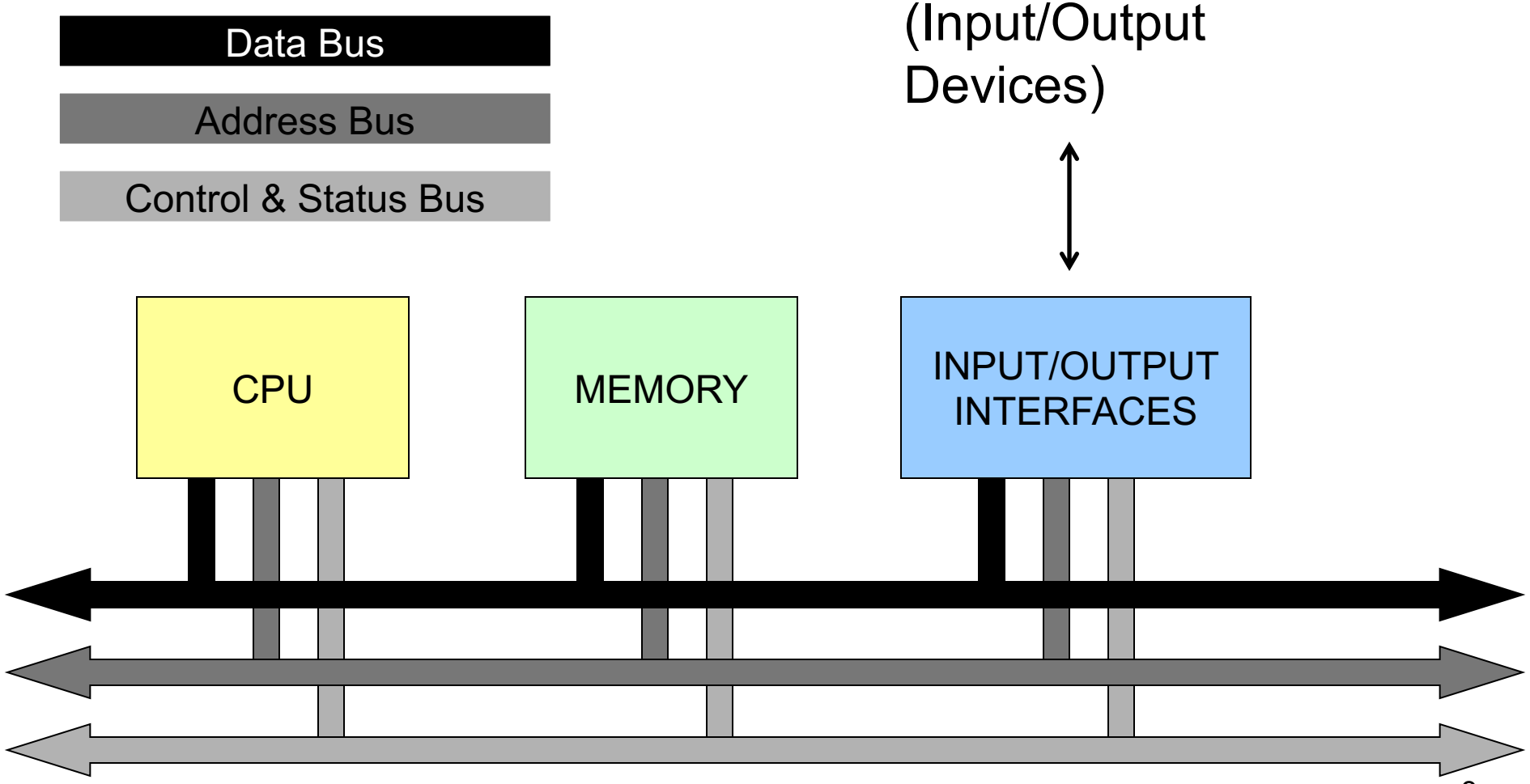
COMP0068: Computer Architecture & Operating Systems
Lecture 11: Computer Start-Up

