USC Viterbi School of Engineering

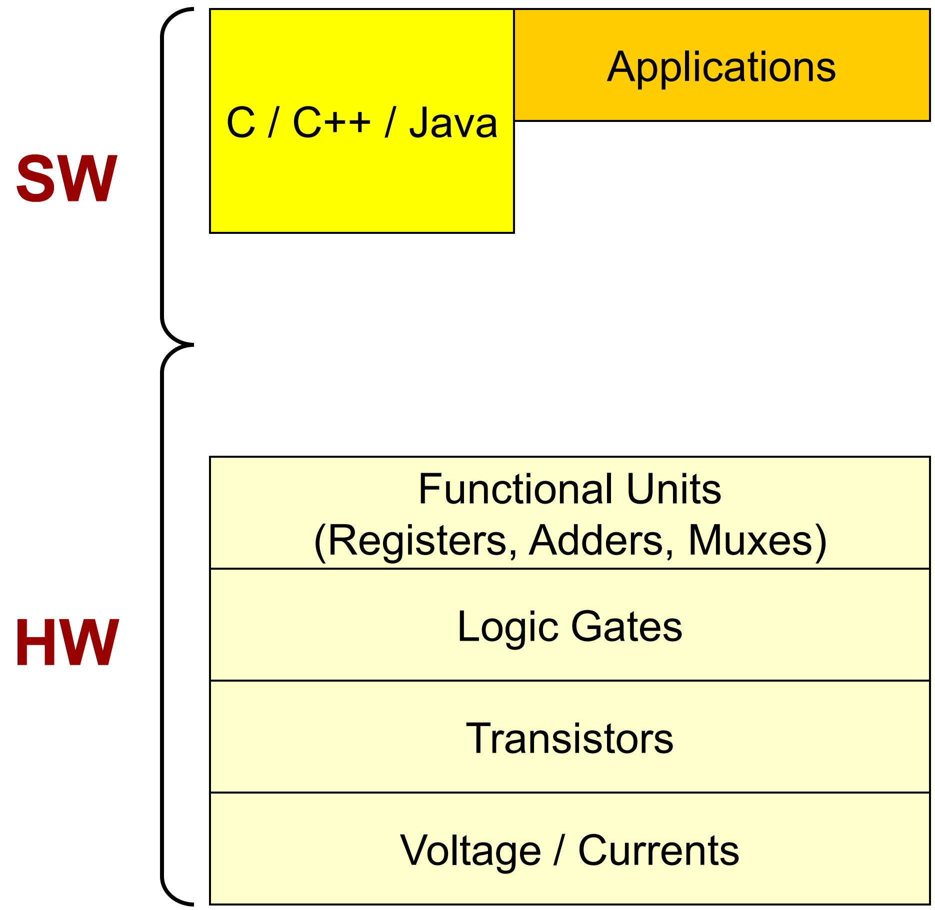
EE 357 Unit 0

Class Introduction

Basic Hardware Organization

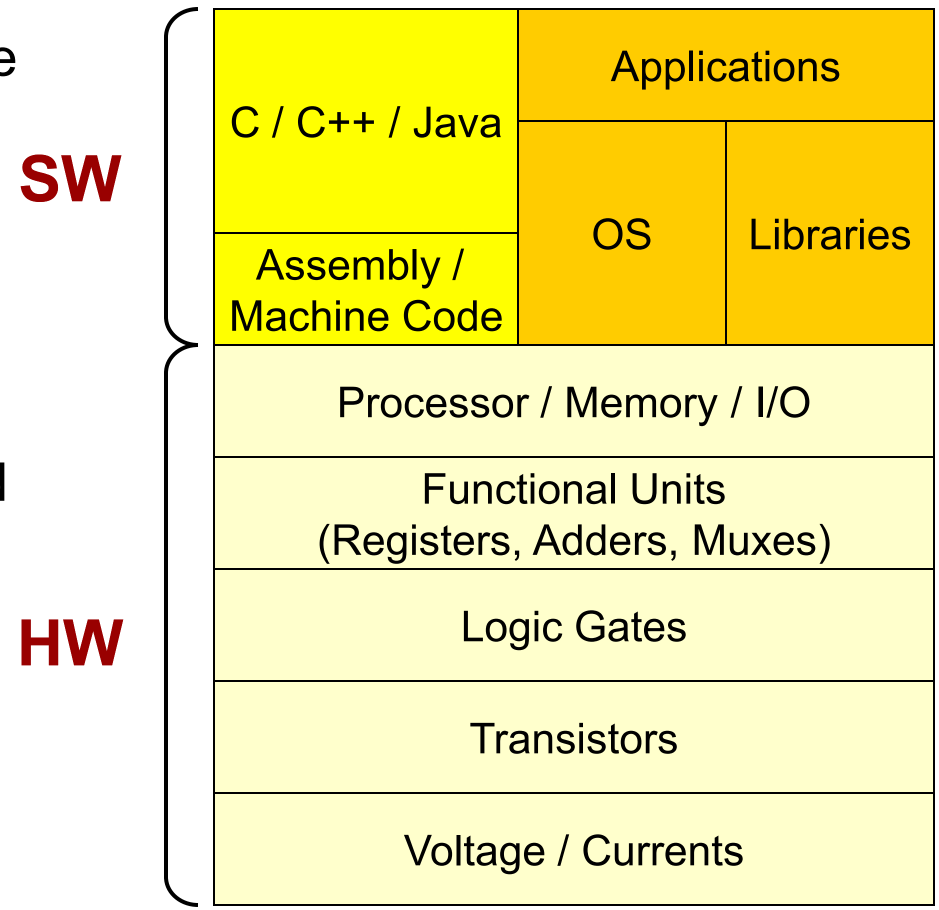
Computer Systems Abstractions

- CS 101,102
  - Programming with high-level languages (HLL's) like C / C++/ Java
- EE 101,201
  - Digital hardware (registers, adders, muxes)



Computer Systems Abstractions

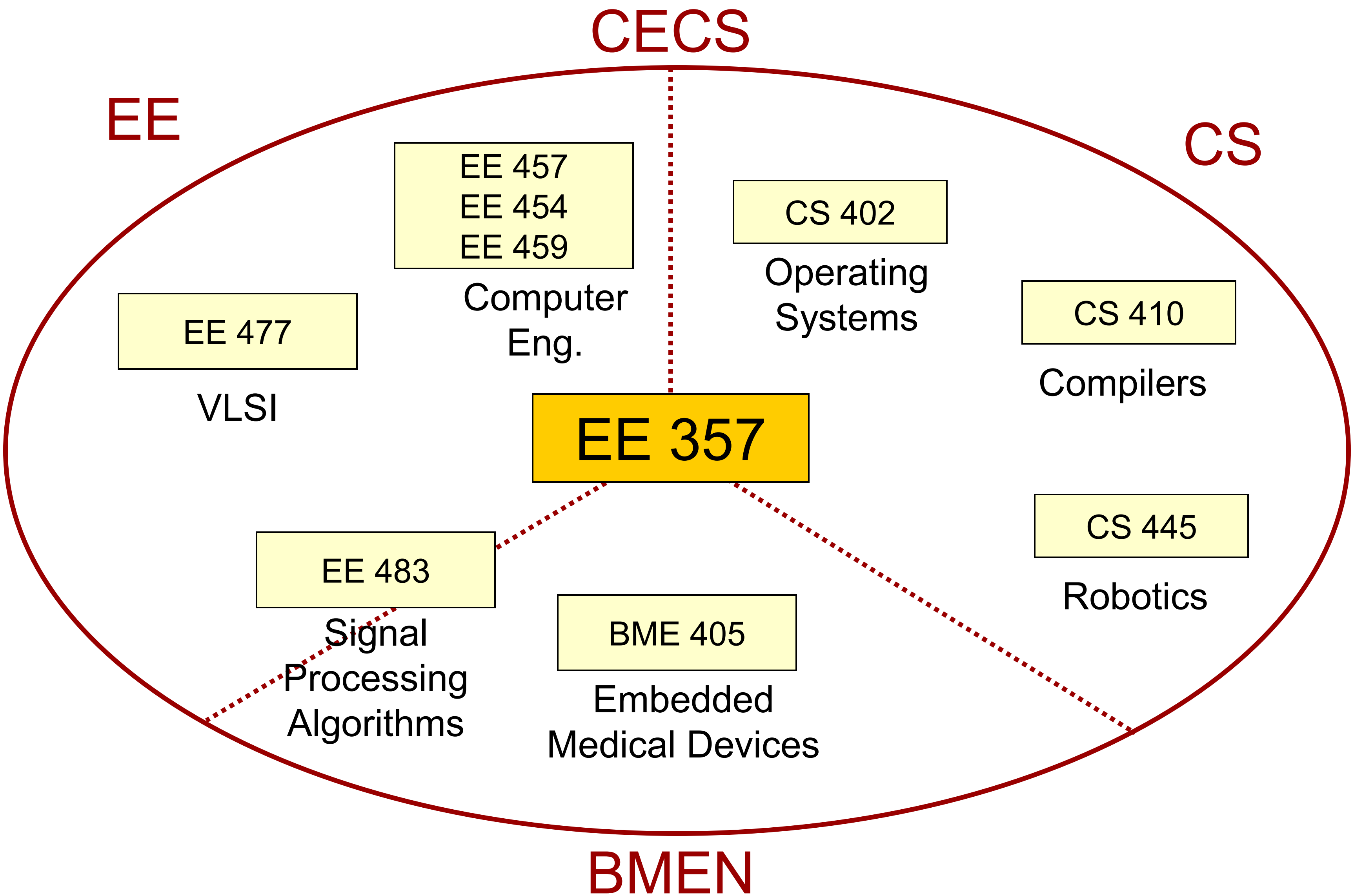
- CS 101,102
  - Programming with high-level languages (HLL's) like C / C++/ Java
- EE 101,201
  - Digital hardware (registers, adders, muxes)
- EE 357
  - Computer organization and architecture
    - *HW/SW System Perspective*
  - Topics
    - HW/SW interface
    - System Software
    - Assembly Language
    - Computer Architecture



