Software: Running Programs

Questions for today

- What does "run" mean?
- How does code run on the CPU?

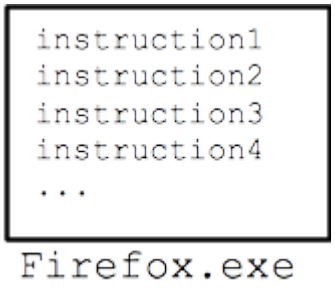
CPU - Machine Code

- The CPU implements "machine code" instructions
- Each machine code instruction is extremely simple
  -e.g. add 2 numbers
  -e.g. compare 2 numbers
- Javascript code we've used: `print(1, 2)`
- Javascript is not machine code
- Javascript does not run on the CPU directly
- Javascript works **in translation**
  - `print(1, 2)` translates to a lot of machine instructions to actually run on the CPU
- Only machine code runs on the CPU

"Software" is the general category of code which runs on the hardware. If the hardware is a player piano, then the software is the music. The common case is a "program" like Firefox -- software you run on your computer to solve a particular problem. A computer can run multiple programs at the same time and is responsible for keeping their memory separate.

A CPU understands a low level "machine code" language (also known as "native code"). The language of the machine code is hardwired into the design of the CPU hardware; it is not something that can be changed at will. Each family of compatible CPUs (e.g. the very popular Intel x86 family) has its own, idiosyncratic machine code which is not compatible with the machine code of other CPU families.

What is a Program/App?



- What is a Program or App?
  - e.g. Firefox
- Firefox is made of millions of machine code instructions
  - Run top to bottom (just like Javascript!)
  - The Firefox window appears (the first 1000 instructions)
  - Its menus appear (the next 1000)
  - The cursor blinks waiting for you to type
- The instructions such that, when run, "Firefox" actions happen
- Firefox.exe is a file in the file system, 80 MB in size
  (".exe" is a Windows convention for the name of a program file)
- The Firefox.exe file is mostly the bytes of machine code instructions
- Each instruction is, say, 4 bytes in size
- Firefox.exe at 80 MB is about 20 million machine instructions

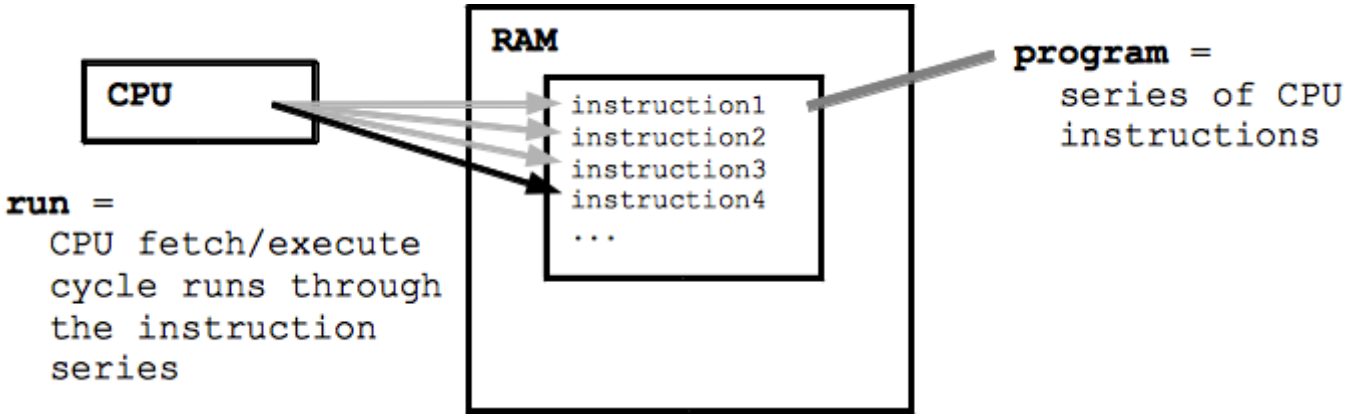
The machine code defines a set of individual instructions. Each machine code instruction is extremely primitive, such as adding two numbers or testing if a number is equal to zero. When stored, each instruction takes up just a few bytes. When we said earlier that a CPU can execute 2 billion operations per second, we meant that the CPU can execute 2 billion lines of machine code per second.

