CENG-336
Introduction to Embedded Systems Development

An Introduction to Embedded Systems

Spring 2018

What is this course about?

CENG 336 Introduction to Embedded Systems Development (2-2) 3

Assembly language and controller architecture. Peripheral interfaces: A/D and D/A conversion, parallel and serial ports, interrupts and timers/counters. I/O bus architectures. Sensors and actuators. Design and analysis techniques. Real time operating systems.

Prerequisite: CENG 232.

http://catalog.metu.edu.tr/

Course Logistics

- Textbook
  - No Textbook
- Reference Material
  - PIC 18F8722 Data Sheets
  - Design with PIC Microcontrollers, John B. Peatman, Prentice Hall,1998.
  - Fundamentals of Microcontrollers … (PIC18), Ramesh s. Gaonkar, Thomson, 2007.
- Course conduct/communication:
  - https://odtuclass.metu.edu.tr/

Course Logistics

- Schedule:
  - Two lecture hours every week
    - Sec01: Thu 14:40-16:30 BMB-3 (if necessary extra Tue 15:40-16:30 BMB-2)
    - Sec02: Wed 15:40-17:30 BMB-1 (if necessary extra Fri 9:40-10:30 BMB-1)
  - Recitations are on Wednesdays 17:40-19:30 BMB-1, all sections combined
- Grading:
  - Midterm: 18%, Take Home Exams: 35%, Board Test 2%, Attendance %5, Lan Review Exam 15%, Final: 25%
  - Lab review exam: 15%.
    - Towards the end of the semester (May 5), 3-4 hour exam in CENG labs
    - You will need to implement and demonstrate a PIC program, *by yourself*

Course Logistics

- Minimum requirements for taking the final
  - At least 30/100 on the *midterm*
  - At least 30/100 on THEs (1-2-3)
  - At least 30/100 the *lab review exam*
  - Complying with all the lab requirements.
  - In the case of academic misconduct (cheating, plagiarism or any other form) for any part of the course requirements (examinations, assignments, etc.), the student will not be allowed to take the final examination of the course.

Outline

- Embedded systems, Applications, Why are they "special"?, General Structure, Examples
- Characteristics of embedded systems, Software issues, Embedded design lifecycle
- Processors: technology, architecture, Microprocessor vs Microcontroller, Common embedded microcontrollers
- Example: Automotive embedded systems

Embedded systems

- What exactly is an "Embedded System"?
  - Any device, or collection of devices, that contain one or more **dedicated** computers, microprocessors, or microcontrollers
- Where can I find one?
  - Everywhere!








home appliances

printers&copiers

toy industry

robotics

aircrafts

automotive

factory coordination

personal gadgetry

Embedded systems: Applications

- Consumer segment, e.g. cameras, camcorders, VCRs, washers, microwave ovens, …
- Automobiles, e.g., engine control, anti-lock brake, air bags, …
- Office automation, e.g., copiers, printers, FAX machines, …
- Telecommunications, e.g.,, cellular phones, PDAs, interactive game boxes, answering machines, …
- Other industrial products, e.g., door locks in hotel rooms, automatic faucets, …

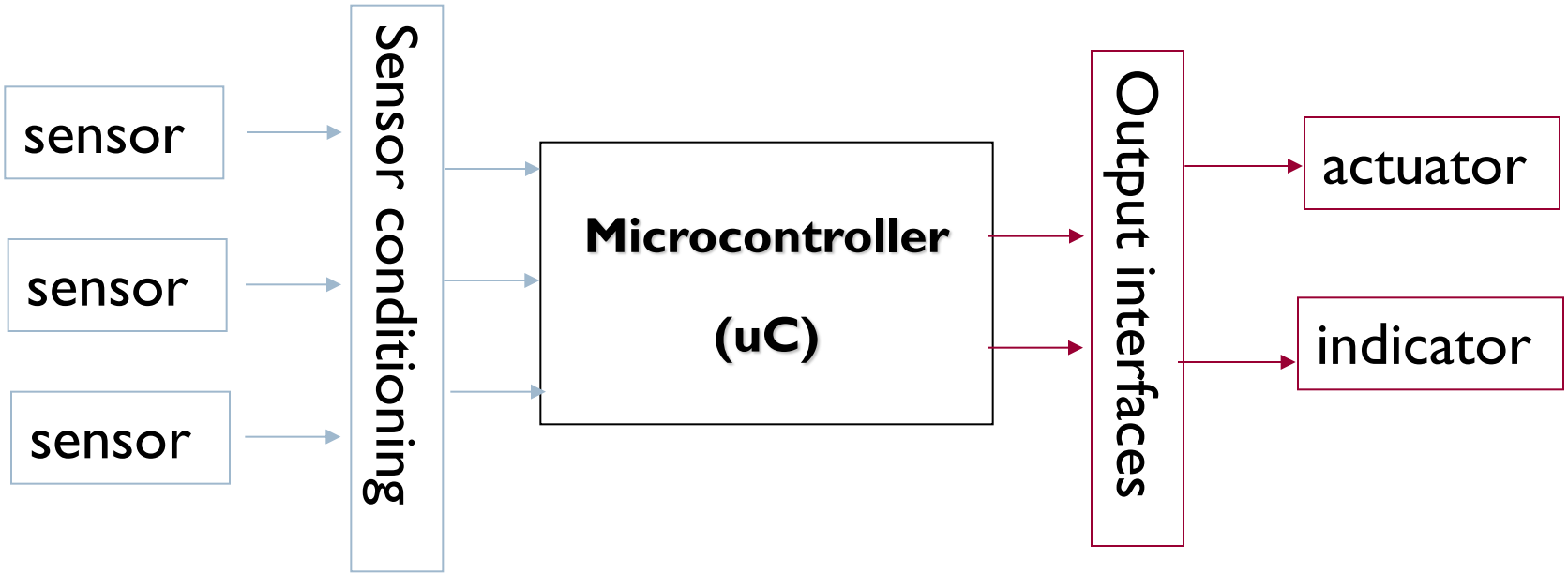
Why are they "special"?