EE 357

- Focus on assembly language & embedded systems
  - What are the basic software instructions and how are they used to implement software programs
  - Programming and low-level bit manipulations
- Focus on computer organization/architecture
  - Organization of HW components (proc., mem., I/O) and its effect on software performance
  - Actual design of simple processor and other system components
- Focus on application and learn-by-example
  - Be prepared to experiment…don't just go through the motions
  - Learn to learn

EE 357 in Context

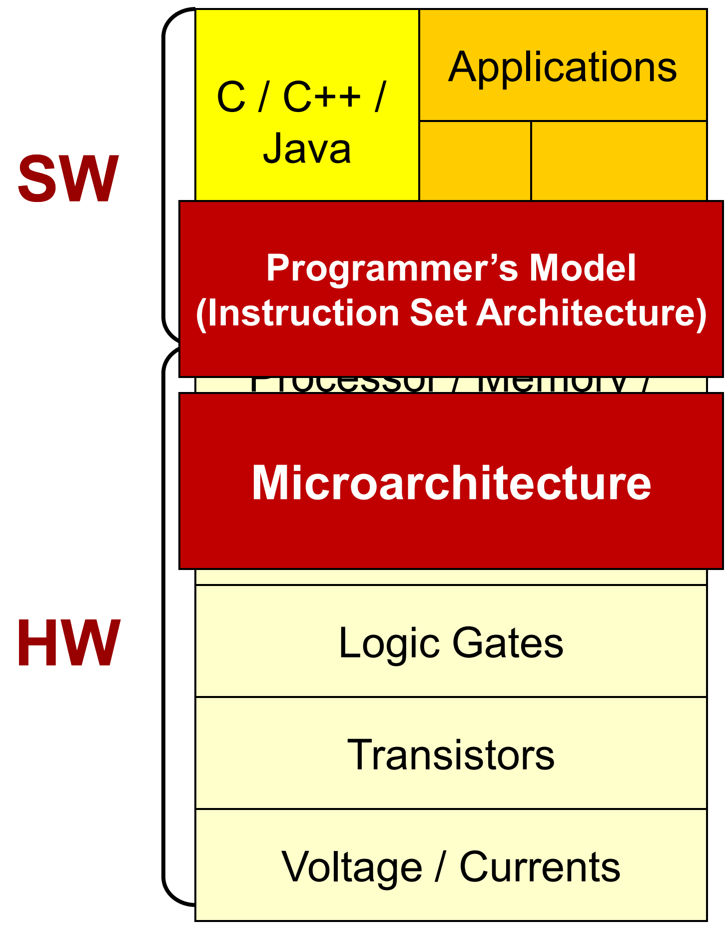


Organization & Architecture

- Computer organization refers to the components and interconnection necessary to form a computer system
- Computer architecture refers to a specific organization of components and other design choices

Levels of Architecture

- System architecture
  - High-level HW org.
- Instruction Set Architecture
  - A contract or agreement about what the HW will support and how the programmer can write SW for the HW
  - Vocabulary that the HW understands and SW is composed of
- Microarchitecture
  - HW implementation for executing instructions
  - Usually transparent to SW programs but not program performance
  - Example: Intel and AMD have different microarchitectures but support essentially the same instruction set

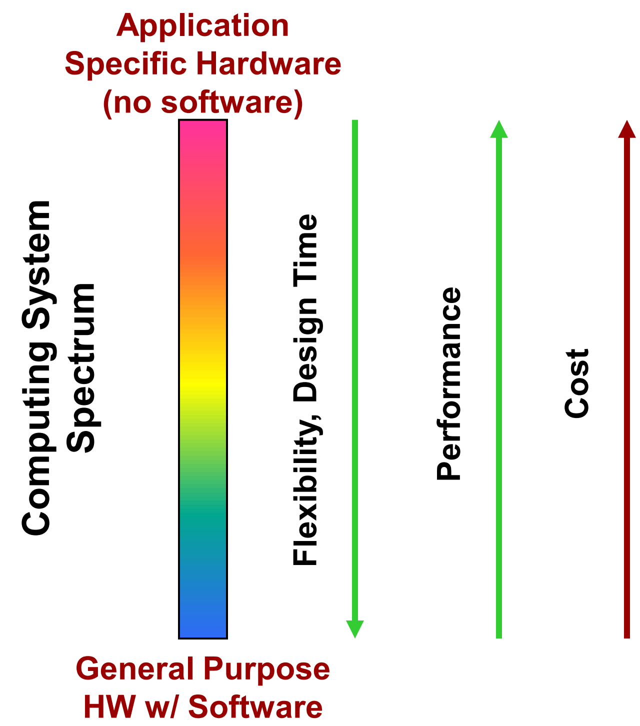


Why is Architecture Important

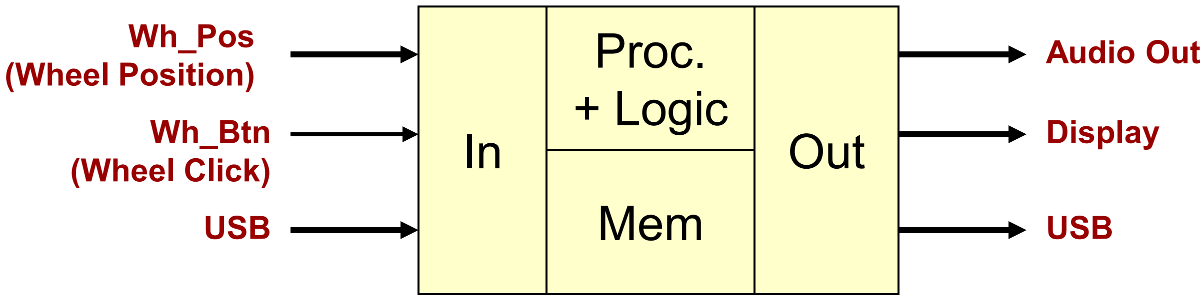
- Enabling ever more powerful computers
- Different systems require different architectures
  - PC's
  - Servers
  - Embedded Systems
    - Simple control devices like ATM's, toys, appliances
    - Media systems like game consoles and MP3 players
    - Robotics

Digital System Spectrum

- Key idea: Any “algorithm” can be implemented in HW or SW or some mixture of both
- A digital systems can be located anywhere in a spectrum of:
  - ALL HW: (a.k.a. Application-Specific IC’s)
  - ALL SW: An embedded computer system
- Advantages of application specific HW
  - Faster, less power
- Advantages of an embedded computer system (i.e. general purpose HW for executing SW)
  - Reprogrammable (i.e. make a mistake, fix it)
  - Less expensive than a dedicated hardware system (single computer system can be used for multiple designs)
- MP3 Player: System-on-Chip (SoC) approach
  - Some dedicated HW for intensive MP3 decoding operations
  - Programmable processor for UI & other simple tasks



Embedded Example: iPod™



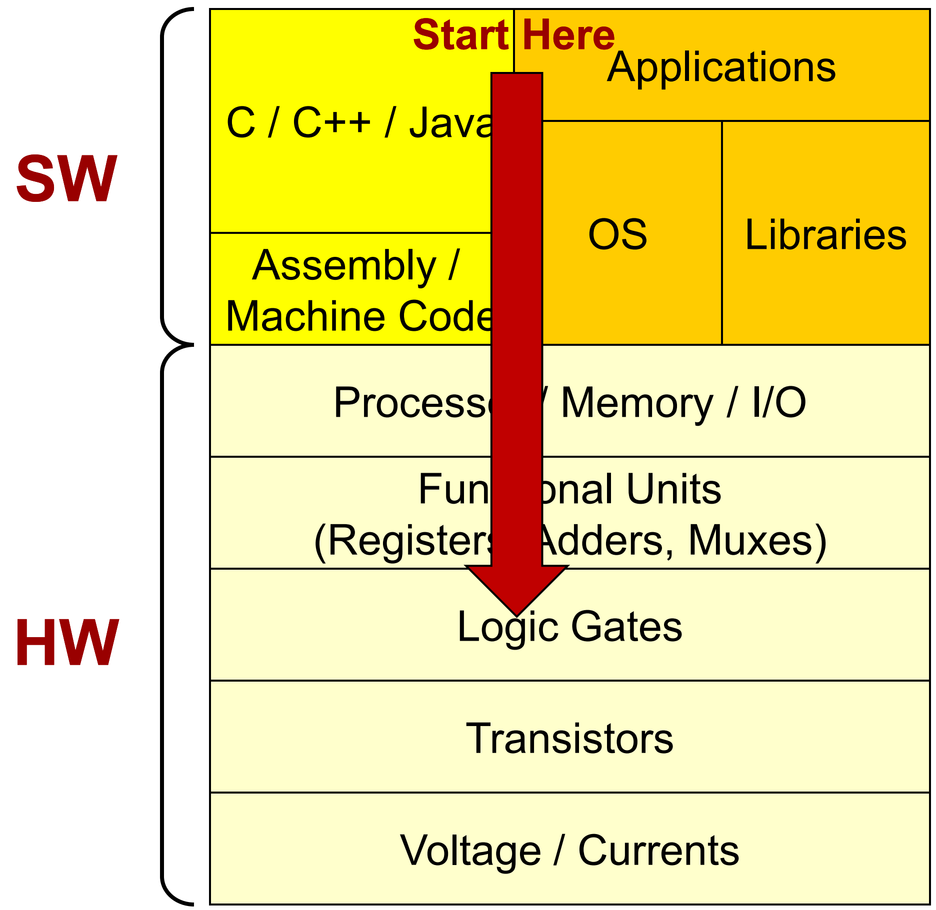
Using an embedded computer system we can write software code to control I/O rather than designing state machines, etc.

```
if (wh_btn && wh_pos==PLAY)
  {
  load_selected_file();
  play_file();
  start_time();
  }
else if (…)

void start_time(){
  time = 0;
  while (PLAYING) {
    sleep_1sec();
    time = time + 1;
    display_time(time);
  }
}
```