A program, such as Firefox, is made up of a sequence of millions of these very simple machine code instructions. It's a little hard to believe that something as rich and complicated as Firefox can be built up out of instructions that just add or compare two numbers, but that is how it works. A sand sculpture can be rich and complicated when viewed from a distance, even though the individual grains of sand are extremely simple.

How Does a Program Run?

- CPU runs a "fetch/execute cycle"
  -fetch one instruction in sequence
  -execute (run) that instruction, e.g. do the addition
  -fetch the next instruction, and so on
- Run a program = Start CPU running on its 1st instruction
  it runs down through all of the machine code, running the program
  the program will have instructions like "return to step 3" to keep it running
- Super simple machine code instructions run at the rate of 2 billion per-second

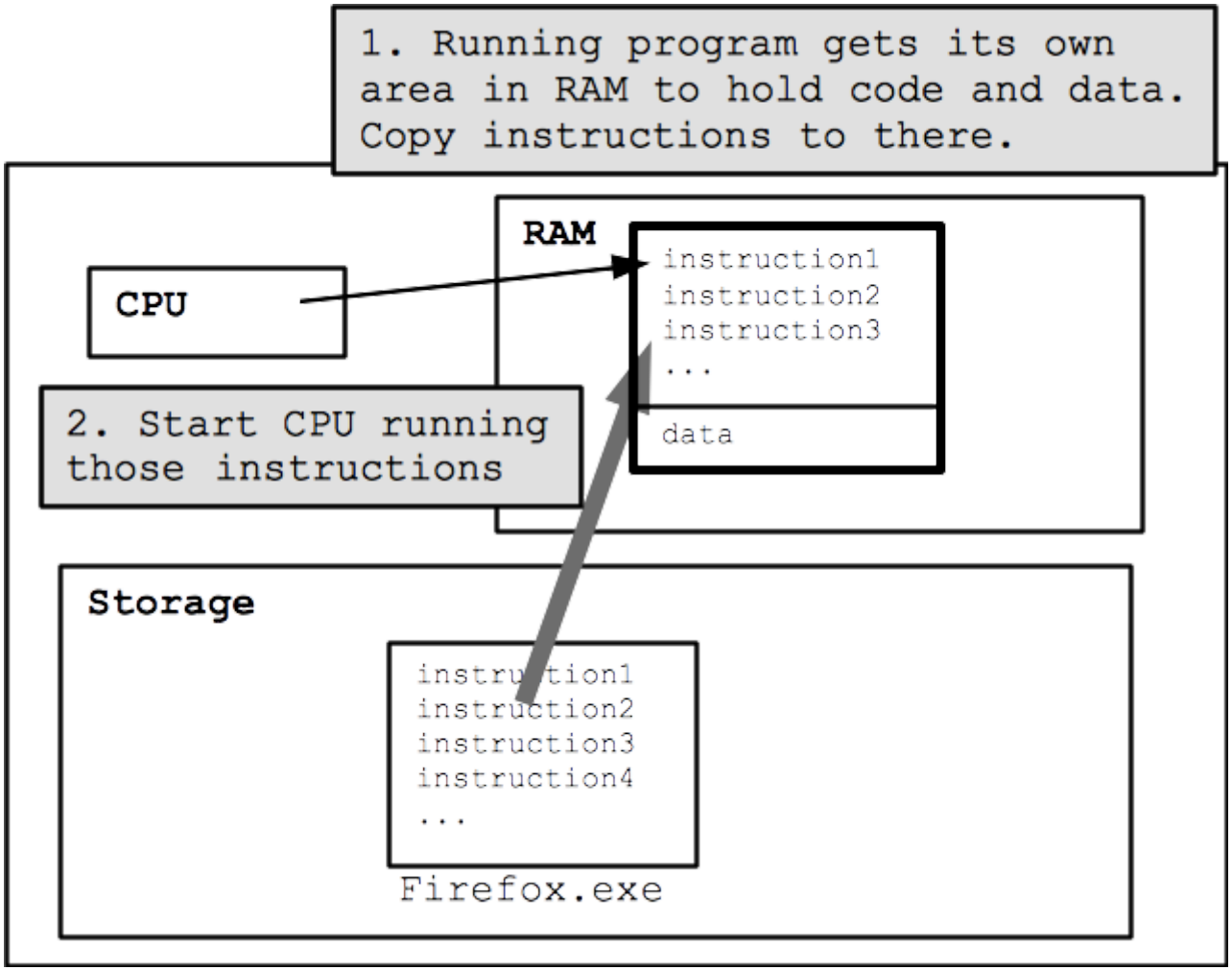
The CPU runs instructions using a "fetch-execute" cycle: the CPU gets the first instruction in the sequence, executes it (adding two numbers or whatever), then fetches the next instruction and executes it, and so on. Some of the instructions affect the order that the CPU takes through the instruction sequence. For example, an instruction might direct the CPU to jump back to an earlier point in the instruction sequence (loops are implemented this way), or to skip over the next instruction if a particular condition is true (if-statements are implemented this way).