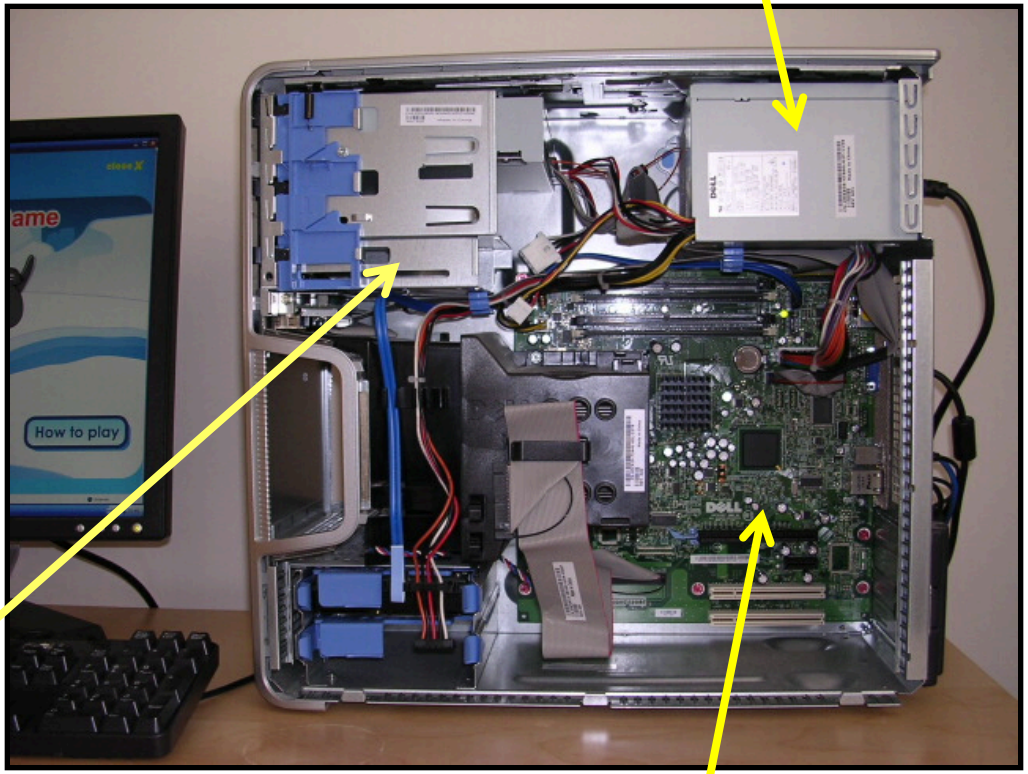
Lecture Overview

- We now have a fairly detailed understanding of how the Instruction Set Architecture (ISA) of a MIPS32 processor works.
- In this lecture I want to give an overview of the sequence of operations the computer executes at start-up and the main components involved in this phase.
- This will lead us into subsequent lectures about the upper layer managing the different components and programs of a computer; namely the operating system.

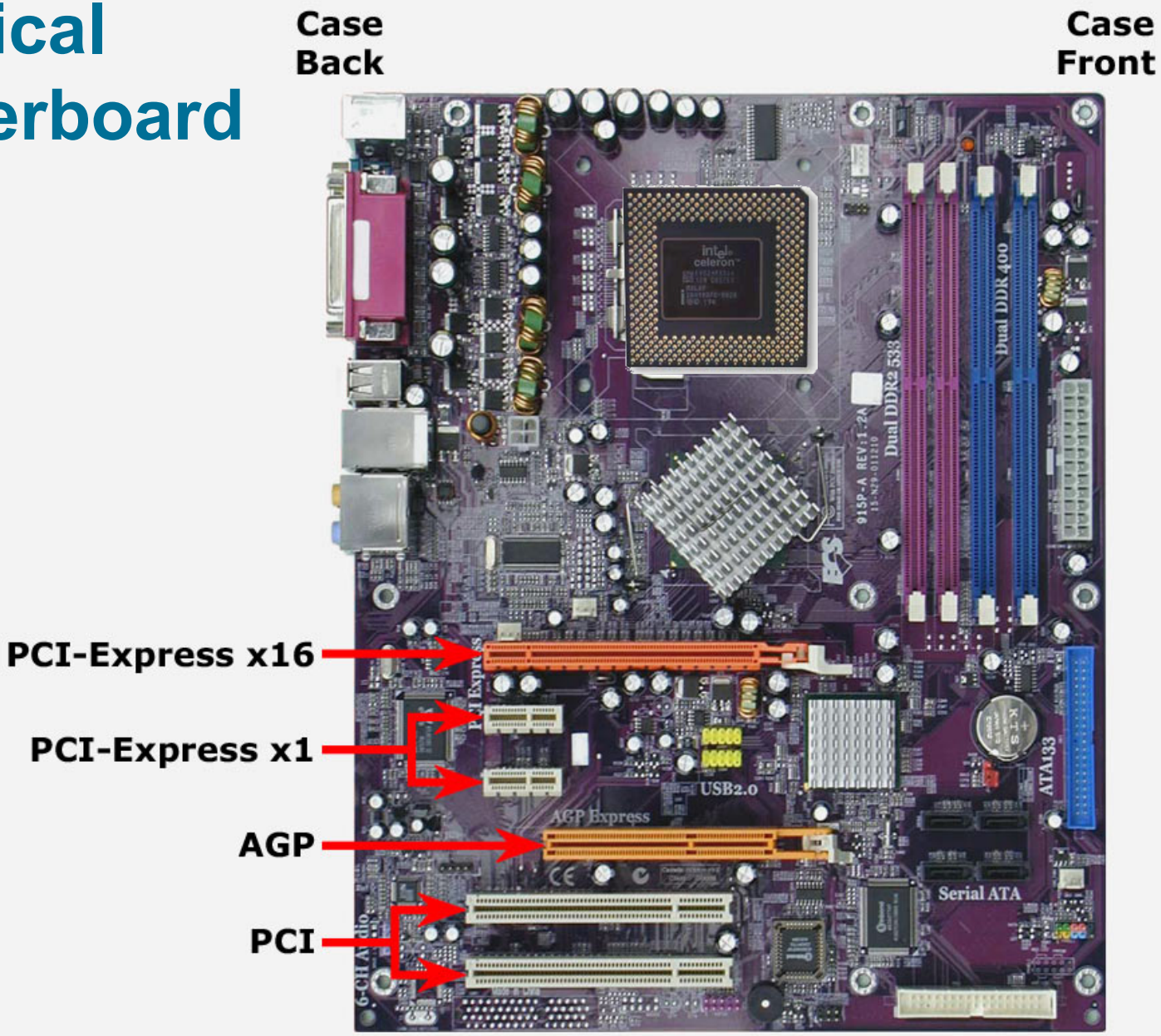
A Processors' (or Programmers') Abstract View of a Computer



