

Recap of last class

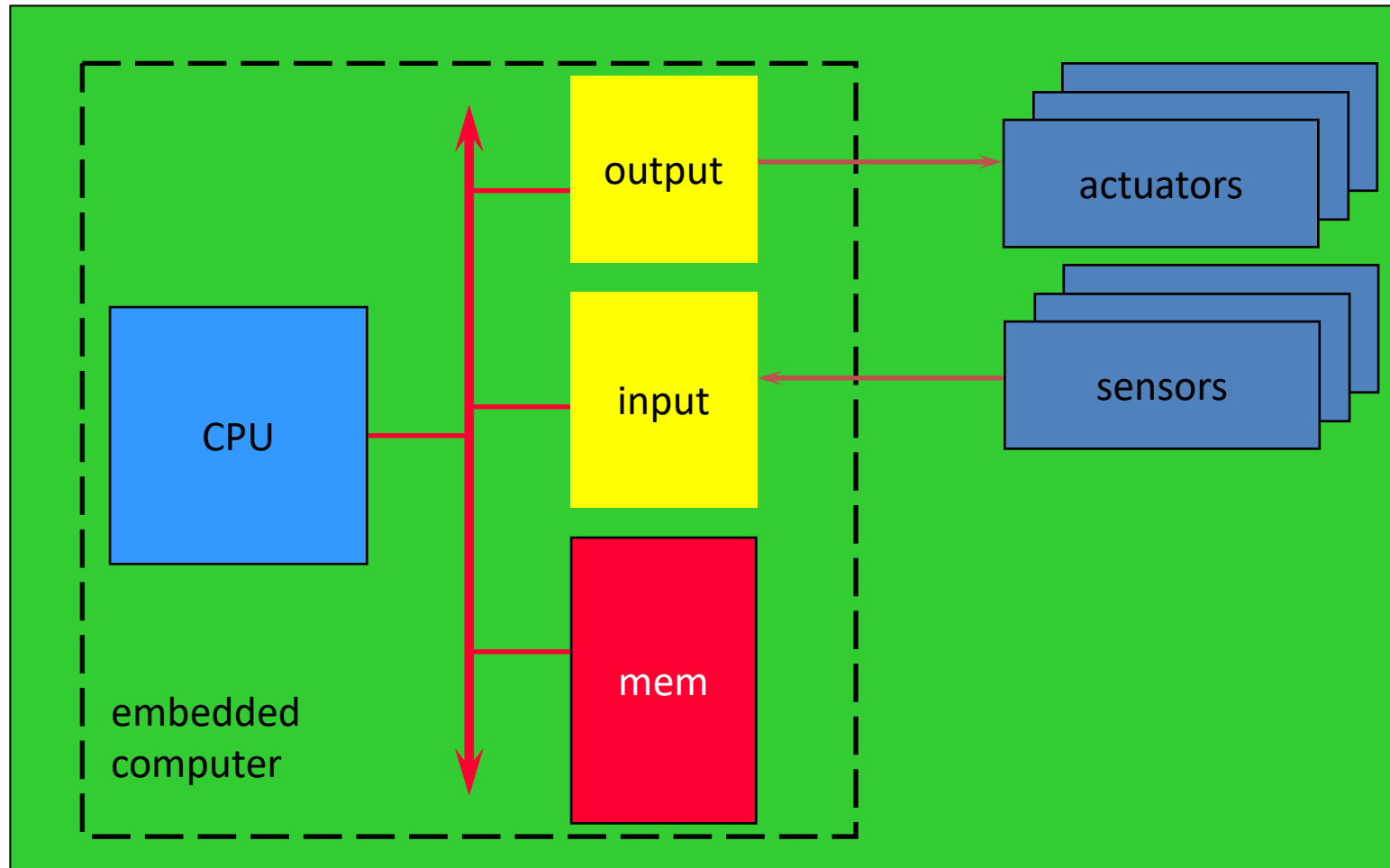
- Characteristics of embedded systems
 - Real-time, low power, performance constraints
- Why use a microprocessor
 - Alternatives: ASIC, microprocessor, FPGA
 - Microprocessor: reprogramability, performance/power ratio
- Design methodology
 - Top-down vs. bottom-up
 - Requirement, specification, architecture, component design, system integration