How Does a Program Start?

- The file Firefox.exe contains its instructions (in the file system)
- To start Firefox.exe running:
  -Each program gets its own area of RAM

-The RAM area holds the program's code and data it manipulates
-The instruction bytes are copied from storage to RAM
-The CPU is directed to start running at the first instruction
-Now the program is running!



In the file system, a file like Firefox.exe just contains the bytes of the machine code instructions that make up the program (".exe" is a windows convention to mark a file as a program). Each machine code instruction takes up about 4 bytes, and whole program is just an enormous sequence of instructions.

When the user double clicks a program file to run it, essentially the block of bytes of the instructions for the program are copied into RAM, and then the CPU is directed to begin running at the first instruction in that area of RAM.

What Starts Firefox Running? The "Operating System"

- What are the steps to run Firefox?
- "Operating System"
  -e.g. Windows, Linux, Android, iOS
- Operating System = Management
- Set of supervisory programs that manage the computer
- The operating system runs when the computer first starts up
- Manage the start/stop of programs
- Manage RAM
- Manage persistent storage
- Computers can run multiple programs at the same time
- Operating system keeps track of the information for each program and shares resources (like RAM) among the programs

The "operating system" of a computer is like a first, supervisory program that begins running when the computer first starts up ("boots up"). The operating system plays an invisible administrative and bookkeeping role behind the scenes. When a laptop or phone starts up, the operating system typically gets things organized and then launches a "file explorer" program which displays available programs and menus etc. that show the user what is available, allowing the user to navigate and run programs.

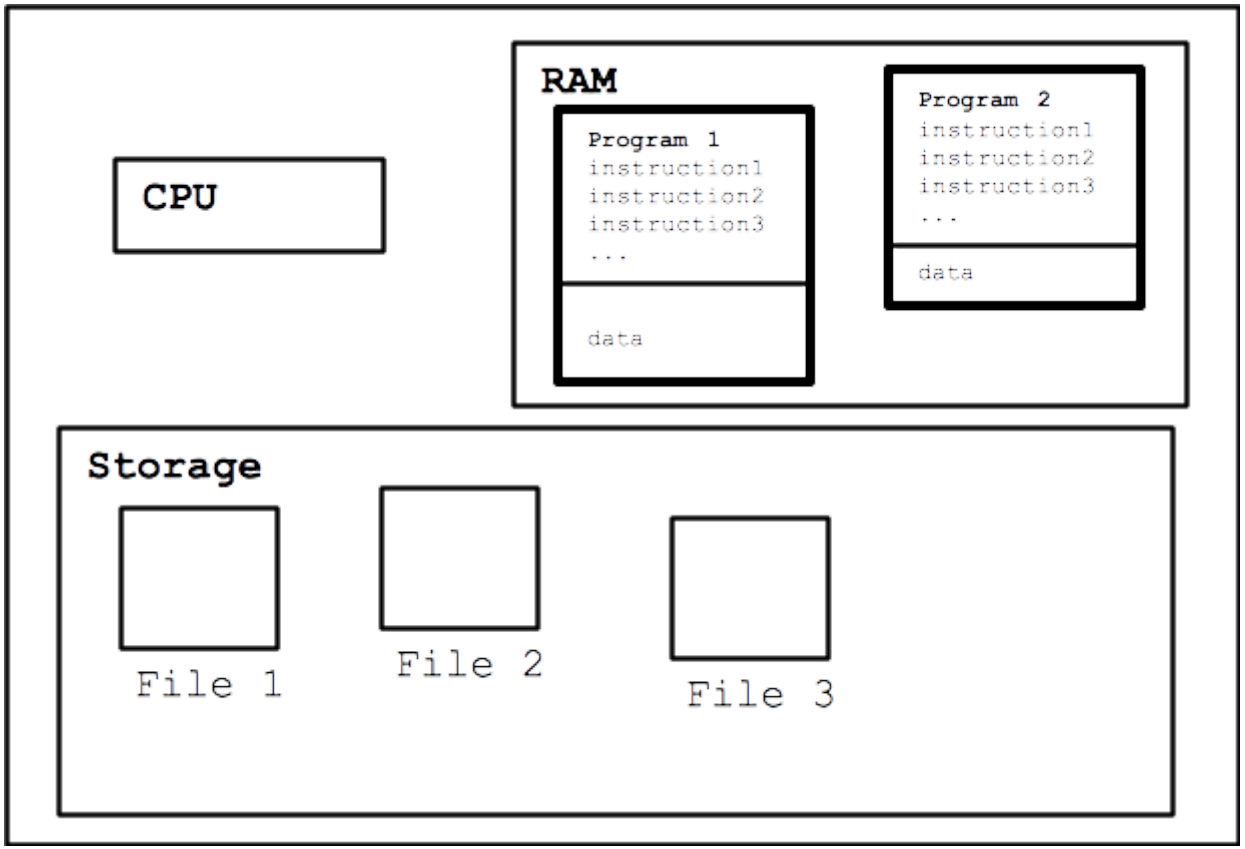
The operating system keeps things organized in the background so that multiple programs can run at the same time, which is known as "multitasking". The operating system gives each program its own area of memory, so each program only accesses its own resources .. attempting to limit what an erroneous or malicious program can do. Keeping the programs separate is sometimes known as "sandboxing" .. mediating the access of each program so it operates independently, without interfering with other programs or the system as a whole. Similarly, each program has some access to the screen through a window, but this output area is separated from the output of other programs.