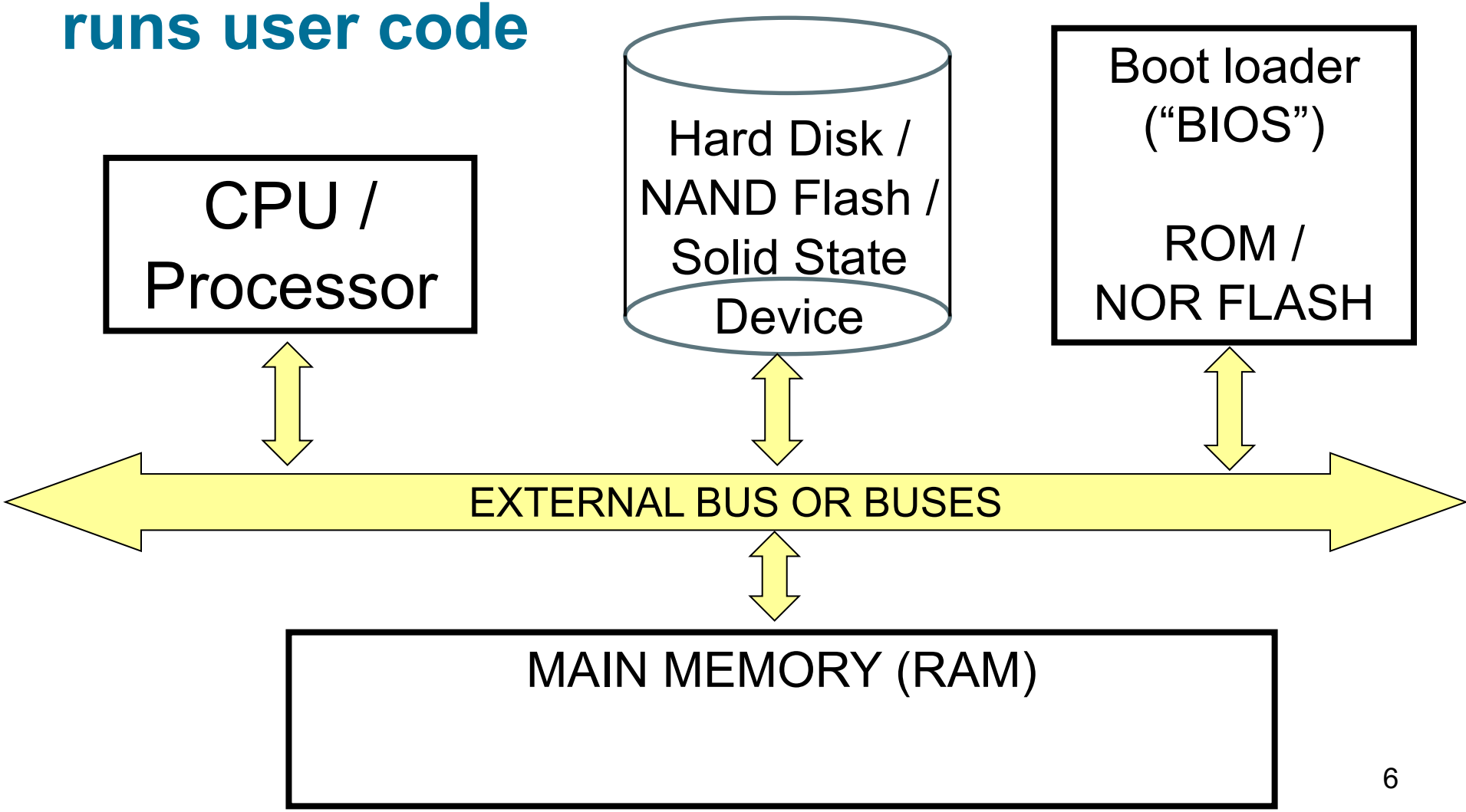
How does this relate to real hardware?