Computer Systems Tour

- How does a SW program get mapped and executed on a computer
- What components make a computer system and what are their functions
- How does the architecture affect performance



Software Process

Software Program

High Level Language Description

```
if (x > 0)
  x = x + y - z;
a = b*x;
```

.c/.cpp files

Compiler

```
      MOVE.L X,D0
      CMPI   #0,D0
      BLE    SKIP
      ADD    Y,D0
      SUB    Z,D0
SKIP  MUL    …
```

Assembly (.asm/.s files)

Assembler

```
1110 0010 0101 1001
0110 1011 0000 1100
0100 1101 0111 1111
1010 1100 0010 1011
0001 0110 0011 1000
```

Object/Machine Code (.o files)

Linker

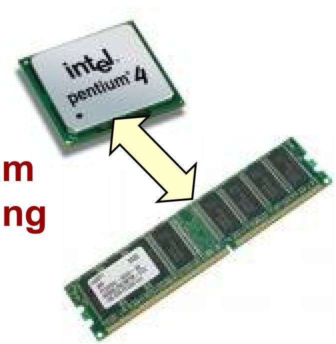
```
1110 0010 0101 1001
0110 1011 0000 1100
) 1101 0111 1111
) 1100 0010 1011
0110 0011 1000
```

Executable Binary Image

A "compiler" (i.e. gcc, VisualC++, etc.) includes the assembler & linker

In EE 357 you will be able to perform all the tasks of the compiler…

Loader / OS



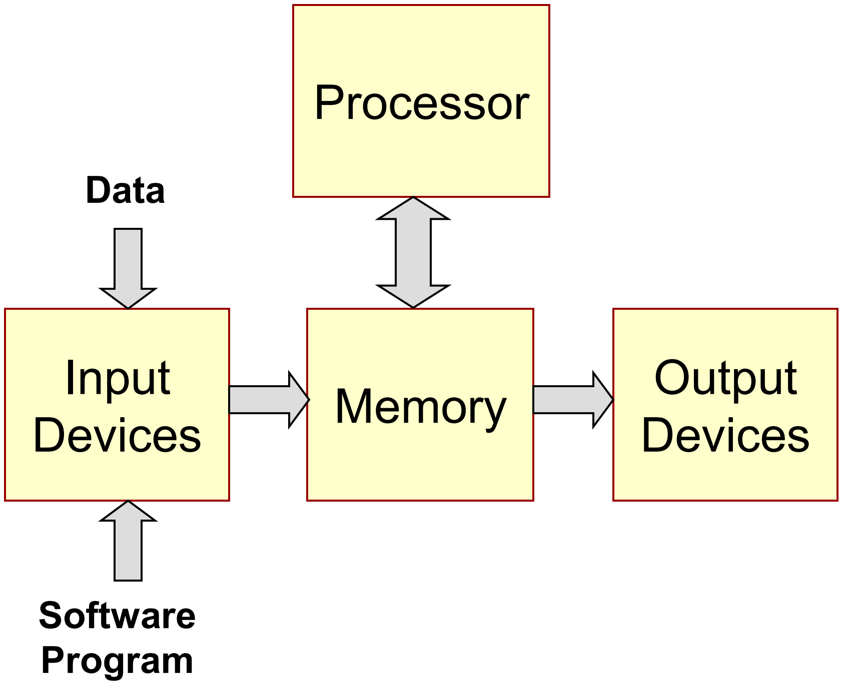
Program Executing

Compiler Process

- A compiler such as ‘gcc’ performs 3 tasks:
  - Compiler
    - Converts HLL (high-level language) files to assembly
  - Assembler
    - Converts assembly to object (machine) code
  - Static Linker
    - Links multiple object files into a single executable resolving references between code in the separate files
  - Output of a compiler is a binary image that can be loaded into memory and then executed.
- Loader/Dynamic Linker
  - Loads the executable image into memory and resolves dynamic calls (to OS subroutines, libraries, etc.)

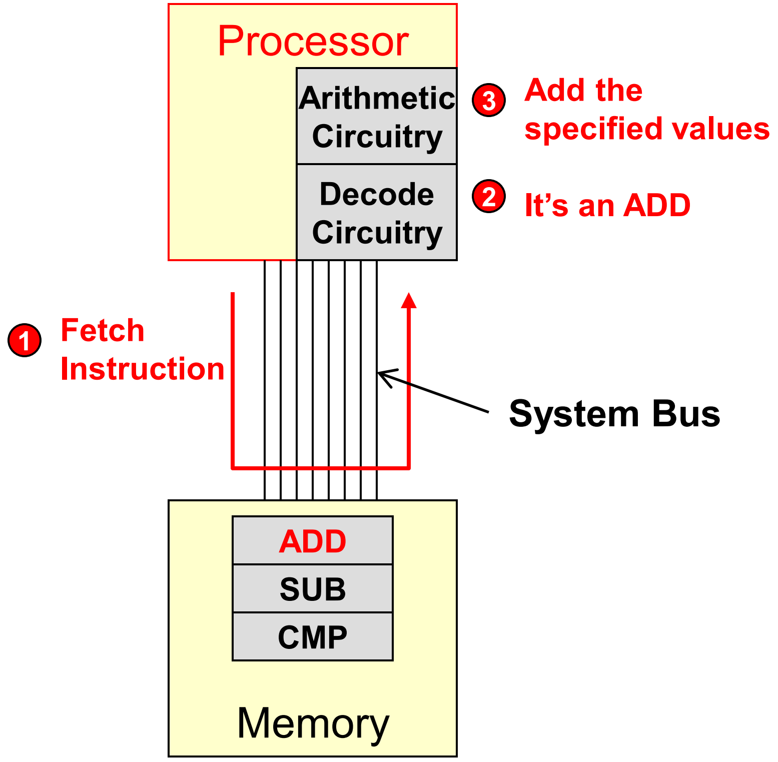
Hardware Components

- Processor
  - Executes the program and performs all the operations
  - Examples: Pentium 4, PowerPC, M68K/Coldfire
- Main Memory
  - Stores *data* and *program* (*instructions)*
  - Different forms:
    - RAM = read and write but volatile (lose values when power off)
    - ROM = read-only but non-volatile (maintains values when power off)
  - Significantly slower than the processor speeds
- Input / Output Devices
  - Generate and consume data from the system
  - Examples: Keyboard, Mouse, CD-ROM, Hard Drive, USB, Monitor display
  - MUCH, MUCH slower than the processor



Processor

- Performs the same 3-step process over and over again
  - Fetch an instruction from memory
  - Decode the instruction
    - Is it an ADD, SUB, etc.?
  - Execute the instruction
    - Perform the specified operation
- This process is known as the **Instruction Cycle**

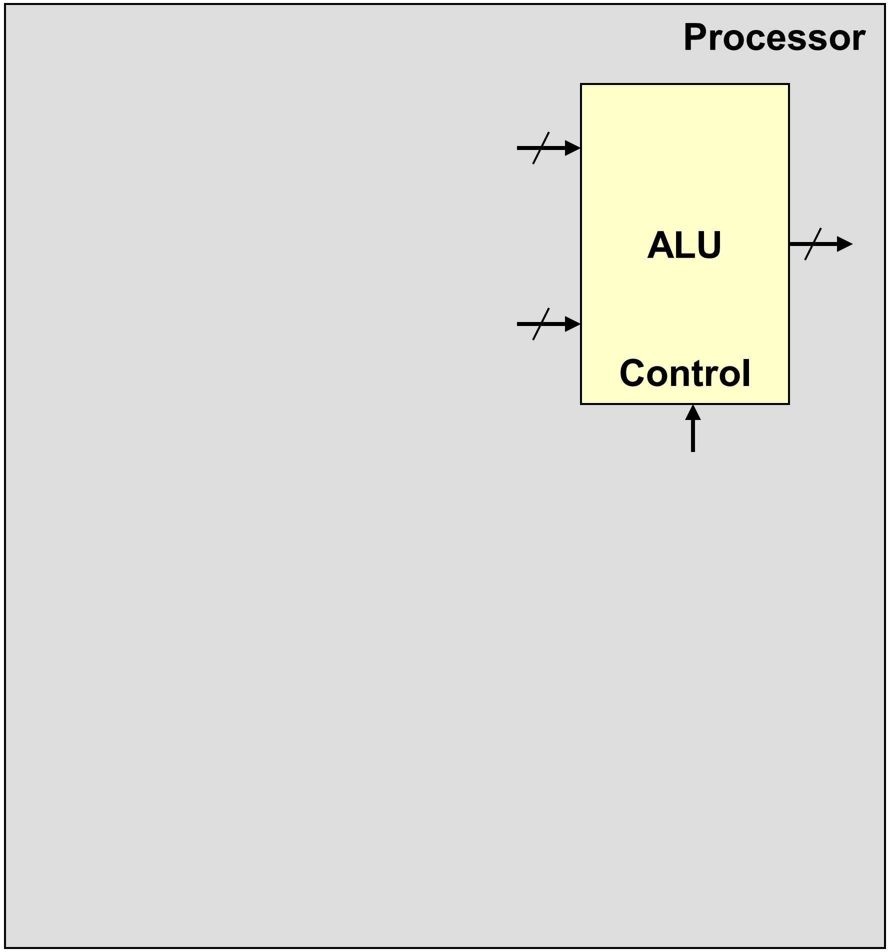


Processors

- Processors contain 4 subcomponents
  1. ALU (Arithmetic & Logical Unit)
  2. Registers
  3. Control Circuitry & System-Bus Interface
  4. Cache (Optional)