Recall that a .exe file or whatever is essentially just a file of machine code instructions. When you double-click the program, the operating system "launches" the program, doing the housekeeping steps of allocating an area of memory within RAM for the program, loading the first section of the program's machine code into that memory, and finally directing the CPU to start running that code.

The Whole Picture - Scenarios

Now we have the whole picture of a program running on the hardware. Look at common scenarios.

- Demo: bring up "Activity Monitor" (Mac) "Task Manager" (windows)
  -See all the programs running
  -Do something costly in Firefox, see its CPU % spike
  -Kill a program



1. Normal Running Programs

- Operating system starts and stops programs
- Each program has its own separate area in RAM: its instructions + data
- CPU "round robin:" CPU runs a few instructions from each program
  so they all the programs appear to be running simultaneously

- Persistent storage is organized as a file system, programs can read and write data here

2. Program Exits Normally

- The program exits normally ("Quit" menu item)
- The operating system stops running that program
- The operating system reclaims the program's area of RAM (to be re-used)

3. Program Stuck/Infinite Loop - Abnormal Exit

- Suppose program gets stuck running an infinite loop, is "stuck"
- The operating system stops running that program - involuntary vs. normal-exit
- The operating system reclaims the program's area of RAM

4. Running Out Of Memory

- A program requests more RAM from the operating system
  e.g. to hold an image, but there's not enough RAM available
- The operating system refuses the request, the program gives an error message

5. Memory Access Error

- A program tries to access the memory of another program
- Maybe because of a bug (common)
- Maybe on purpose because it is malware
- The operating system blocks the access (ideally)
- Maybe kills the offending program too

6. Reboot

- Why does this fix anything????
- Sometimes some operating-system managed bytes in RAM is not quite right
  -a bug in the operating system, or perhaps a hardware error
- A reboot wipes all the data from RAM
- Starts up the operating system fresh
- This might fix the problem
- This should never be necessary
- But sometimes it works!
- The need for reboots is a hallmark of the presence of computers in previously reliable systems!

Why is it called Reboot?

- Chicken and egg problem.. who runs the operating system?
- Old phrase: "get over a fence by pulling on your own bootstraps"
- When first powered on, computer runs a special "bootstrap" program
- That program typically looks for persistent storage containing an operating system to run
- Boot up - start
- Reboot - do a fresh shutdown/startup cycle