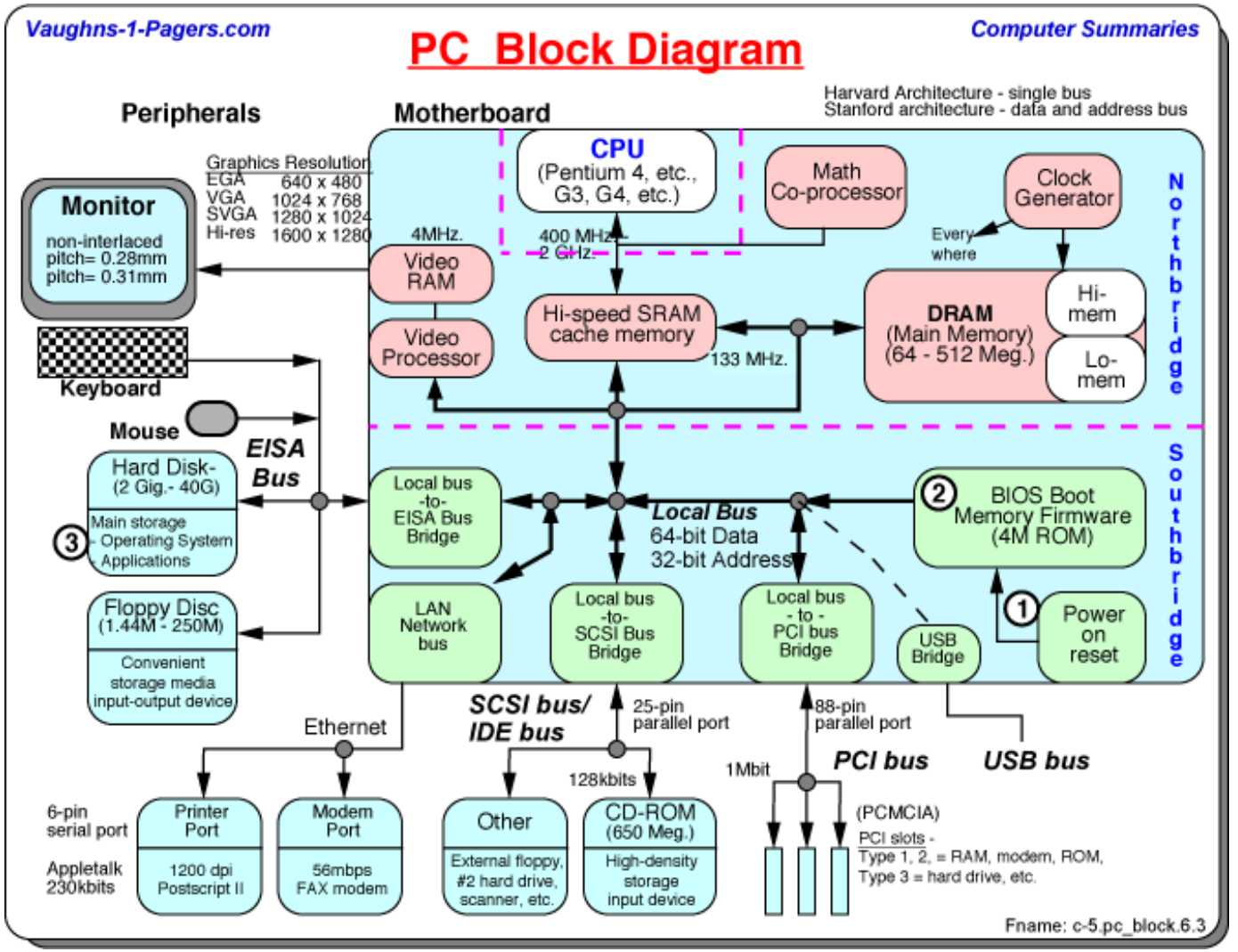
- Habermann: “Everything between hardware and applications”
- OS handles abstraction levels
  - Everything, from electrons to Java
- To become a OS developer one needs:
  - knowledge of computer architecture
  - notions of electronic circuits
  - outstanding programming skills

- How will you learn this?
  - Deep diving into a PC operating system

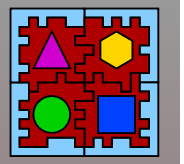


Dedicated Operating System

Why a PC?