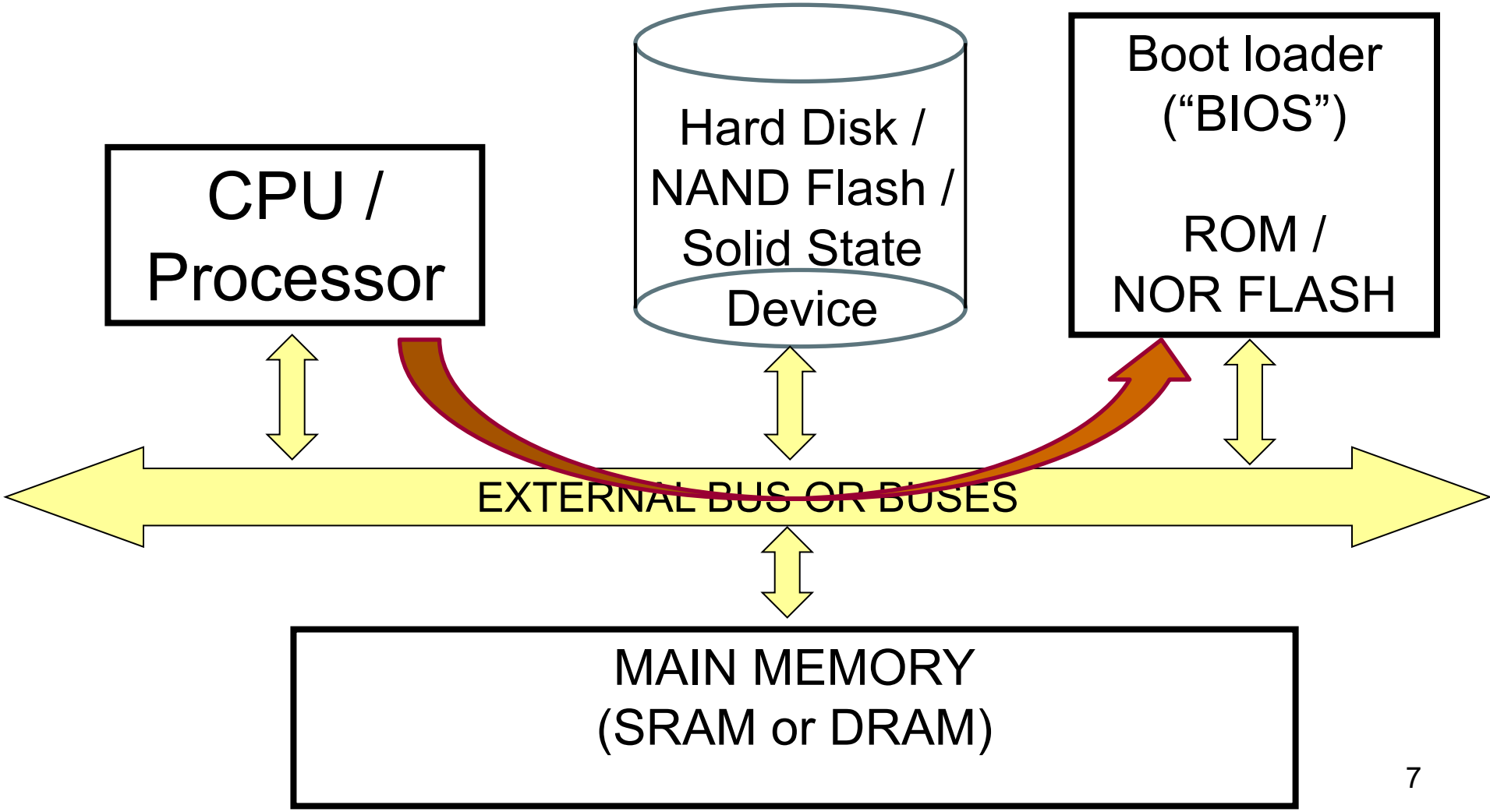
A typical Motherboard



Typically these components interact to "boot up" an operating system that then runs user code



1. On boot ... processor runs boot loader instructions from ROM or NOR Flash



What happens after turning on a computer?

Suwan Waenlor/Shutterstock

Essential steps

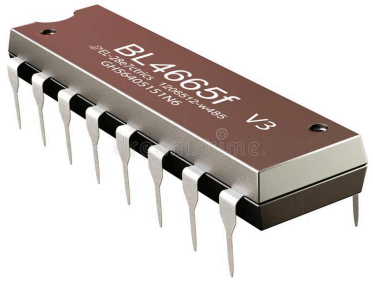
1. BIOS(Basic Input/Output Service) starts
2. It loads settings from CMOS (boot order, date/time, etc)
3. Initialises hardware devices
4. Run POST (Power-On Self-Test)
5. Looks for a bootable device with OS (Operating System)

   on it (e.g. the hard disk, flash drive, etc)
6. It hands over control to OS.