- Embedded computing devices have *rigidly defined operational bounds*.
  - Not general purpose computers (PC, Unix workstation)
  - How about a PDA? GPS? Cell phone? PC in an industrial robot?
- Entirely different set of constraints
  - cost-sensitive because of market considerations (toys, appliances)
  - real-time constraints for safety critical tasks (flight control, ABS)
  - limited computation and memory resources because of size, weight power and cost constraints
  - Must be very robust to operate in harsh conditions for long time

Very constrained codesign of hardware and software.

Not something you have learned about so far.

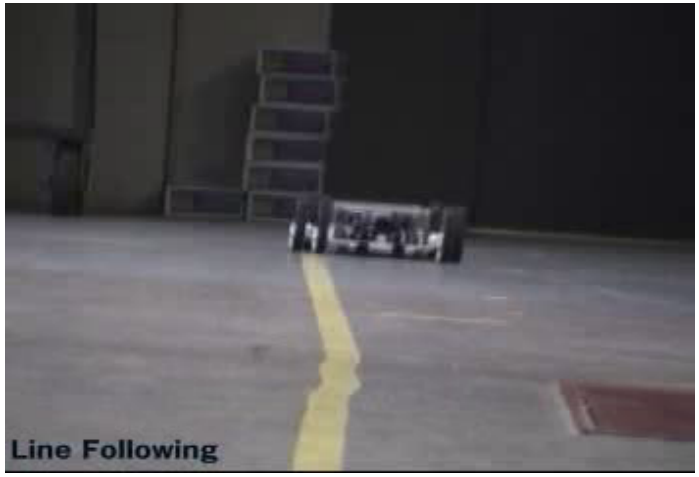
Embedded system: General structure



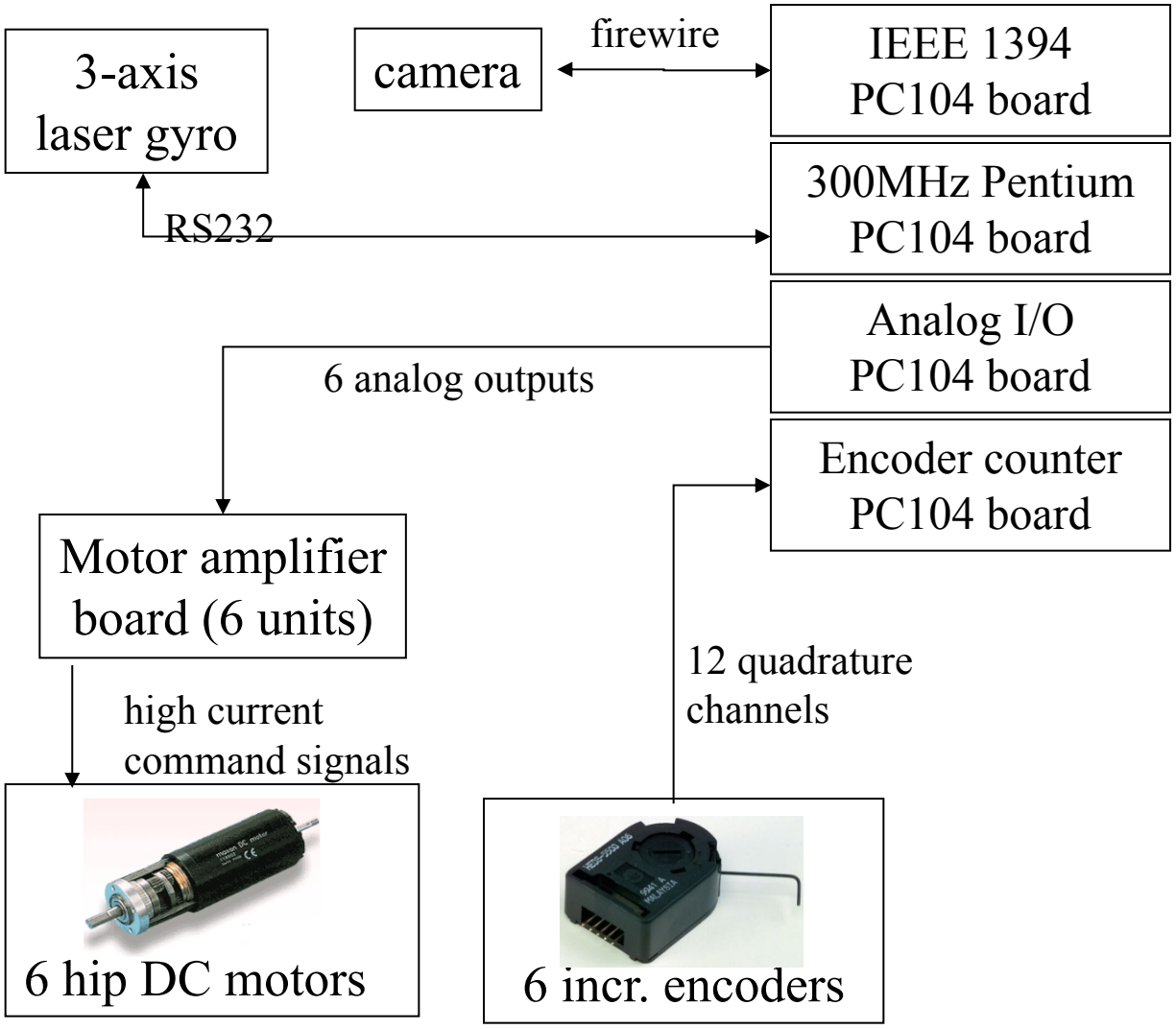
- Intense interaction with the external world
- Sensors convert physical quantities into electrical signals
- Microcontroller acquires data, processes it and issues outputs
- Actuators convert output signals into physical work

Example: A hexapod robot

- RHex: Relatively simple, autonomous legged robot



RHex: System design (simplified)



Software tasks:

- **Periodically** send appropriate motor commands
- **Periodically** read encoder sensors and computer motor position and speed
- **Compute** motor commands based on "desired" position
- **Periodically** read gyro (angular velocity) and compute body position
- **Wait** for camera to be ready and read image
- **Process** image and determine which direction to go

Outline

- Embedded systems, Applications, Why are they "special"?, General Structure, Examples
- Characteristics of embedded systems, Software issues, Embedded design lifecycle
- Processors: technology, architecture, Microprocessor vs Microcontroller, Common embedded microcontrollers
- Example: Automotive embedded systems