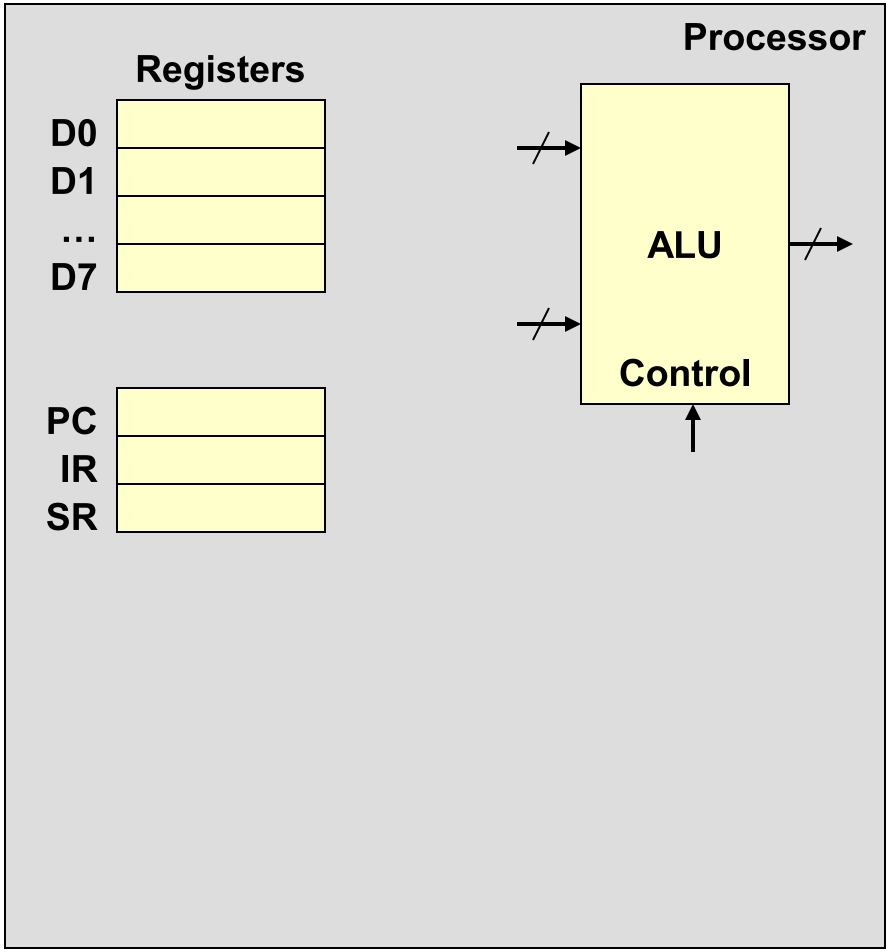
ALU

- Performs arithmetic and logical operations
- 2 inputs and 1 output value
- Control inputs to select operation (ADD, SUB, AND, OR…)



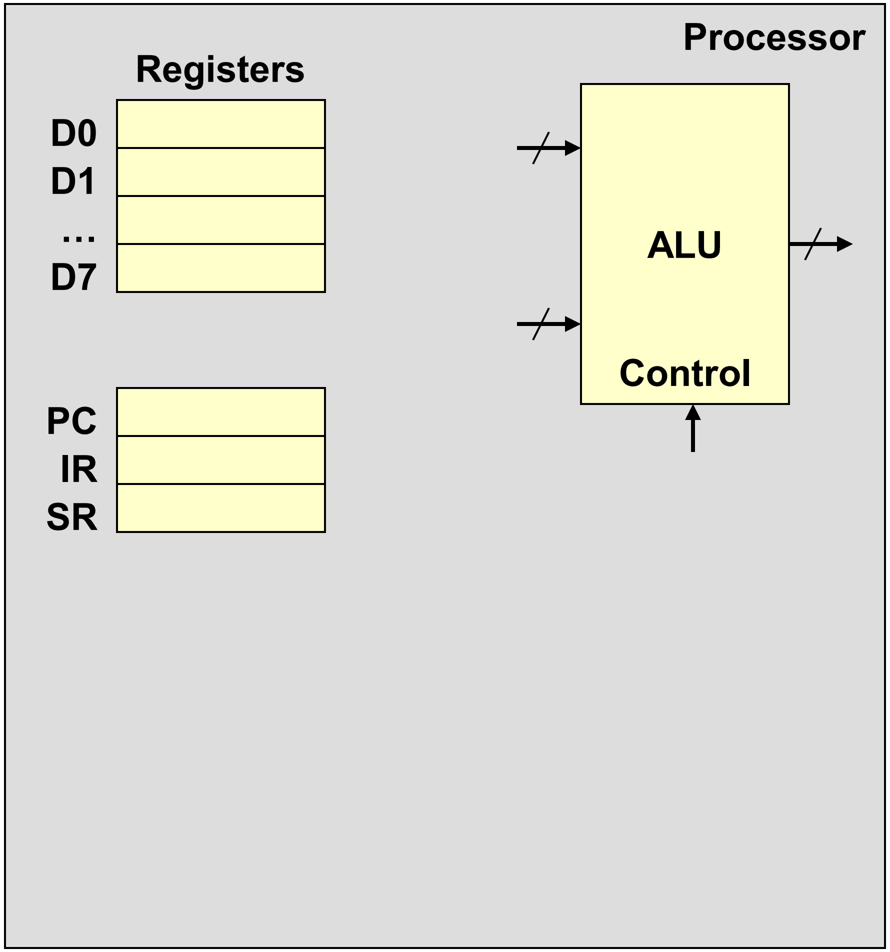
Registers

- Provide temporary storage for data
- 2 categories of registers
  - General Purpose Registers (GPR's)
    - for program data
    - can be used by programmer as desired
    - given names (e.g. D0-D7)
  - Special Purpose Registers
    - for internal processor operation (not for program data)



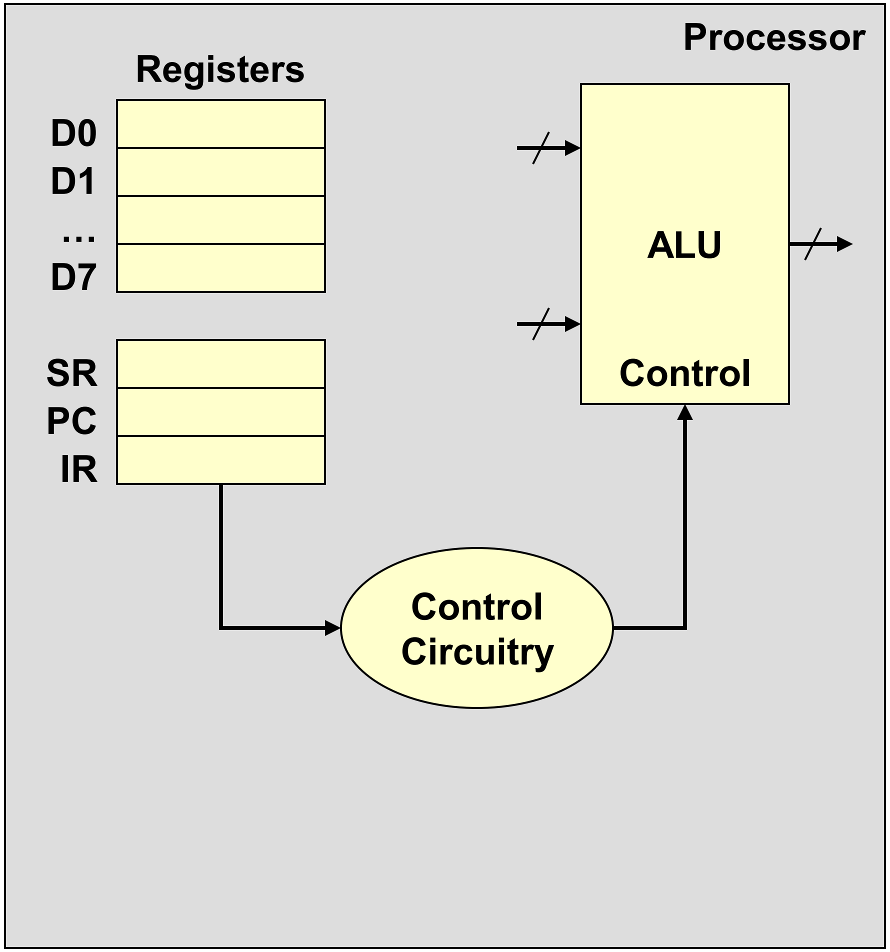
Registers

- GPR's
  - Faster to access than main memory
  - Keep data you are working with in registers to speed up execution
- Special Purpose Reg's.
  - Hold specific information that the processor needs to operate correctly
  - PC (Program Counter)
    - Pointer to (address of) instruction in memory that will be executed next
  - IR (Instruction Register)
    - Stores the instruction while it is being executed
  - SR (Status Register)
    - Stores status/control info