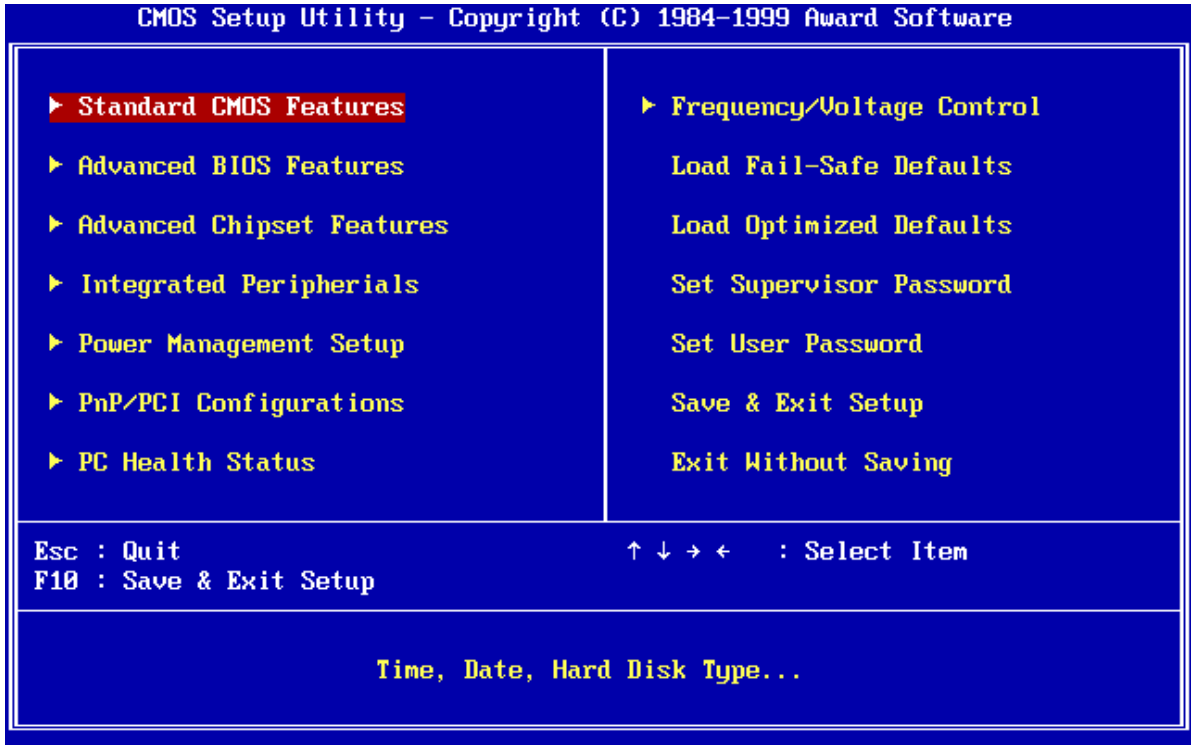
The BIOS

- BIOS (Basic Input/Output System)
  - A mini-OS burned onto a chip
  - First appeared in the CP/M operating system in 1975
- Begins executing a soon as a PC powers on
  - Code from the BIOS chip gets copied to RAM

Accessing and configuring the BIOS



- BIOS often has configurable options
  - Values stored in battery-backed CMOS memory



Initialising Devices

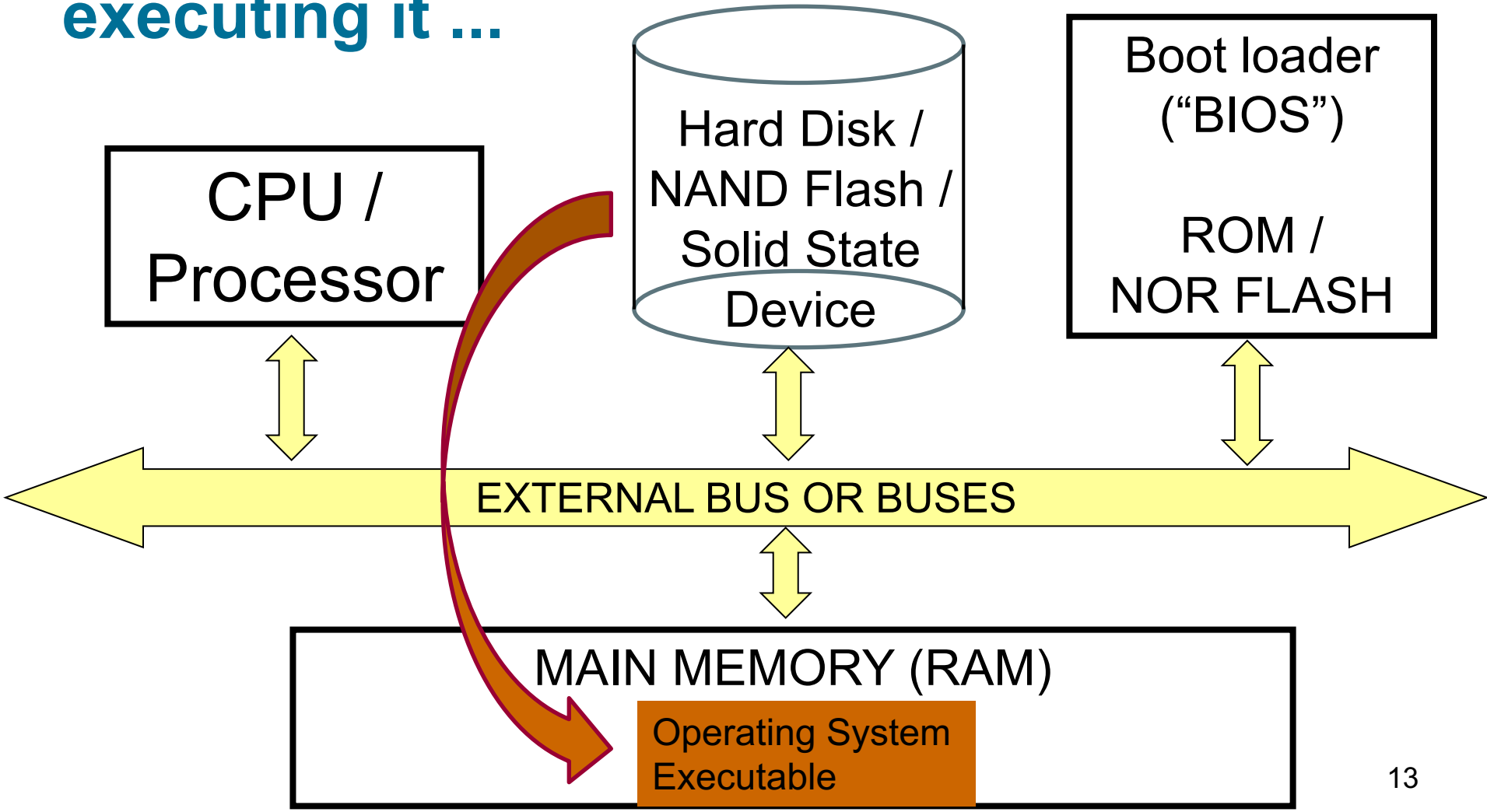
- Scans, checks and initialises hardware
  - CPU and memory
  - Keyboard and mouse
  - Video
  - Bootable storage devices
- Installs interrupt handlers in memory
  - Builds the Interrupt Vector Table …more later
- Runs additional BIOSes on expansion cards
  - Video cards and SCSI cards often have their own BIOS
- Runs POST test
  - Check RAM by read/write to each address