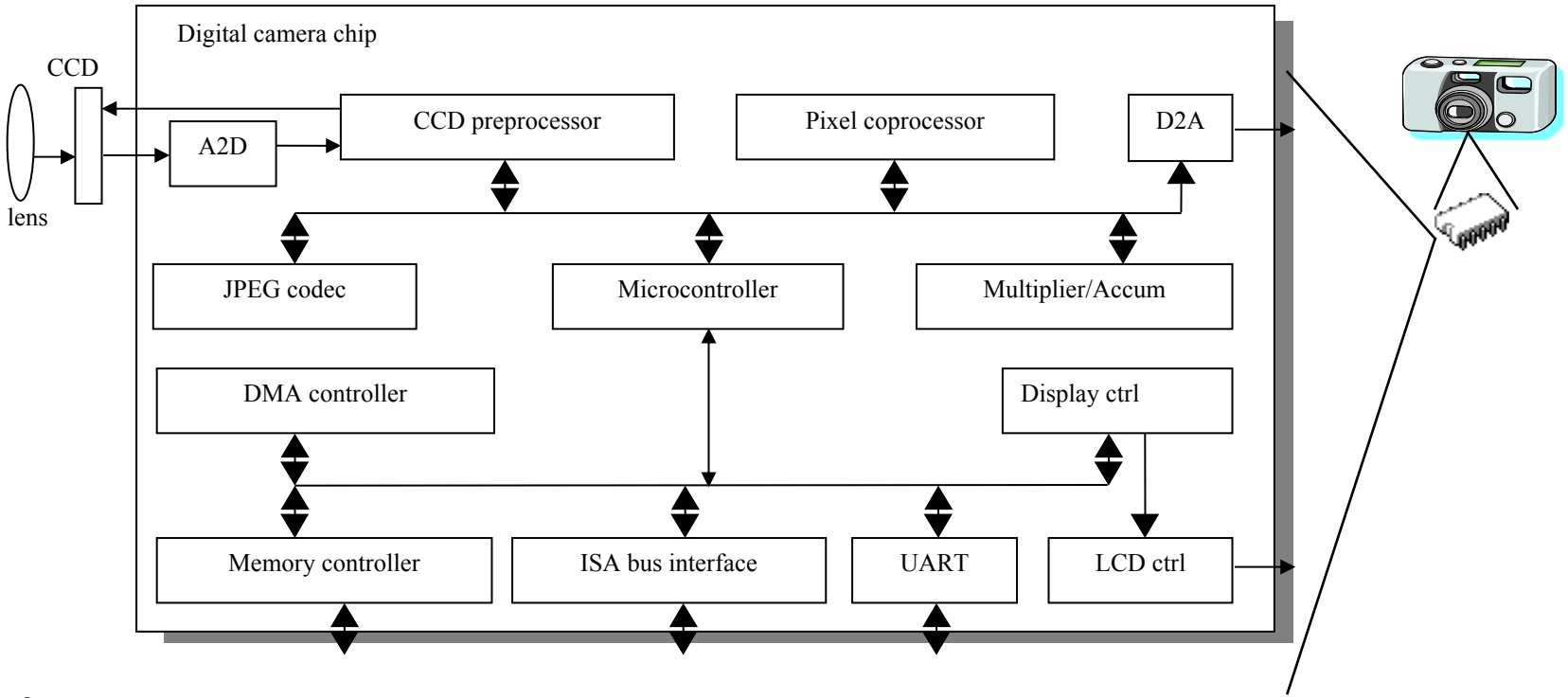
Characteristics of embedded systems

- Real-time constraints
  - Sensor and actuator accesses are *time-sensitive*
  - Exact time of image acquisition and angular velocity readings must be known for proper interpretation
  - Main cause: Physical events do not wait for software to catch up!
- Cost-sensitivity
  - Products will often be mass-produced for a particular task
  - Cost, power requirements and size of general-purpose computing platforms is prohibitive. Only necessary components are used
- High cost of failure, need for fast self-recovery
  - Embedded systems should not need baby-sitting
  - Failure can be life-threatening for some systems: cars, planes, medical equipment, nuclear reactors etc.

Characteristics of embedded systems

- Power limitations and constraints
  - Consider a device to track wild animal movements in nature
  - Low power = less heat = fewer failures
  - Low power = less energy = longer time of operation
  - Low power = smaller batteries = physically lighter and smaller
  - Usually requires both hardware and software elements to achieve
- Must operate in extreme environmental conditions
  - Embedded systems are everywhere and this means ***everywhere***
  - Your PC would fail in a second if left outside on top of Everest
  - Once again, both hardware and software should be aware of this
- Far fewer resources than a general purpose computer
  - lower speed and memory, limited inputs and I/O etc.

An embedded system example – Digital Cam.