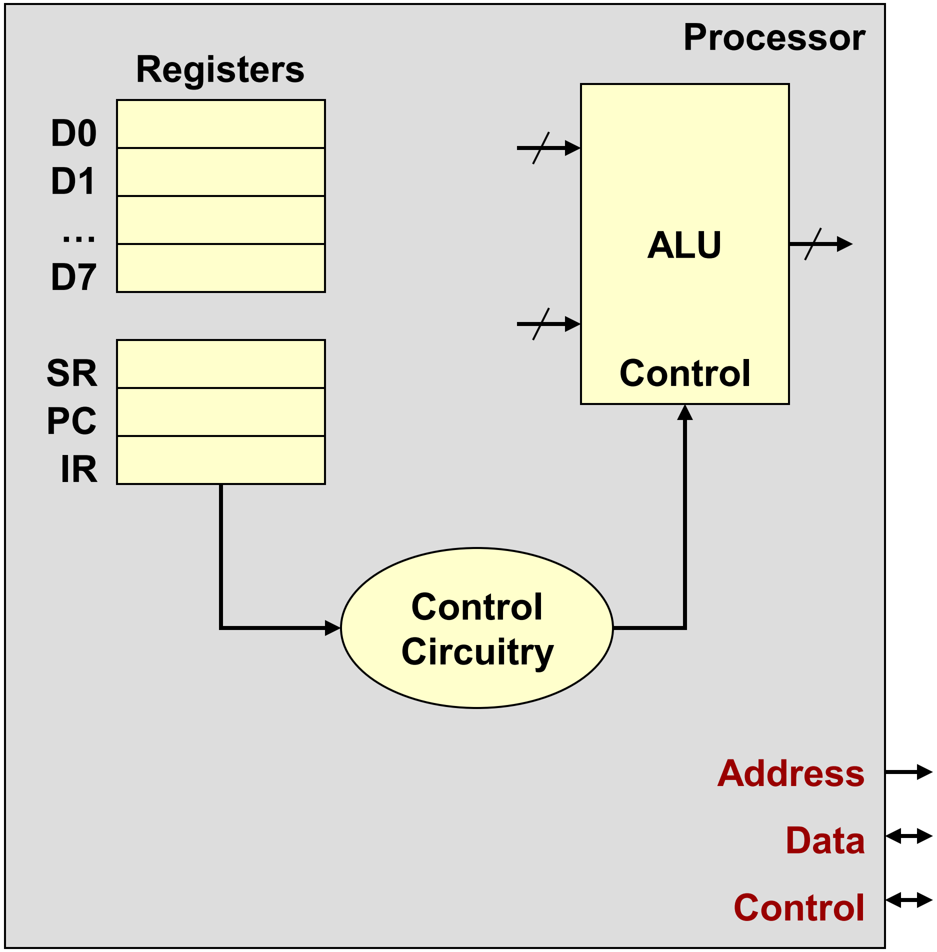
Control Circuitry

- Decodes each instruction
- Selects appropriate registers to use
- Selects ALU operation
- And more…



System Bus Interface

- System bus is the means of communication between the processor and other devices
  - Address
    - Specifies location of instruction or data
  - Data
  - Control



Memory

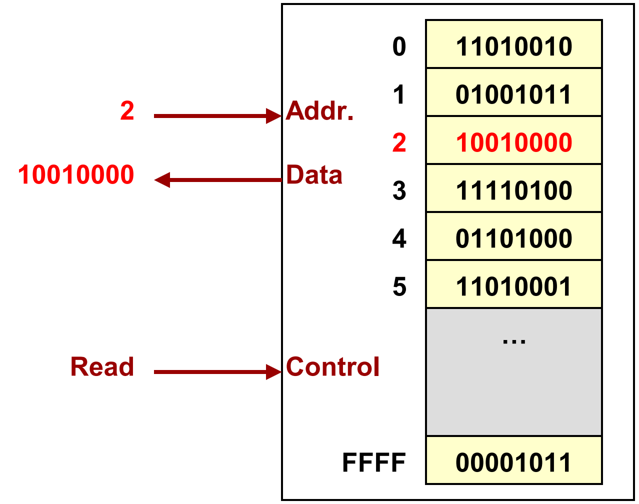
- Set of cells that each store a group of bits (usually, 1 byte = 8 bits)
- Unique address assigned to each cell
  - Used to reference the value in that location
- Numbers and instructions are all represented as a string of 1's and 0's

Address	Data
0	11010010
1	01001011
2	10010000
3	11110100
4	01101000
5	11010001
	...
FFFF	00001011

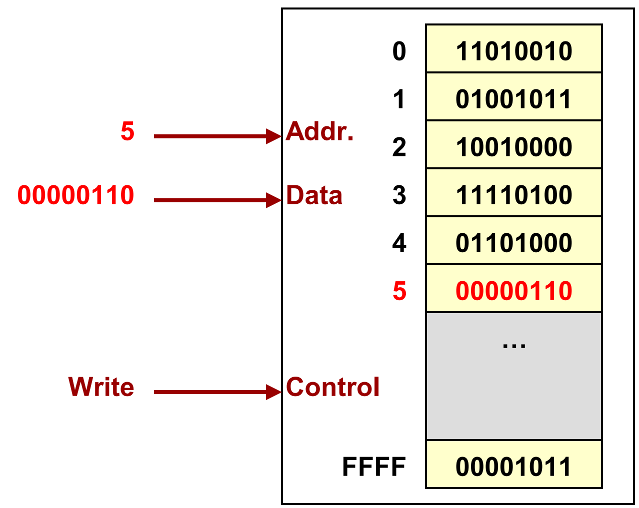
Memory Device

Memory Operations

- Memories perform 2 operations
  - Read: retrieves data value in a particular location (specified using the address)
  - Write: changes data in a location to a new value
- To perform these operations a set of address, data, and control inputs/outputs are used
  - Note: A group of wires/signals is referred to as a 'bus'
  - Thus, we say that memories have an address, data, and control bus.



