



UiO : **University of Oslo**

FYS3240- 4240

Data acquisition & control

# Introduction

**Spring 2019– Lecture #1**

Reading: RWI (Real World Instrumentation) Chapter 1.



# Topics

**Instrumentation:**

**Data acquisition and control**

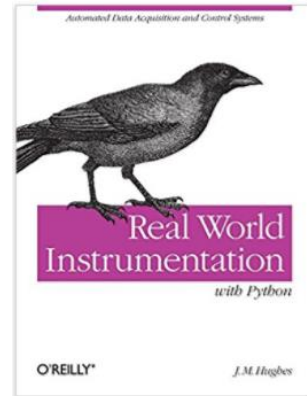
- **PC-based and embedded systems**

# Topics you will learn about

- Atmel XMEGA Microcontrollers (C-programming)
- LabVIEW programming
- Data acquisition (DAQ)
- Control algorithms and process control
- Real-Time and Embedded systems
- Time and Synchronization
- Data fusion and estimation
  
- Computer buses and interfaces
- Networked and distributed systems
- High-speed data streaming

# Curriculum

- **Real World Instrumentation with Python: Automated Data Acquisition and Control Systems** 1st Edition, ISBN-13: 978-0596809560
  - Selected chapters
- **Lecture notes**
- **Two “papers”**
  - Stochastic models, estimation and control; by Maybeck
  - An Introduction to the Kalman Filter; by Welch and Bishop
    - Section 1 and 3
- **Laboratory exercises**



Chapter 1, "Introduce to" from STOCHASTIC MODELS, ESTIMATION, AND CONTROL, Volume 1, by Peter S. Maybeck, copyright © 1979 by Academic Press, reproduced by permission of the publisher. All rights of reproduction in any form reserved.

**Stochastic models,  
estimation,  
and control**  
VOLUME 1

*PETER S. MAYBECK*  
DEPARTMENT OF ELECTRICAL ENGINEERING  
AIR FORCE INSTITUTE OF TECHNOLOGY  
WRIGHT-PATTERSON AIR FORCE BASE  
OHIO

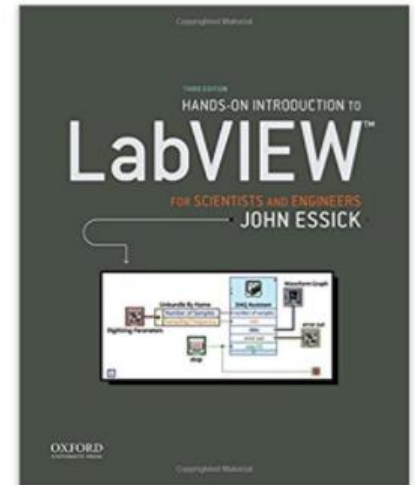
**An Introduction to the Kalman Filter**

by  
Greg Welch<sup>1</sup>  
and  
Gary Bishop<sup>2</sup>

TR 95-041  
Department of Computer Science  
University of North Carolina at Chapel Hill  
Chapel Hill, NC 27599-3175

# Recommended additional book

- **Hands-On Introduction to LabVIEW for Scientists and Engineers 3rd Edition, ISBN-13: 978-0190211899**
  - Chapter 1 to 5 for basic LabVIEW



# Admin information

# Course evaluation

- Midterm evaluation
  - Questionnaire
- After the Exam
  - Dialog meeting (lecturer, teaching assistants, student representative group)
  - Final evaluation
- Need 1 or 2 students for the student representative group (voluntary)

# Exam

- Usually written exam...
- **Written exam only in English!**
- But, you can of course answer in Norwegian!



# About the lectures and labs

- Lectures given in Norwegian (English on request)
- **2 lecture hours per week**
- **4 (bachelor) / 5 (master) lab assignments - room V442**
- The lab is “open” every day; note the building opening hours!
- Submit the **lab reports** including the source code at **devilry.ifi.uio.no**
- Three weeks assigned to each microcontroller lab.



## Mandag–fredag

Bygning:	Åpent uten adgangskort	For laveregrads-student: Adgangskort uten kode	For høyeregrads-student/ansatt: Adgangskort med kode
Øvrige bygninger	07.00–18.00	07.00–23.00	Hele døgnet (Unntak: <a href="#">Nattestengte bygninger</a> )