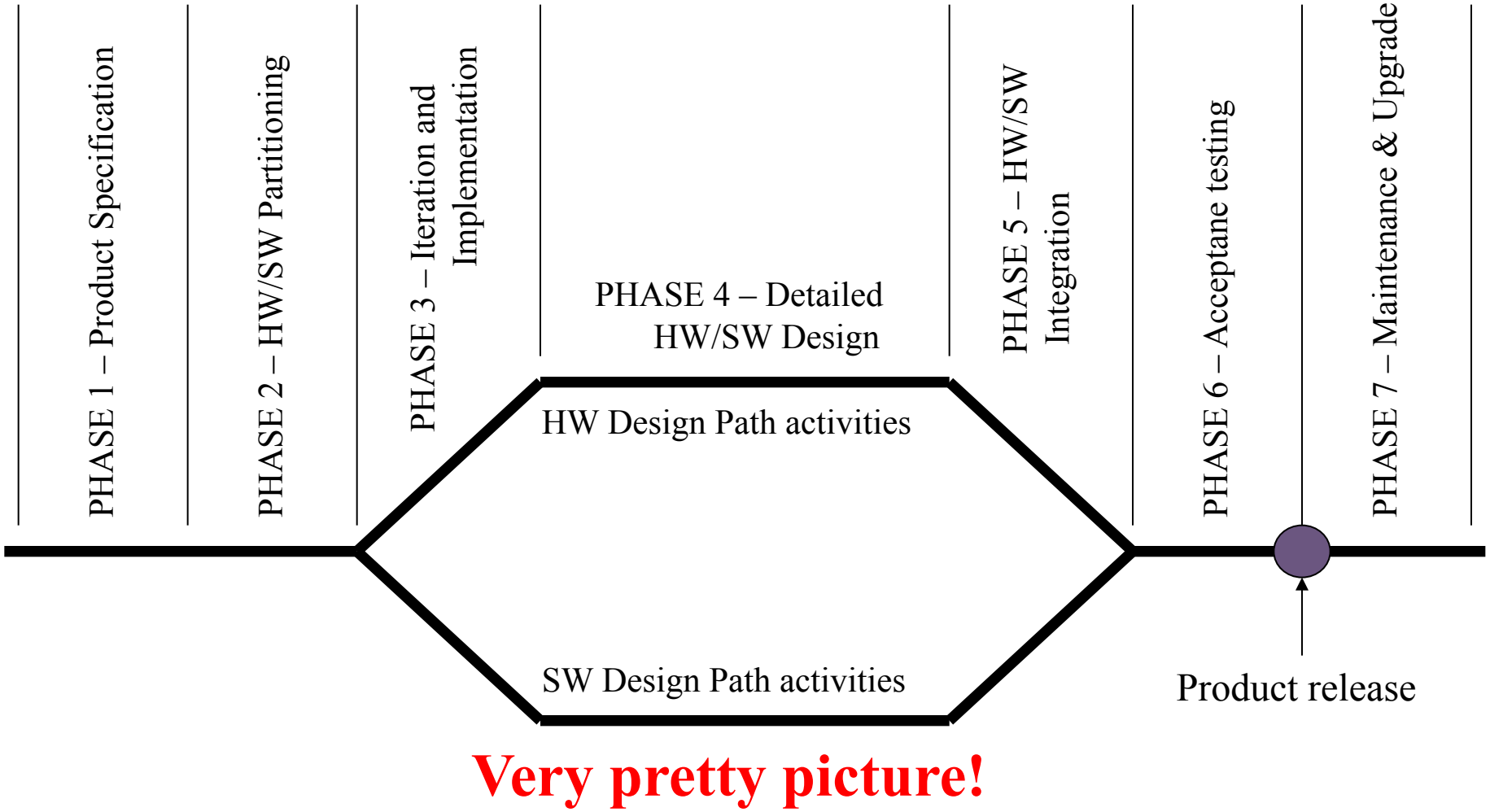


- Single-functioned -- always a digital camera
- Tightly-constrained -- Low cost, low power, small, fast
- Reactive and real-time -- only to a small extent

Software issues

- Specialized development tools
  - You will surely not be able to use printf() or popup alerts
  - Development takes places on a *host*, separate than the *target*
  - Final product usually has ROM and a very small chunk of RAM
- Need for a Real-Time Operating System(RTOS)
  - Regular Windows or Linux not suitable for time critical tasks
  - Programming for guaranteed timeliness is hard. Deadlocks, interrupt processing, polling for external events etc. must be properly resolved and synchronized
- Impossible to avoid codesign of software with hardware

Embedded design lifecycle



Very pretty picture!

Specification and partitioning

- Product specification
  - Embedded *products* are *dedicated* systems, not general purpose
  - Minimal specification to capture "desired" functionality
- Hardware/Software partitioning
  - Think winmodems vs. hardware capable modems
  - Dedicated encoder counter chips vs. software counting
  - Hardware rendering vs. software shading for graphics
  - Hardware is faster but less flexible and more difficult to debug
  - Software/firmware can be updated but runs slower and consumes computational resources
  - You will have to balance conflicting requirements and limitations. No general methods, just guidelines

HW/SW design and integration

- Development of HW/SW
  - You have been learning about parts of this so far (except difficulties particular to embedded systems)
- Integration
  - Can be quite a nightmare if not thought out beforehand
  - Software engineers usually have some tools beforehand: evaluation boards, instruction set simulators etc.
  - Hardware designers have other tools: logic analyzers, in-circuit emulators etc.
  - Unlikely to work right away: system working at full speed will quickly reveal many inconsistencies and bad interactions
  - Successful debugging requires special tools. No printf()'s or simple debuggers will be available

Testing and debugging

- Hardware failures can cause software failure and vice-versa. For example
  - Intermittent power connection causing resets
  - Overdriving motors from software causing burnouts
  - Multiple channels connected together mistakenly configured as outputs
- Compliance testing may be necessary
  - RF suppression for compliance with specific requirements
  - Meeting robustness requirements (operational range of temperatures etc.)