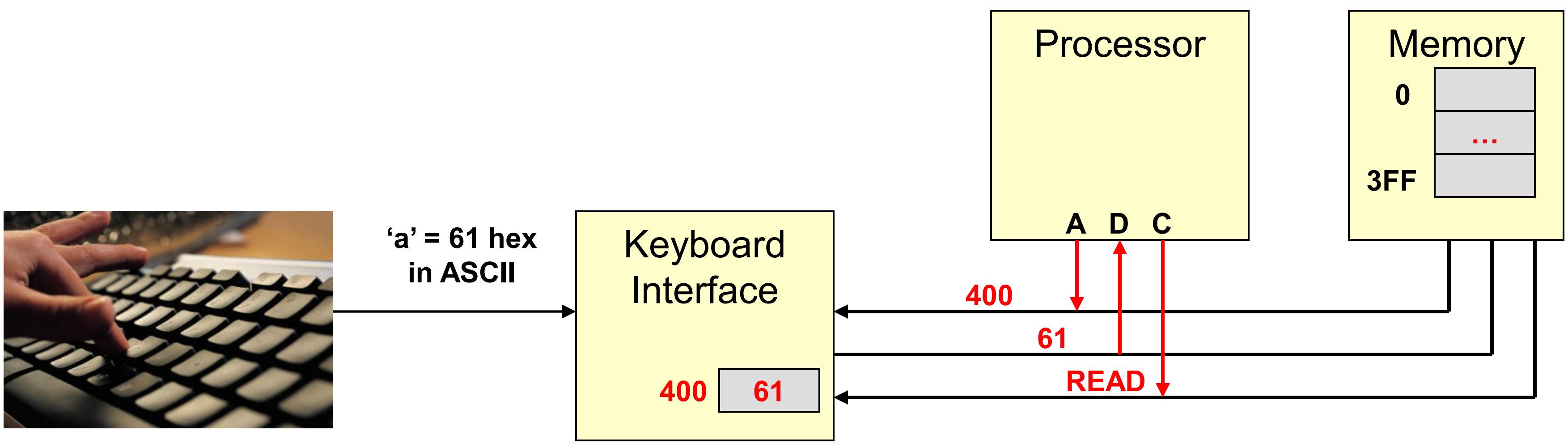
A Read Operation



A Write Operation

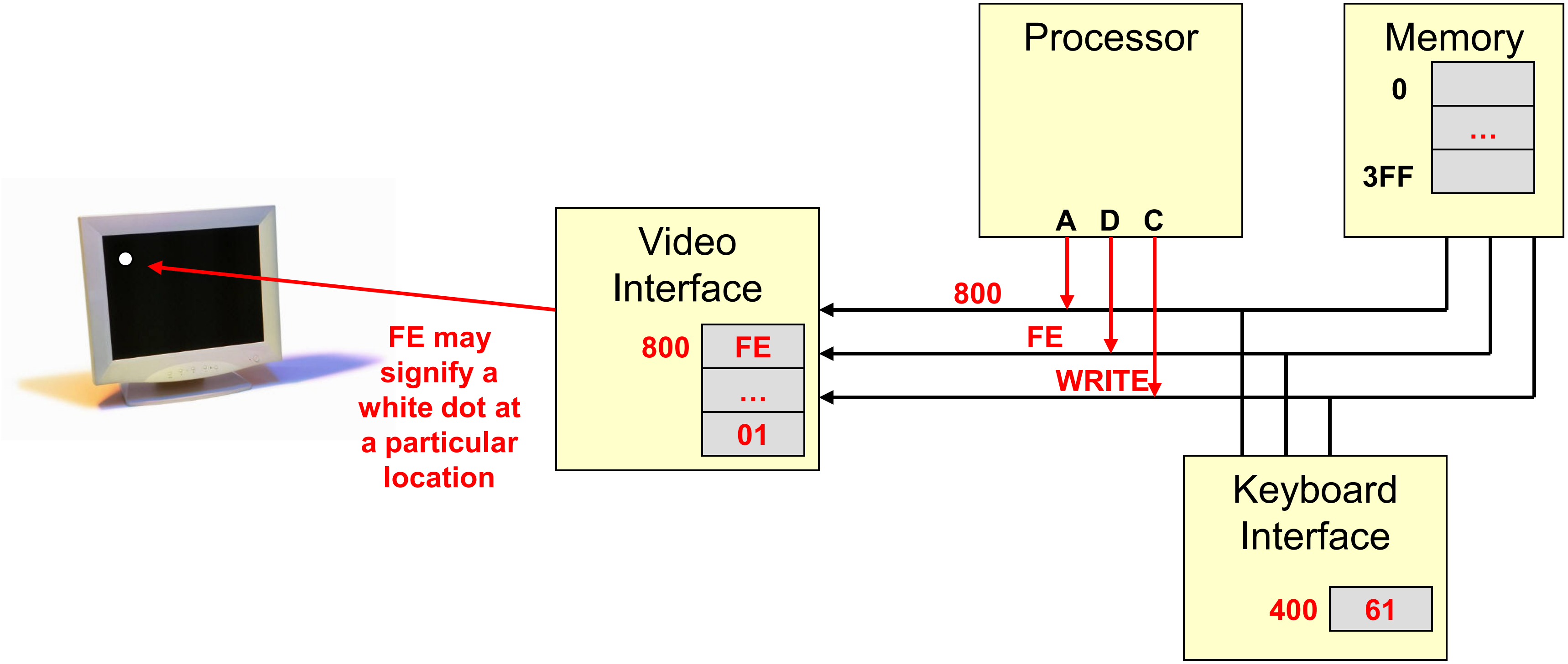
Input / Output

- Keyboard, Mouse, Display, USB devices, Hard Drive, Printer, etc.
- Processor can perform reads and writes on I/O devices just as it does on memory
  - I/O devices have locations that contain data that the processor can access
  - These locations are assigned unique addresses just like memory



Input / Output

- Writing a value to the video adapter can set a pixel on the screen



Computer Organization Issues

- Components run at different speeds
  - Processor can perform operations very quickly (~ 1 ns)
  - Memory is much slower (~ 50 ns) due to how it is constructed & its shear size [i.e. it must select/look-up 1 location from millions]
    - Speed is usually inversely proportional to size (i.e. larger memory => slower)
  - I/O devices are much slower
    - Hard Drive (~ 1 ms)
  - Intra-chip signals (signals w/in the same chip) run much faster than inter-chip signals
- Design HW and allocate HW resources to accommodate these inherent speed differences

Unit 0: Moore's Law

CONTEMPORARY ISSUE

Architecture Issues

- Fundamentally, architecture is all about the different ways of answering the question:

  "What do we do with the ever-increasing number of transistors available to us"

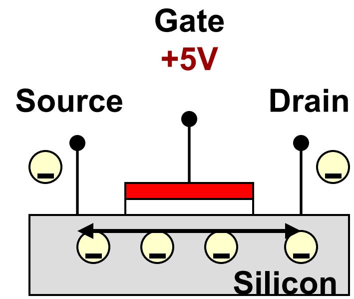
- Architecture takes Moore's Law and produces an equivalent increase in computational ability

Moore’s Law & Transistors

- Moore’s Law = Number of transistors able to be fabricated on a chip will double every 1.5 – 2 years
- Transistors are the fundamental building block of computer HW
  - Switching devices: Can conduct [on = 1] or not-conduct [off = 0] based on an input voltage

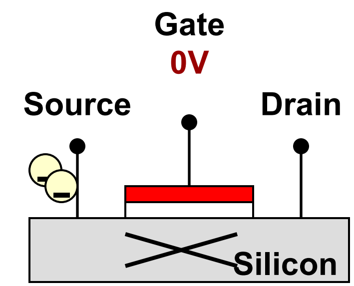
Transistor

- 3-terminal device
  - Gate input: the control input; it's voltage determines whether current can flow
  - Source & Drain: terminals that current flows from/to
- Many transistors can be fabricated on one piece of silicon (i.e. an integrated chip, IC)



Transistor is 'on'

High voltage at gate allows current to flow from source to drain



Transistor is 'off'

Low voltage at gate prevents current from flowing from source to drain

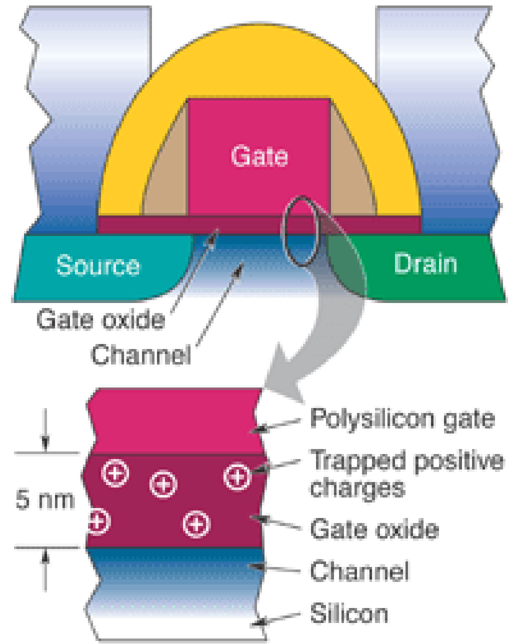
Integrated Circuit

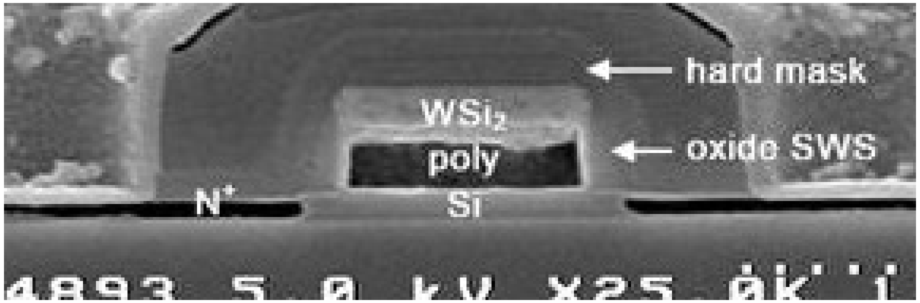
Actual silicon wafer is quite small but can contain ~300 million transistors