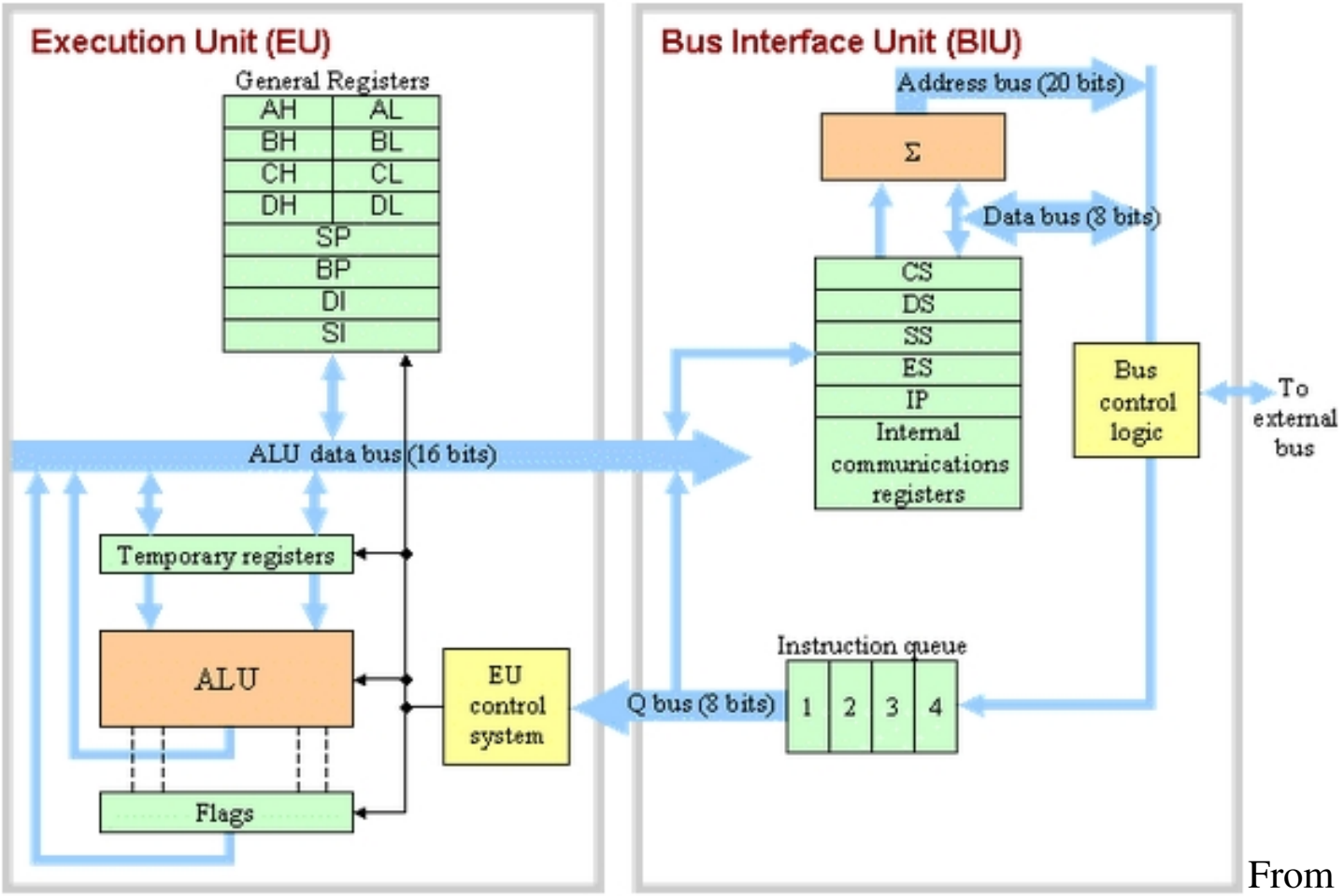
- OSs for PCs are very complex
  - Legacy
  - Lack of standards