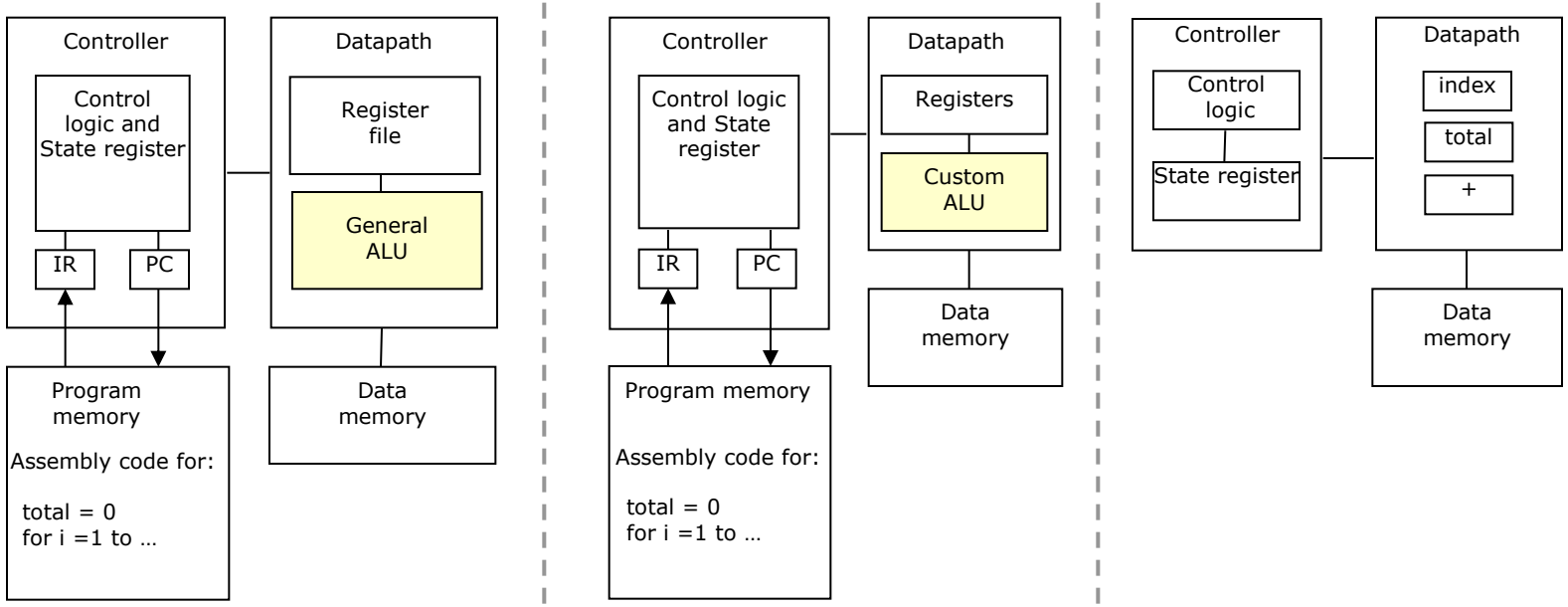
Summary: Embedded system development is not just "software on small machines"

Outline

- Embedded systems, Applications, Why are they "special"?, General Structure, Examples
- Characteristics of embedded systems, Software issues, Embedded design lifecycle
- Processors: technology, architecture, Microprocessor vs Microcontroller, Common embedded microcontrollers
- Example: Automotive embedded systems

Processor technology

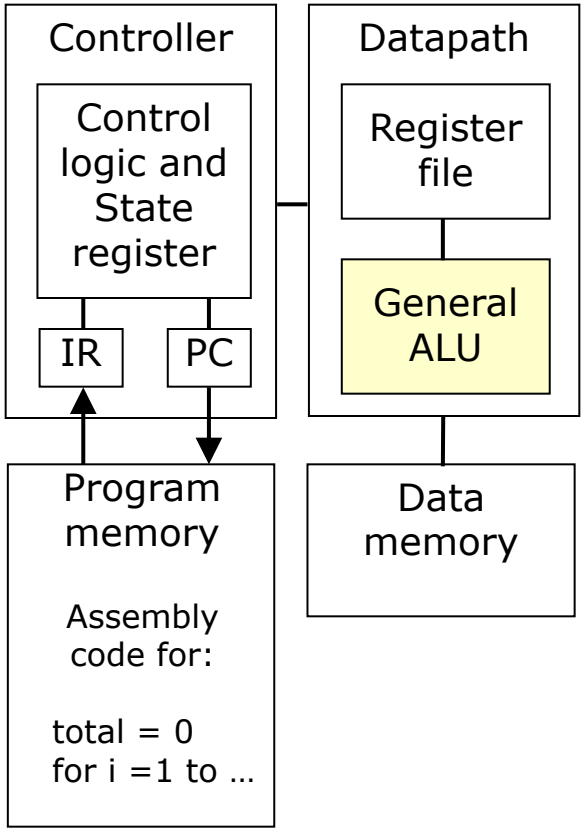
- The architecture of the computation engine used to implement a system's desired functionality
- Processor does not have to be programmable
  - "Processor" is *not* equal to general-purpose processor



General-purpose ("software") **Application-specific** **Single-purpose** ("hardware")