Silicon wafer is then packaged to form the chips we are familiar with

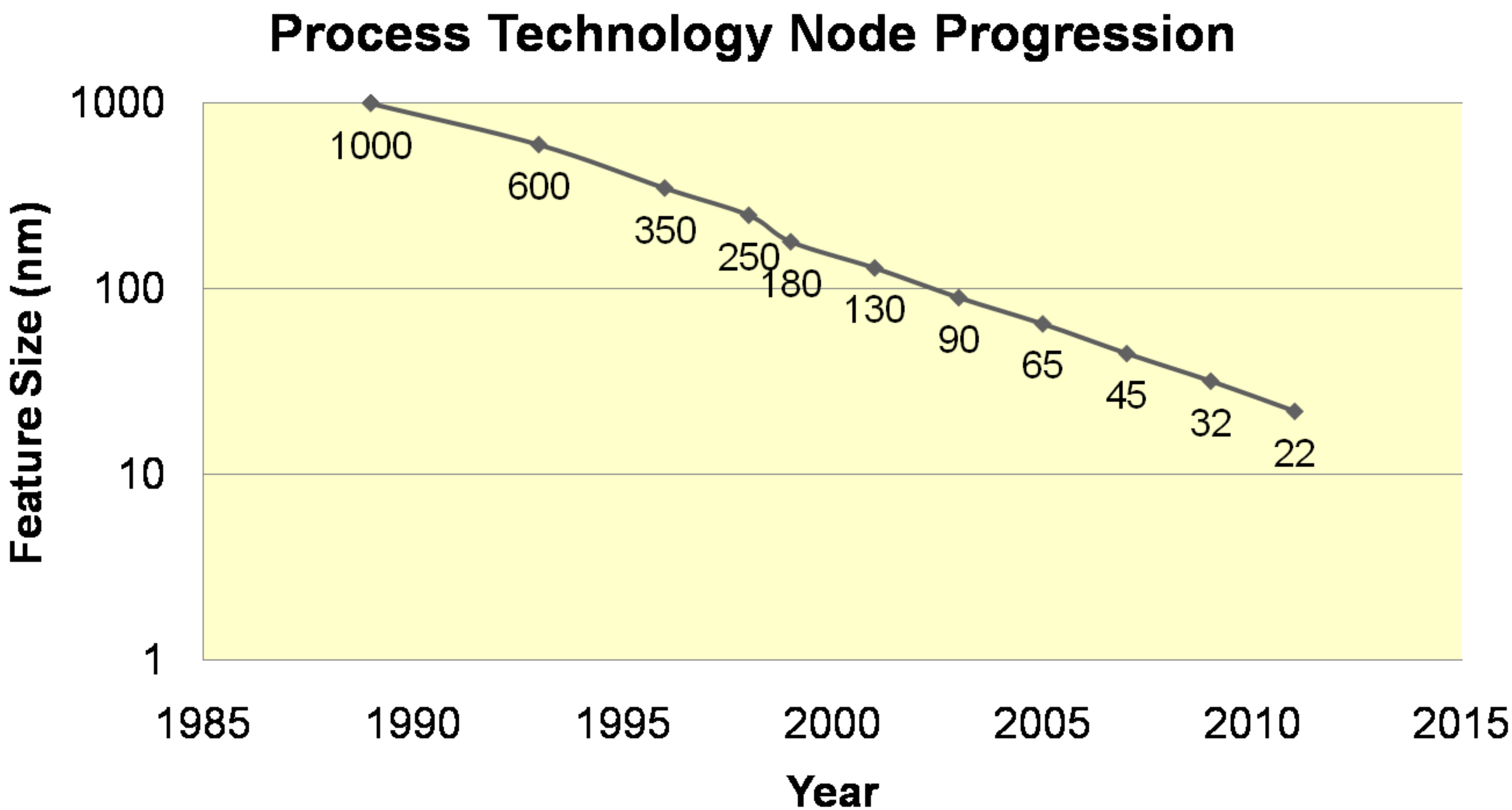
Transistor Physics

- Cross-section of transistors on an IC
- Moore's Law is founded on our ability to keep shrinking transistor sizes
  - Gate/channel width shrinks
  - Gate oxide shrinks
- Transistor feature size is referred to as the implementation "technology node"

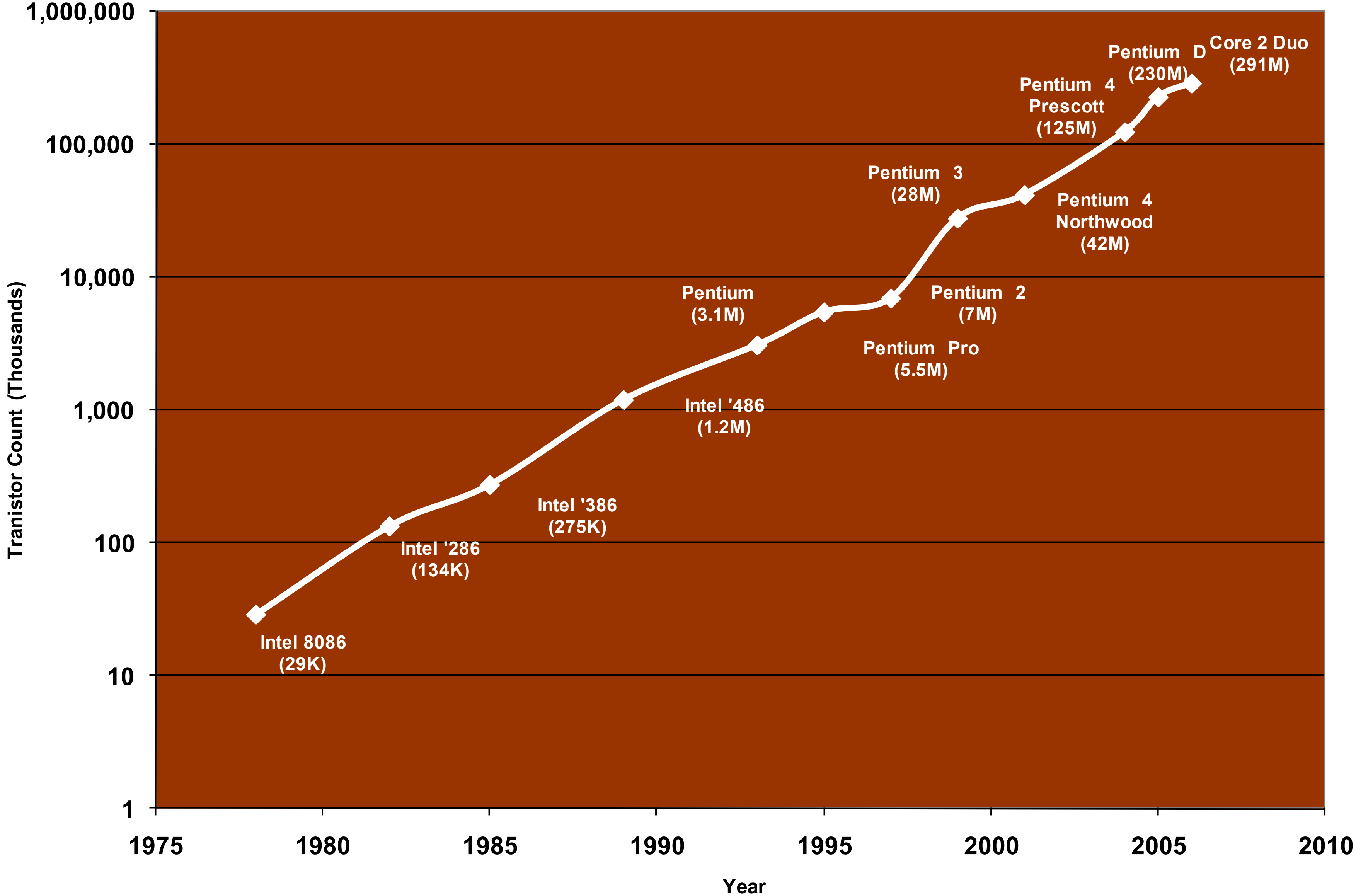




Technology Nodes



Growth of Transistors on Chip



Future of Moore's Law

- What will the next switching technology be?
- How will that affect the way computers are organized, designed, and programmed?

Unit 0: Architecture Issues

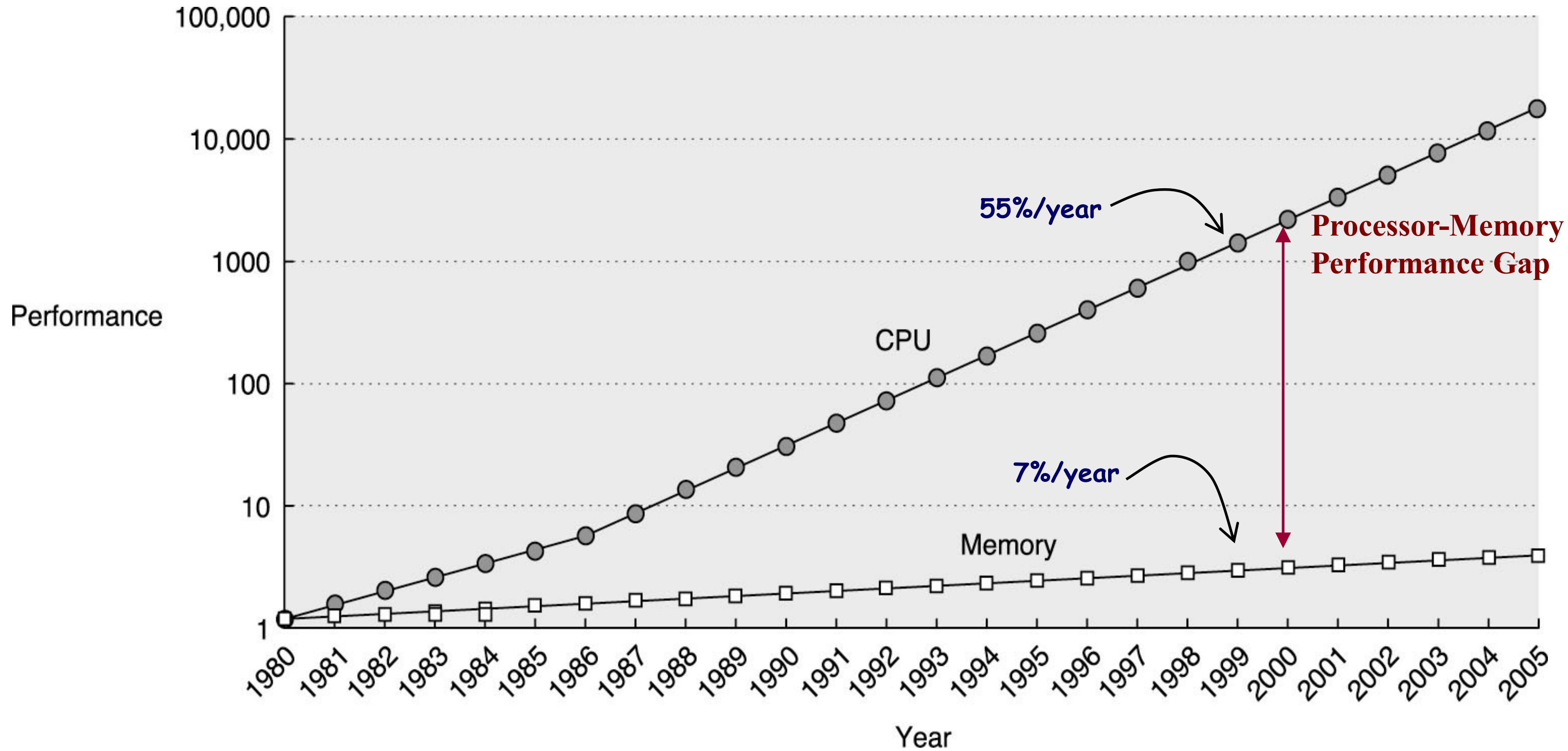
CONTEMPORARY ISSUE

Architecture Issues

- What do we do with the transistors available to us
  - Moore's Law
- How do we mitigate the issues that affect performance
  - Memory wall
  - Power consumption
  - Sequential programming paradigm
  - Reliability

