Realizzare un emulatore di videogiochi

Lorenzo Mancini

EuroPython 2011 – Florence

lmancini@develer.com

Our plan

- Give some emulation background
- Introduce the canonic 80's video game model
- Write an emulator from scratch

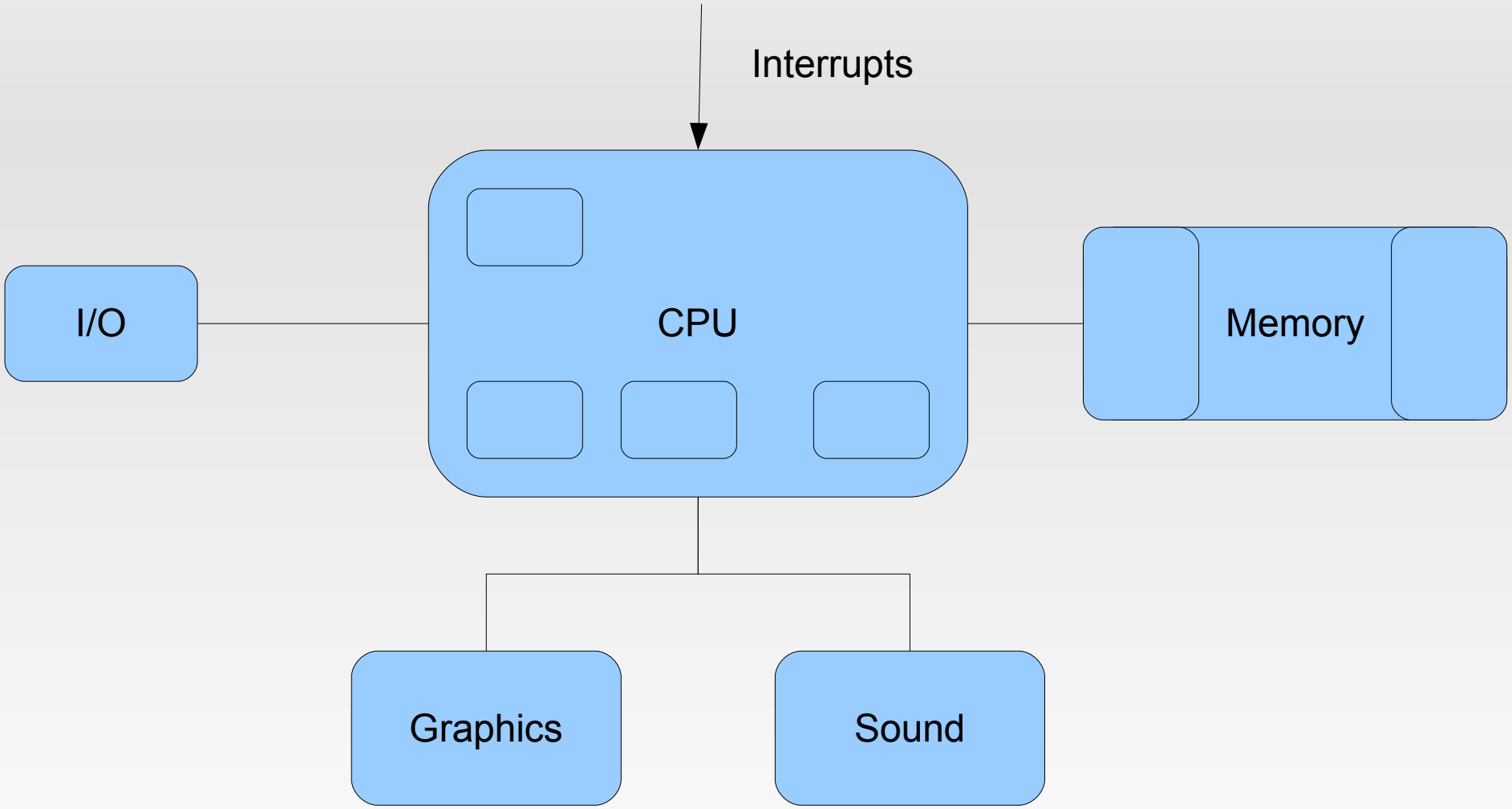
Emulation 101

- What is an emulator?
  - Software/hardware imitating a system
  - Accuracy/speed compromise
  - “Perfect” emulation – possible?
- What are emulators useful for?
  - Preservation of existing software
  - Existing software restyling
  - Compatibility layer