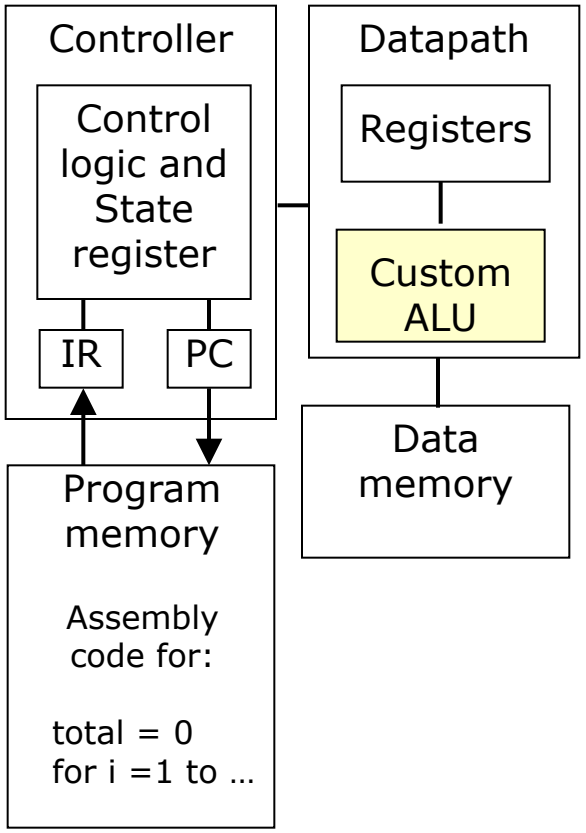
General-purpose processors

- Programmable device used in a variety of applications
  - Also known as "microprocessor"
- Features
  - Program memory
  - General datapath with large register file and general ALU
- User benefits
  - Low time-to-market and NRE costs
  - High flexibility
- "Pentium" is the most well-known, but there are hundreds of others



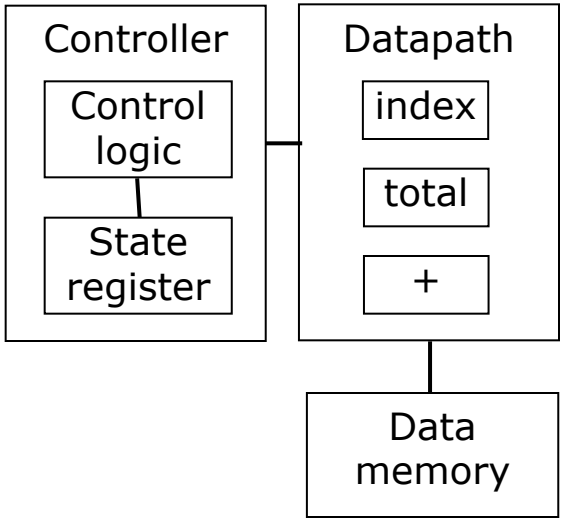
Application-specific processors

- Programmable processor optimized for a particular class of applications having common characteristics
  - Compromise between general-purpose and single-purpose processors
- Features
  - Program memory
  - Optimized datapath
  - Special functional units
- Benefits
  - Some flexibility, good performance, size and power



Single-purpose processors

- Digital circuit designed to execute exactly one program/task