Bootstrapping

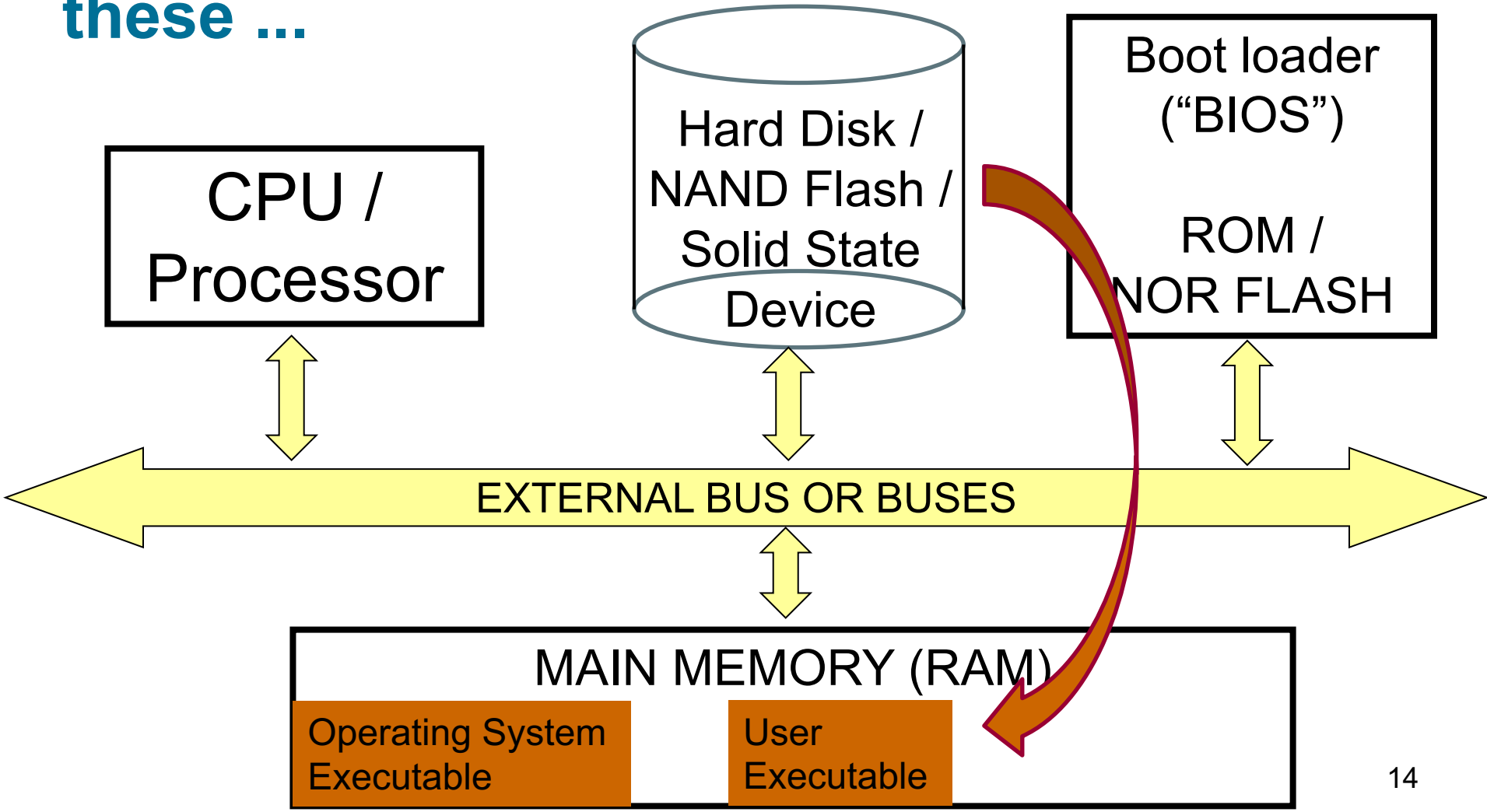
Find and load a real OS

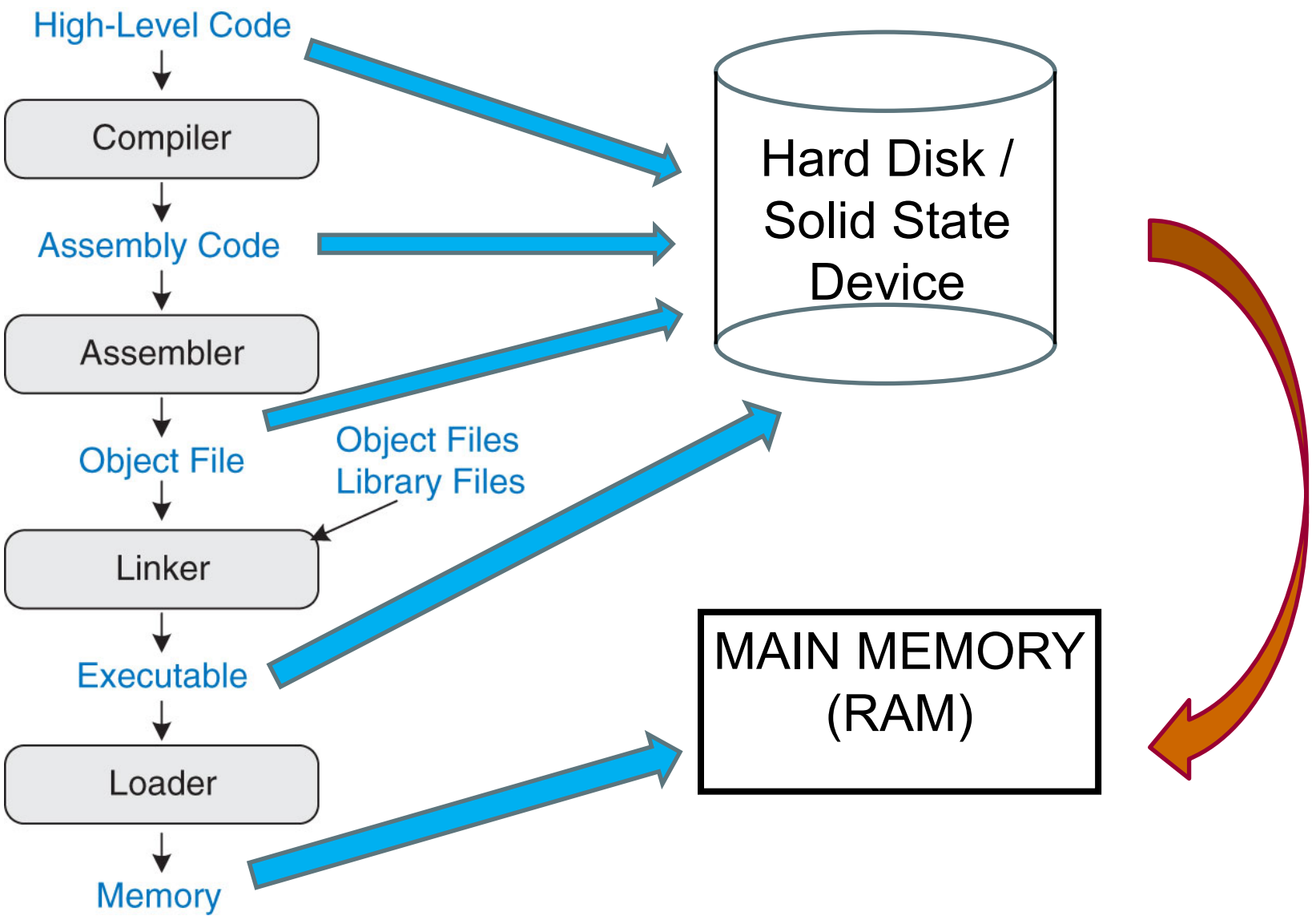
- BIOS identifies all potentially bootable devices
  - Tries to locate Master Boot Record (MBR) on each device
  - Order in which devices are tried is configurable
- MBR has code that can load the actual OS
  - Code is known as a bootloader
  - Load the boot record into RAM
  - Tells the CPU to execute the loaded code
- Example bootable devices:
  - Hard drive, SSD, floppy disk, CD/DVD/Blu-ray, USB flash drive, network interface card.

2. These transfer an operating systems from disk / Flash / SSD into RAM and starts executing it ...



3. User controls machine with OS and can load user executables into RAM and run these ...





Harris & Harris has this example of high level C-code being compiled ...

Code Example 6.30 COMPILING A HIGH-LEVEL PROGRAM

High-Level Code

```
int f, g, y; // global variables



int main(void)
{
  f = 2;
  g = 3;
  y = sum(f, g);
  return y;
}



int sum(int a, int b) {
  return (a + b);
}
```