Starting the Machine

- When the PC's 'Power' button is pushed:
  - Start the SMPS (switched-mode power supply)
    - Integrity depends on precise voltage regulation
    - SMPS stabilizes and then
  - CPU comes to a cold-reset
    - It needs to be initialized (booted)
    - Most common architecture today: Intel's x86
    - Why?
      - IBM: x86 and DOS to replace Motorola's 68k and CPM16

Intel 8088 Architecture



From CSedukit.com

-- Static registers (groups of D Flip-Flops) used to hold or transfer binary data

-- Logic gate circuits designed to perform arithmetic or logical functions

-- Logic gate circuits designed to provide internal control to processor

-- Internal data busses used to pass information between components

“Modern” x86 Architecture

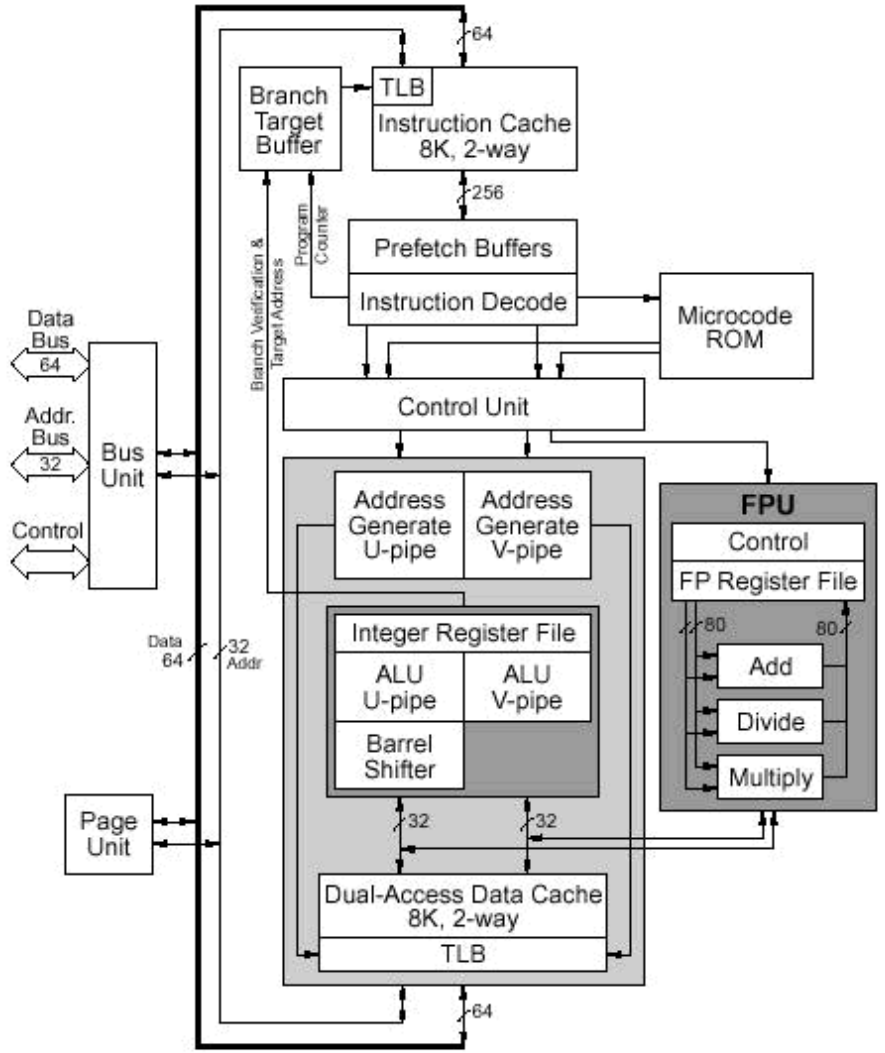
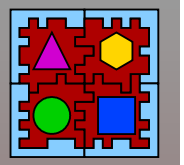