Practical applications

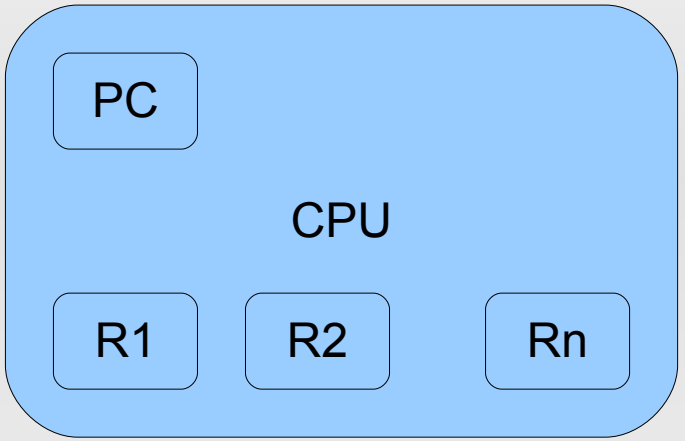
- Nintendo's Virtual Console ($66M 2010)
- Printers (HP LaserJet)
  - As compatibility layer
- VMWare's VMTools ($100M 2010)

The overall picture



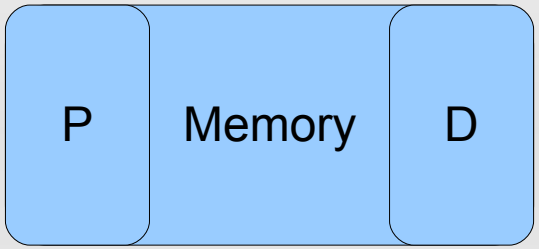
CPU

- Defines an instruction set
  - Can execute programs
- The CPU has a state
  - Program counter
  - Registers
- Opcodes and cycles
- Often an off-the-shelf component



Memory

- Heterogeneous storage
  - Instructions (program)
  - Data
- Addressable
- R/W access from CPU



Emulation approaches

- Interpreted
  - Read and execute instructions one by one
  - The CPU is simulated
- Dynamic recompilation
  - On-the-fly translation for target
  - Uses target's CPU (less overhead)

- Let's start with the interpreted approach

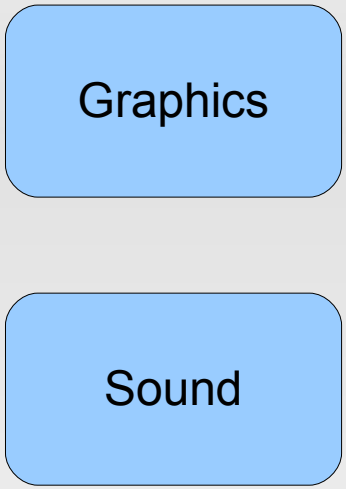
The emulator core

- Fetch-decode-execute loop

```
for _ in range(cycles_to_execute):
    # Fetch
    opcode = memory[PC]

    # Decode
    instruction = decode(opcode)

    # Execute
    execute(instruction)

    PC += 1
```

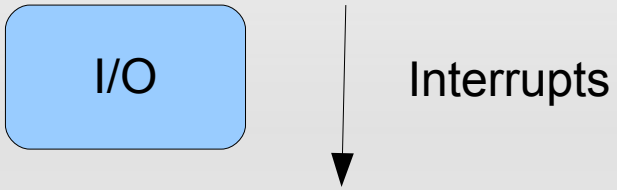
Graphics / Sound

- Graphic subsystem
  - Can be as simple as a video buffer
  - Can be as complicated as a GPU
- Sound subsystem
- They offload the CPU
- Often they're custom components



I/O, interrupts, timers

- Three related concepts
- I/O and the world outside
  - How does one check for input data?
- Interrupts vs. polling
- Timers



A complete emulator loop

```
while running:

    # Fetch-decode-execute
    executeCPU(n_cycles)

    # Do we need to generate interrupts?
    generateInterrupts()

    # Update machine status
    updateVideo()
    updateSound()
    updateTimers()

    # synchronize according to n_cycles
    sync()
```

Introducing the Chip-8

- A video game VM for hobbyists machines
- In a sense, the original Chip-8 was itself an emulator!

Chip-8 tech sheet

- CPU
  - 16 data registers (8 bit), 1 address register (16 bit)
- Memory
  - 4kb RAM (minus 200 bytes actually)
- Graphics
  - 64x32 pixels screen, monochrome, sprite-based
- Input: hex keyboard
- Timers: delay and sound
- No interrupts!

Our target

- Emulate the Chip-8, using Brix rom as testbed