  - a.k.a. coprocessor, accelerator or peripheral
- Features
  - Contains only the components needed to execute a single program
  - No program memory
- Benefits
  - Fast
  - Low power
  - Small size



Choices for processor architecture

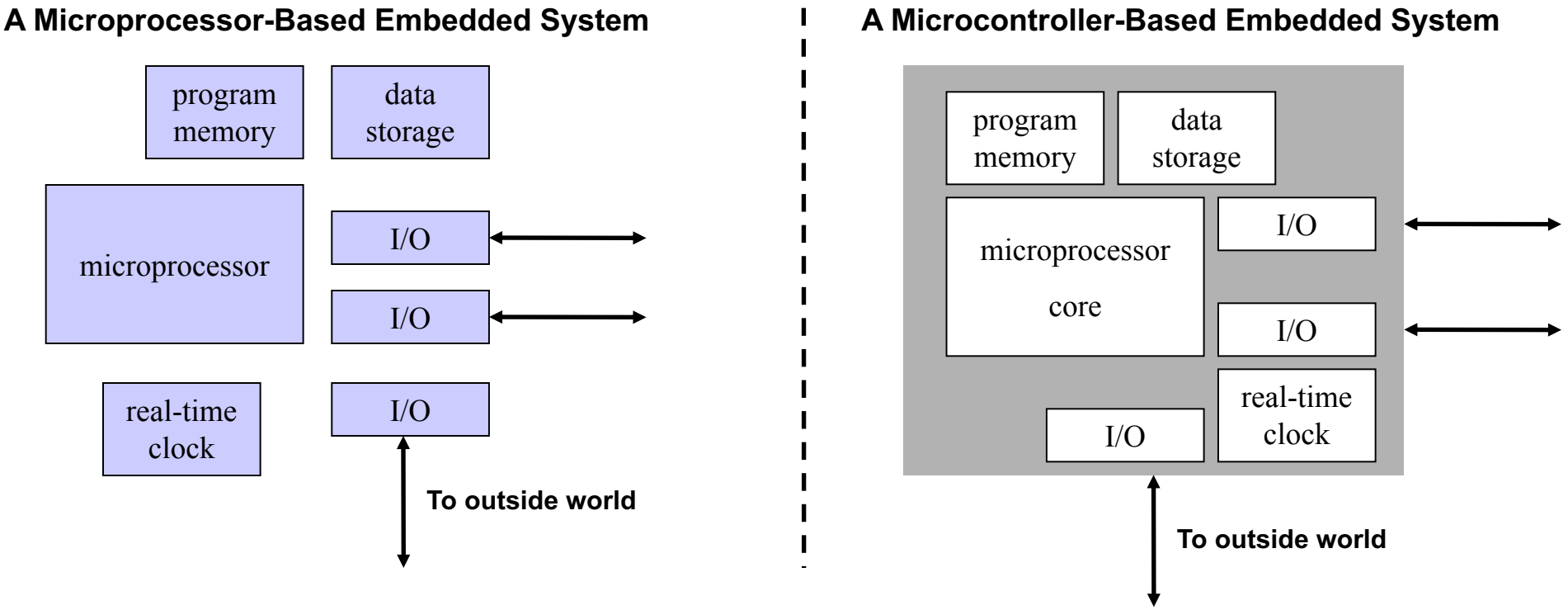
- CISC - Complex Instruction Set Computer
  - Many instructions which can perform involved operations: compact code
  - Can be many clock cycles per instruction
  - Large silicon area > Higher cost per die
- RISC - Reduced Instruction Set Computer
  - More modern architecture
  - One instruction executed per clock cycle > Very fast
- DSP - Digital Signal Processor
  - Specialized type of uP
  - Designed for real time mathematical manipulation of data streams
    - Radar image processing, audio/voice processing, ultrasound and photographic image processing
  - Includes instructions designed for multiplication and accumulation