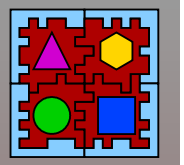
Figure 1. Pentium block diagram.

- Limitations
  - No register banks
  - No internal bus
    - cx => IO
    - bx => ULA
  - Backward compatibility
- Will live with that
  - x86 is super-scalar
  - Microcode
    - Communication to the internal architecture
    - Speculative execution
    - Indeterminism



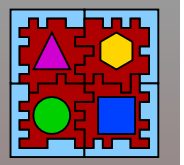
BIST – Built-In Self Test

- Microcode (firmware) ran at boot
  - Checks hardware integrity
  - Turns on stable units
  - e.g.:
    - division bug on the Pentium I processor
    - handling of failures in production process
- BIST generates report on hardware status
  - At the BIOS, 1048 bits
  - Things may be turned on and off by micro-fuses
- After the BIST
  - Processor ready to call first software instruction



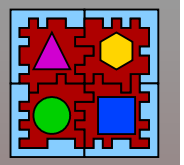
Booting “For Real”

- Intel's Basic Architecture Manual
  - Section 9.4.1 – First Instruction Executed
  - 0xFFFFFFF0
  - In x86 mode
  - It is a “jmp #BIOS_ADDR”
  - Why does it have 16-bytes instead of 32 bits?
  - Why at the “top” (4GB)?
- It allows different sizes of BIOS memories
- Flexibility for system developers



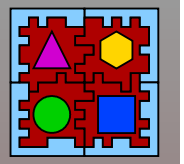
BIOS POST

- POST – Power-On Self Test
  - What comes first? BIOS or VGA?
    - Hooks for peripheral initialization
    - VGA comes first
  - Initializes legacy peripherals
    - Keyboard, serial, parallel ports, buses
    - South-bridge - ISA (timing legacy)
  - Initialize remaining things (new stuff)
  - Memory test - write-read-compare procedure
    - A few chips feature smart controllers (self-test)
    - For others run test until it fails (memory top reached)
  - POST report status to NVRAM (CMOS)
    - At internal RTC (!)
    - No standard report – useless for generic OS



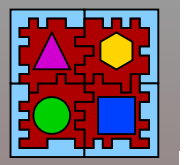
Initializing the machine

- After POST, BIOS initialization code
  - Run at 8086 mode (Real mode)
  - BIOS implements drivers in 16-bits code
    - Useless for modern OS
    - Drivers re-implemented with 32-bits code by OS
- Why are BIOS still in use?
  - Hardware bugs “workarounds” at BIOS
- Final initialization hooks
  - Auxiliary boot
    - e.g., network (remote boot)
    - USB is not here, BIOS emulates it as a disc
  - If no auxiliary boot
    - Load and fetch the first sector of the first boot
    - MBR – Master Boot Record, 512 bytes at 0x7c00
    - Here is the bootstrap