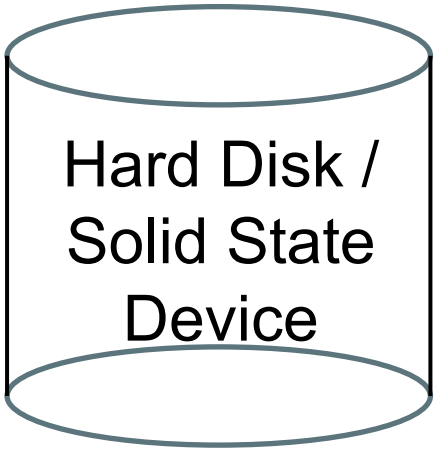
MIPS Assembly Code

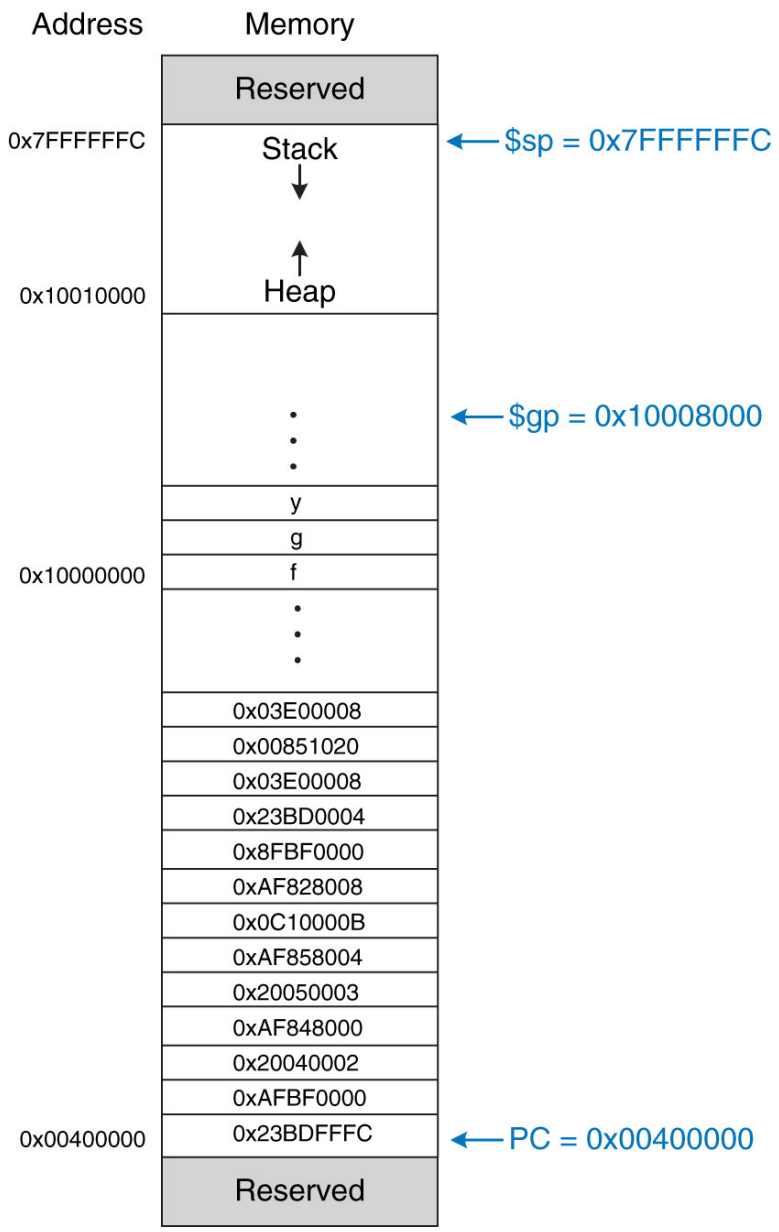
```
.data
f:
g:
y:

.text
main:
  addi $sp, $sp, -4  # make stack frame
  sw   $ra, 0($sp)   # store $ra on stack
  addi $a0, $0, 2    # $a0 = 2
  sw   $a0, f        # f = 2
  addi $a1, $0, 3    # $a1 = 3
  sw   $a1, g        # g = 3
  jal  sum           # call sum function
  sw   $v0, y        # y = sum(f, g)
  lw   $ra, 0($sp)   # restore $ra from stack
  addi $sp, $sp, 4   # restore stack pointer
  jr   $ra           # return to operating system

sum:
  add  $v0, $a0, $a1 # $v0 = a + b
  jr   $ra           # return to caller
```

Executable File on Disk

Executable file header	Text Size	Data Size
	0x34 (52 bytes)	0xC (12 bytes)
Text segment	**Address**	**Instruction**
	0x00400000	0x23BDFFFC
	0x00400004	0xAFBF0000
	0x00400008	0x20040002
	0x0040000C	0xAF848000
	0x00400010	0x20050003
	0x00400014	0xAF858004
	0x00400018	0x0C10000B
	0x0040001C	0xAF828008
	0x00400020	0x8FBF0000
	0x00400024	0x23BD0004
	0x00400028	0x03E00008
	0x0040002C	0x00851020
	0x00400030	0x03E00008
Data segment	**Address**	**Data**
	0x10000000	f
	0x10000004	g
	0x10000008	y

```
addi $sp, $sp, –4
sw   $ra, 0($sp)
addi $a0, $0, 2
sw   $a0, 0x8000($gp)
addi $a1, $0, 3
sw   $a1, 0x8004($gp)
jal  0x0040002C
sw   $v0, 0x8008($gp)
lw   $ra, 0($sp)
addi $sp, $sp, –4
jr   $ra
add  $v0, $a0, $a1
jr   $ra
```





MAIN MEMORY (RAM)

Lecture Summary

- In this lecture I wanted to give you a high-level view of the key components making up different types of computers and how these components interact together.

- We also covered the sequence of operations that are executed when you turn on your computer
  - Beginning with the starting-up of the BIOS to the loading of the operating system in memory

- In the next lecture we will start to look at the fundamentals of operating systems ...