ECE 1175
Embedded Systems Design

Embedded System Architecture

Wei Gao

Embedding A Computer



Microprocessor Architecture

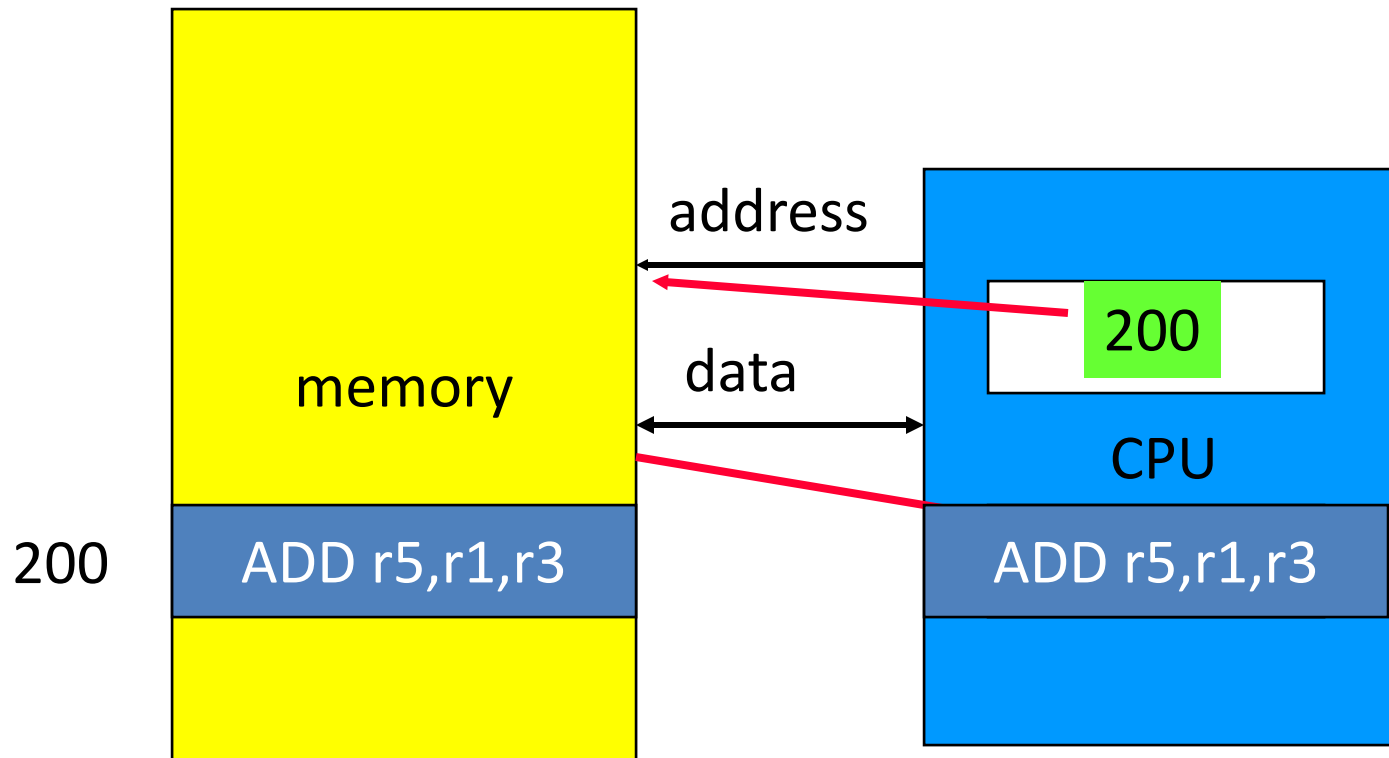
- von Neumann vs. Harvard
- CISC vs. RISC
 - **Complex** vs. **Reduced** Instruction Set Computer
- Two opposite examples
 - Super Harvard Architecture Single-Chip Computer (SHARC)
 - ARM7

Von Neumann Architecture

- **Feature:** memory holds data and instructions
 - Memory can be either read or written
- Central processing unit (CPU) fetches instructions from memory.
 - Separate CPU and memory distinguishes programmable computer.
- CPU registers help out:
 - Program counter (PC),
 - Instruction register (IR),
 - General-purpose registers, etc.

```
data segment
    dw 8 dup (0)
data ends
code segment
start: mov AX, data
        mov SS, AX
        mov SP, 16
```