Memory Wall Problem

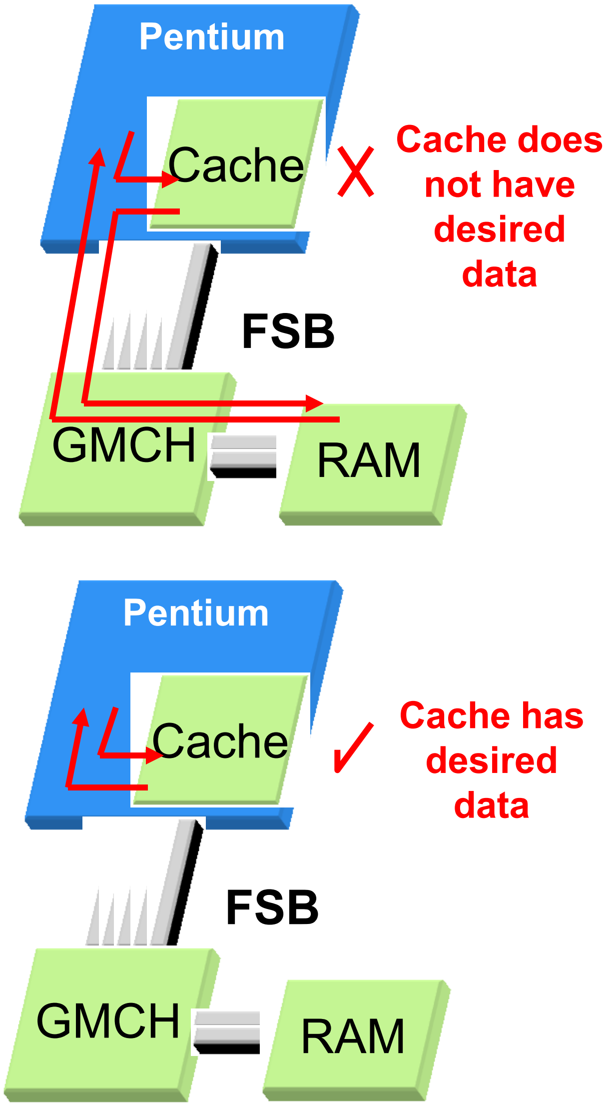
- Processor performance is increasing much faster than memory performance



Hennessy and Patterson,
Computer Architecture –
A Quantitative Approach (2003)

Cache Example

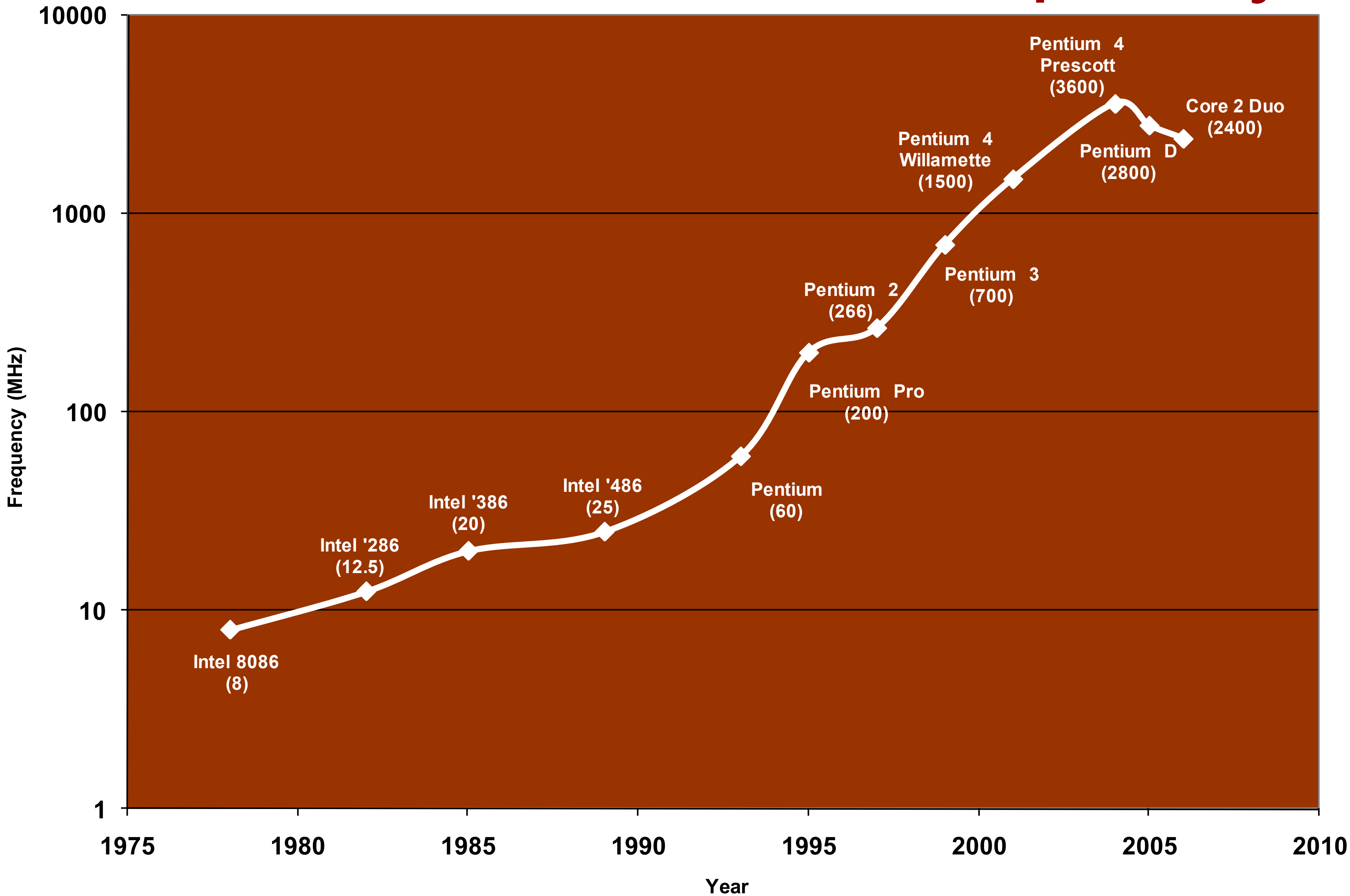
- Small, fast, on-chip memory to store **copies** of recently-used data
- When processor attempts to access data it will check the cache first
  - If the cache does not have the data, it must go to the main memory (RAM) to access it
  - If the cache has the desired data, it can supply it quickly



Power Consumption

- Problem
  - Inability to dissipate heat that results from power consumption can destroy a chip
  - Dynamic power equation: $P = \frac{1}{2} CV^2f$
    - C = Capacitance = function of transistor size
    - V = Supply voltage [Logic '1' voltage]
    - f = Frequency = speed of operation of the processor
- Solution
  - Reduction in frequency also allows reduction in supply voltage (cubic reduction in power…)
    - Implication: Rather than doing one thing fast, do many things a bit more slowly

Increase in Clock Frequency

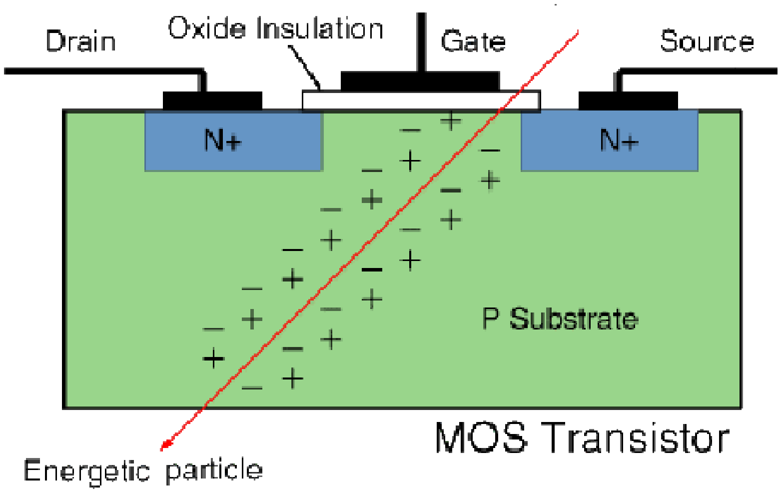


Sequential Programming Paradigm

- Problem
  - Traditional programming paradigm has been a single (sequential) thread of execution (easy for programmer)
  - Now HW is able to do MANY things in parallel (at the same time) with multicore and other architectures
- Solution = Extract Parallelism
  - Implicitly: Let hardware or compiler extract parallelism
    - Find instructions in the original sequential thread that can be executed at the same time
  - Explicitly: Change the programming paradigm and make programmer define parallel tasks

Reliability

- As transistors get smaller, the amount of physical charge that is stored to represent a ‘1’ or ‘0’ is decreasing
- Electromagnetic interference or even cosmic rays can cause the charge to be dissipated and the bit to be flipped
- How do we architect a system that accounts for unreliability



http://www.tsl.uu.se/radiation_testing/pictures/see.png