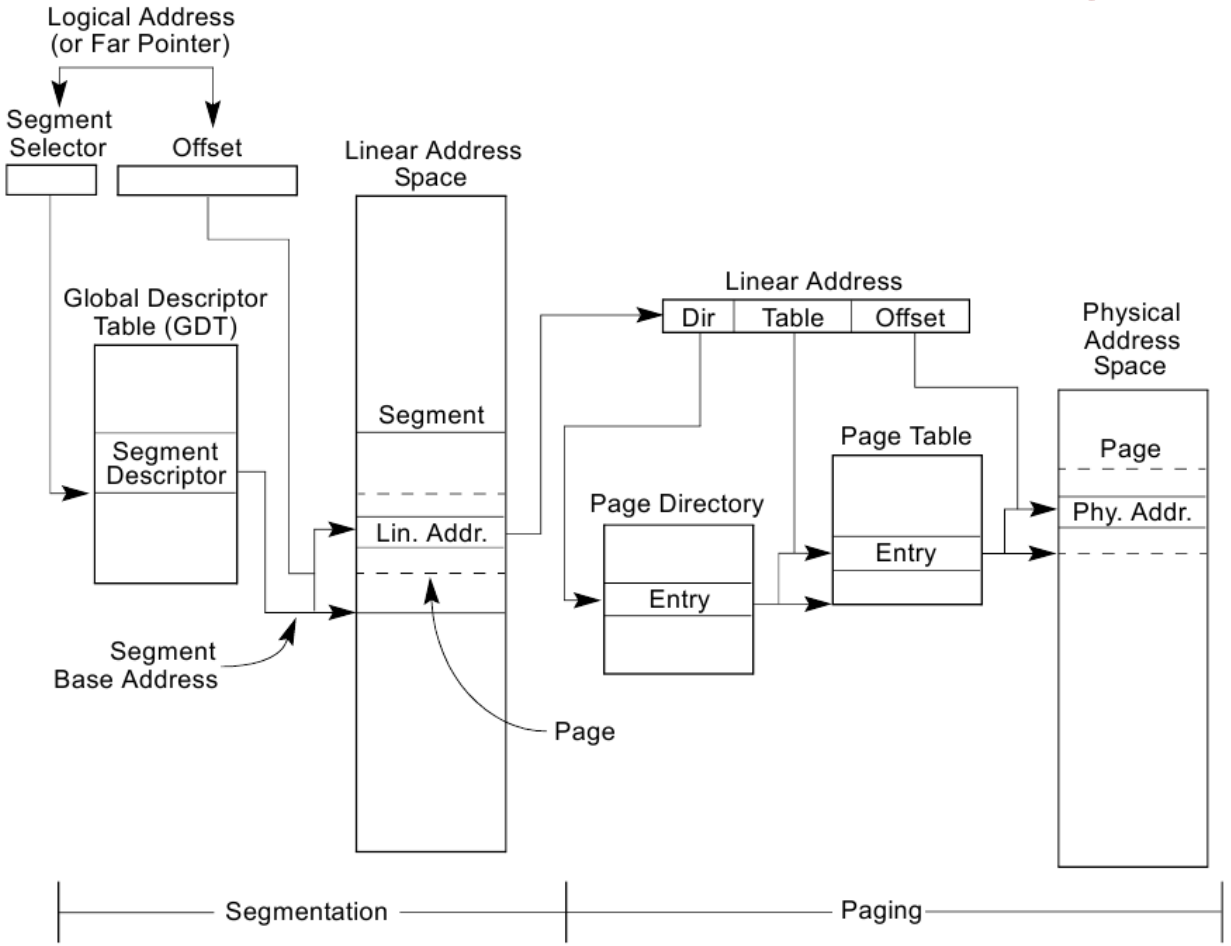


x86 Bootstrap

- What is that?
  - It is all that need to be done before forgetting about the BIOS
  - Read bootloader from disc to RAM
  - Enter Protected mode (32 or 64 bits)
  - Call bootloader
  - Needs special tools to be assembled (as86)
  - From this point on system is functional
    - May execute “generic”, 32 bits, compiled code
  - Although lots of architecture-specific configuration still needs to be performed...



x86 Architecture Legacy Overview



More at next class

- MMU (Memory Management Unit)
  - Paging X Segmentation
  - Internal X external fragmentation
  - Unfinished 8086 => (CS << 4) + offset