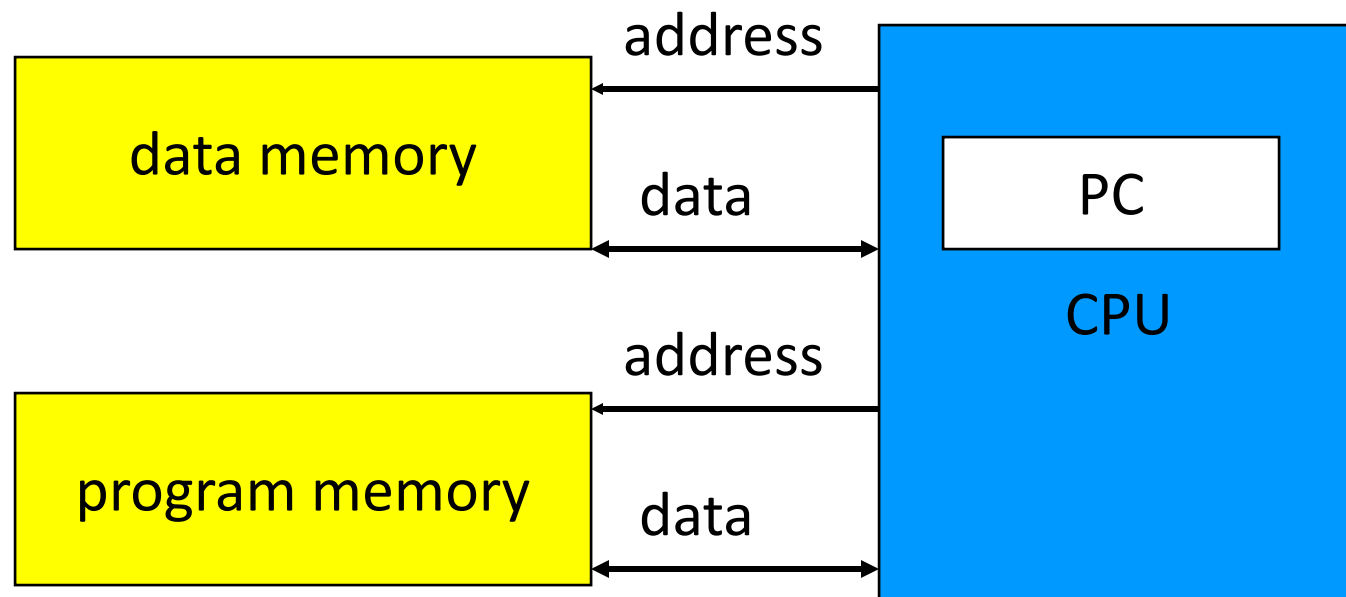
CPU + Memory in von Neumann



Von Neumann vs. Harvard

- von Neumann
 - **Same** memory holds **both** data and instructions.
 - A single set of address/data buses between CPU and memory
- Harvard
 - **Separate** memories for data and instructions.
 - Two sets of address/data buses between CPU and memory