Choice can be one of the three or a combination

Some definitions

- **Microprocessor**
  - An integrated circuit which forms the central processing unit for a computer or embedded controller, but requires additional support circuitry to function
  - MC68000, 80486, Pentium, K6, etc.
- **Microcontroller**
  - A microprocessor plus additional peripheral support devices integrated into a *single package*
  - Peripheral support devices may include:
    - Serial ports (COM), Parallel (Ports), Ethernet ports, A/D & D/A
    - Interval timers, watchdog timers, event counter/timers, real time clock Other local processors (DSP, numeric coprocessor, peripheral controller)
  - PIC, MPC555 are microcontrollers

Microprocessor vs Microcontroller



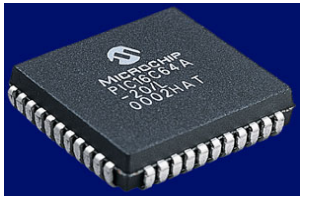
- Microcontroller advantages
  - lower cost, more reliable, better performance, faster and lower RF signature
  - may be less flexible for research and development projects

World-wide figures

- Over 100 Billion μCs used in embedded systems
- Average embedded system has 4-6 μC, probably more by now
- Of all μC sold, 90% go into "non-computers", 10% in "computers"
- You will most likely end up working with a "non-computer"at some point in your career

Common embedded microcontrollers

- **4-bit Microcontrollers: PIC ($1.79)**
- **8-bit Microprocessors and microcontrollers**
  - **Zilog: Z80 families ($1.39)**
  - **Intel: 8042, 8048, 8051 families ($4.95)**
  - **Motorola: 6805, 68HC11 families ($8.00)**
- **16-bit Microprocessors and microcontrollers**
  - **Intel, AMD: 80186 families**
  - **Motorola: 68300 families**
  - **NEC, Hitachi, Phillips**
- **32-bit Microprocessors and microcontrollers**
  - **Intel 80386, 80486, Pentium, PII, PIII,PIV, StrongARM, X-scale**
  - **ARM: ARM7TDMI, ARM9**
  - **AMD 486E, SC520 (Aspen)**
  - **Motorola 680X0, ColdFIRE, PowerPC, MPC5xx**
  - **Intel X-Scale**
- **64-bit Microprocessors and microcontrollers**
  - **MIPS family, Athlon 64**



Choice depends on a variety of performance and memory size considerations

Outline

- Embedded systems, Applications, Why are they "special"?, General Structure, Examples
- Characteristics of embedded systems, Software issues, Embedded design lifecycle
- Processors: technology, architecture, Microprocessor vs Microcontroller, Common embedded microcontrollers
- Example: Automotive embedded systems

Example: Automotive embedded systems

- Today's high-end automobile may have more than 100 microprocessors:
  - 4-bit microcontroller checks seat belt;
  - microcontrollers run dashboard devices;
  - 16/32-bit microprocessor controls engine.
- Example: BMW 745i
  - 2,000,000 LOC, Windows CE OS
  - 53 x 8-bit µP
  - 11 x 32-bit µP
  - 7 x 16-bit µP
  - Multiple Networks
  - Buggy!

Outline

- Embedded systems, Applications, Why are they “special”?, General Structure, Examples
- Characteristics of embedded systems, Software issues, Embedded design lifecycle
- Processors: technology, architecture, Microprocessor vs Microcontroller, Common embedded microcontrollers
- Example: Automotive embedded systems