Harvard Architecture



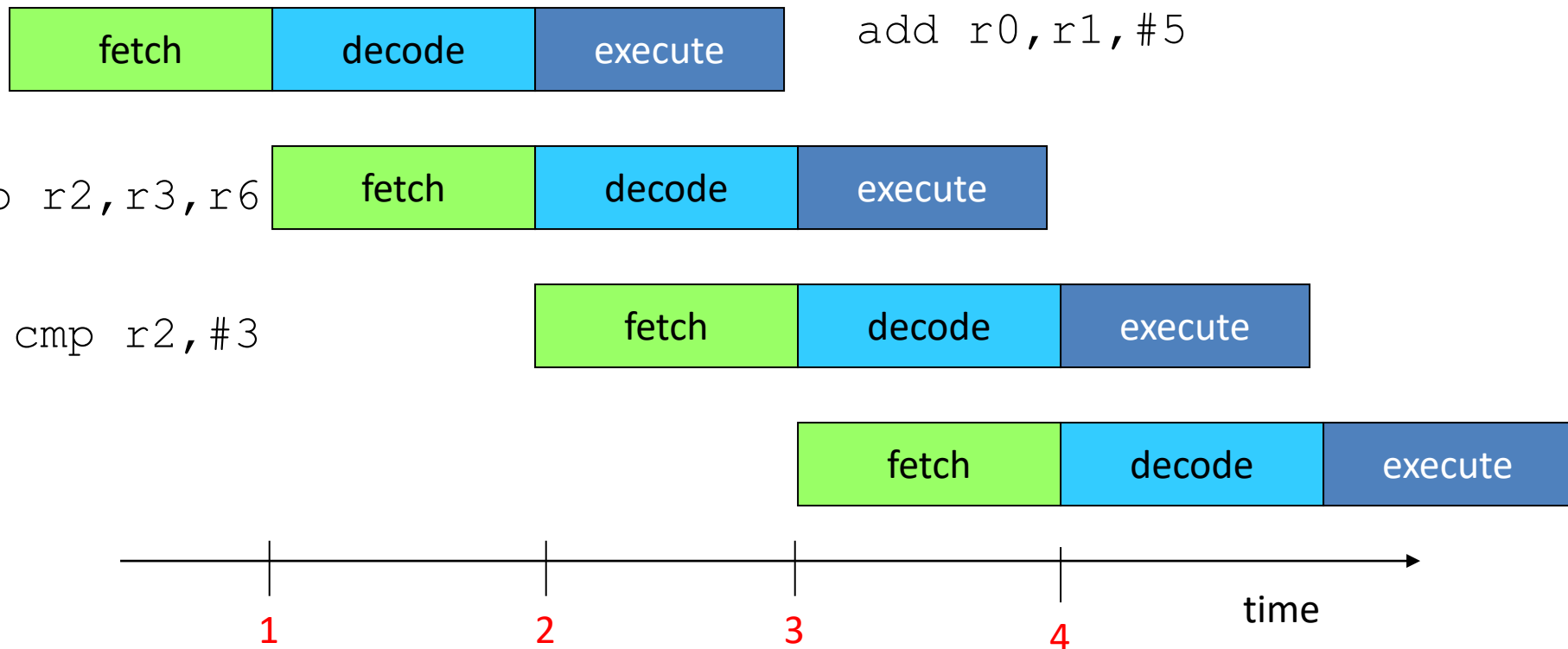
Von Neumann vs. Harvard

- Harvard makes it harder to write
 - Self-modifying code (data values used as instructions)
 - Less reprogrammable
- Harvard allows two simultaneous memory fetches.
- Harvard architectures are widely used because
 - Most DSPs use Harvard for streaming data
 - The separation of program and data memories → greater memory bandwidth → higher performance for digital signal processing
 - Speed is gained at the expense of more complex electrical circuitry.
 - Other examples: On chip cache of CPUs is divided into an instruction cache and a data cache

RISC vs. CISC

- Complex Instruction Set Computer (CISC)
 - Used in early computer architectures
 - Many addressing modes and instructions
 - High code density
 - Often require manual optimization of assembly code for embedded systems
- Reduced Instruction Set Computer (RISC)
 - Compact, uniform instructions → facilitate pipelining
 - More lines of code → poor memory footprint
 - Allow effective compiler optimization

ARM Pipeline Execution



Multiple implementations

- Successful architectures have several implementations:
 - Varying CPU clock speeds
 - different bus widths
 - different cache sizes
 - etc.
- The key performance bottleneck
 - Speed difference between CPU and memory
 - Solutions
 - Higher bus bandwidth
 - Use CPU cache: multiple levels

Microprocessors

RISC	ARM7 MSP430	ARM9
CISC	Pentium	DSP (SHARC)
	von Neumann	Harvard

Digital Signal Processor = Harvard + CISC

- Streaming data
 - need high data throughput
 - Harvard architecture
- Memory footprint
 - require high code density
 - Need CISC instead of RISC

DSP Optimizations

- Signal processing
 - Support **floating point** operation
 - Support **loops** (matrix, vector operations)
 - Finite Impulse Response (FIR) filters → multiply-accumulate in one cycle!
- Real-time requirements
 - Execution time must be predictable → opportunistic optimization in general purpose processor may not work (e.g., caching, branch prediction)

SHARC Architecture

- Modified Harvard architecture.
 - Separate data/code memories.
 - Program memory can be used to store data.
 - Two pieces of data can be loaded in parallel
- Support for signal processing
 - Powerful floating point operations
 - Efficient loop
 - Parallel instructions

SHARC

- CISC + Harvard architecture
- Computation
 - Floating point operations
 - Hardware multiplier
 - Parallel operations
- Memory Access
 - Parallel load/store
 - Circular buffer
- Zero-overhead and nested loop

ARM7

- von Neumann + RISC
 - Used as chip in Nokia 6110
- Compact, uniform instruction set
 - 32 bit or 12 bit
 - Usually one instruction/cycle
 - Poor code density
 - No parallel operations
- Memory access
 - No parallel access
 - No direct addressing

Sample Prices

- ARM7: \$14.54
- SHARC: \$51.46 - \$612.74

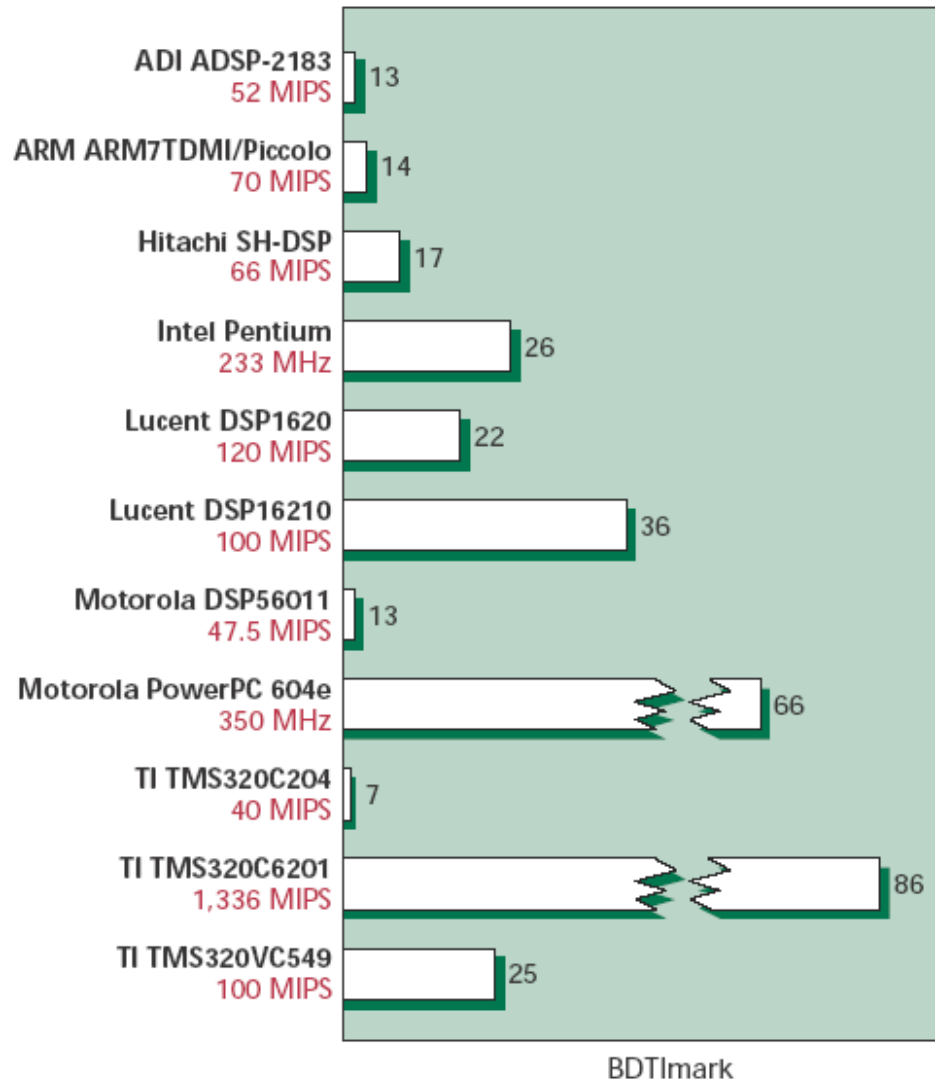
MIPS/FLOPS: Poor Metrics for DSPs

- MIPS: Million Instructions Per Second
- FLOPS: FLoating point Operations Per Second
- They do not indicate **how much work is accomplished by each instruction**, which depends on the architecture and instruction set.

Evaluating DSP Speed

- Implement and compare complete application on each DSP
 - Time consuming: implement and (manually) **optimize** a complete application on multiple DSPs
- Benchmarks: a set of small pieces (**kernel**) of representative code
 - Ex. FIR filter
 - Inherent to most embedded systems
 - Small enough to allow manual optimization on multiple DSPs
- Application profile + benchmark testing
 - Assign relative importance of each kernel

Benchmarks



The figure is from a paper in the reading list:

J. Eyre and J. Bier, [DSP Processors Hit the Mainstream](#), IEEE Micro.

Other Important Metrics

- Power consumption
- Cost
- Code density
-

Summary

- Von Neumann vs. Harvard
 - Harvard has separate memories for data and instructions
 - Higher performance for digital signal processing
- RISC vs. CISC
 - CISC: high code density, suitable for DSP
 - RISC: compact instructions, pipelining but poor memory footprint
- Two opposite examples
 - SHARC
 - DSP, CISC + modified Harvard
 - ARM7
 - RISC + von Neumann

Reading

- Optional: J. Eyre and J. Bier, [DSP Processors Hit the Mainstream](#), IEEE Micro, August 1998.
- Optional: More about SHARC
<http://www.analog.com/processors/processors/sharc/>