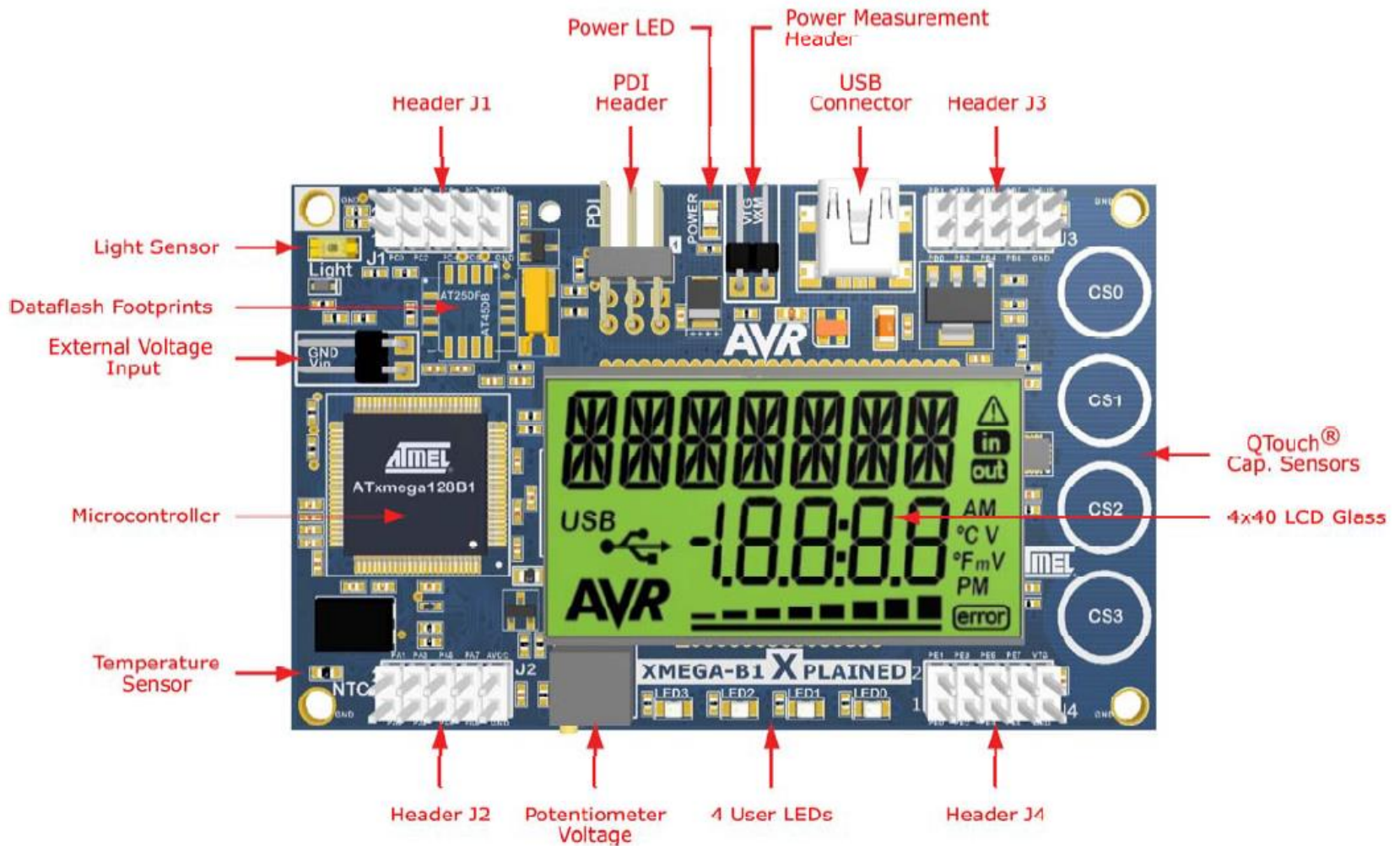
Saturday: 08 - 18 if access card without code  
 Sunday : only if you have access card with code

# Lab

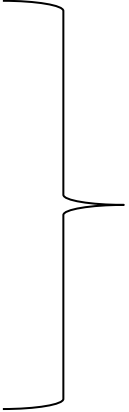
- Up to 20 lab groups
- 2 students for each group
  - 10 computers available
- **Sign up for a lab group (Monday or Tuesday)!**
  - The lab teaching assistant, [Tom Morten Berge](#), will be available from **10.00 to 15.00** on the two lab days

Group	Monday	Group	Tuesday
1	<etternavn, fornavn> <etternavn, fornavn>	11	<etternavn, fornavn> <etternavn, fornavn>
2		12	
3		13	

# Atmel XMEGA-B1 Xplained evaluation kit



# About the lab work

- **Lab 0: LabVIEW introduction with exercises**
    - Learn basic application development with LabVIEW
  - **Lab 1: Simple I/O programming**
    - LEDs and switches
  - **Lab 2: Control an LCD display**
  - **Lab 3: Control the ADC (analog to digital converter)**
- 
- Microcontroller labs
- **Lab 4: Project for Master students (FYS 4240)**
    - LabVIEW project - Distributed DAQ
      - **Sender** : Sample video and AI data, and transmit them using UDP (over Ethernet)
      - **Receiver**: Read UDP data, visualize data and save data to file (TDMS and AVI)

# Install LabVIEW on own PC

- Go to <https://www.winprog.uio.no/>
- Select **LabVIEW**

Winprog

Software for Windows at the University of Oslo

Norwegian website Jan K

Home

## Software available from the University of Oslo

- ArcGis
- EndNote
- f4transkript
- FastX
- LabVIEW
- Matlab
- Microsoft Office 2016
- Microsoft Visio 2013
- Minitab
- MinitabExpress
- MUSIT
- Nvivo
- OneDriveForBusiness
- Oracle
- Oxygen
- Reference Manager
- Solidworks
- SPSS
- Stata SE
- WebDrive
- x-win32



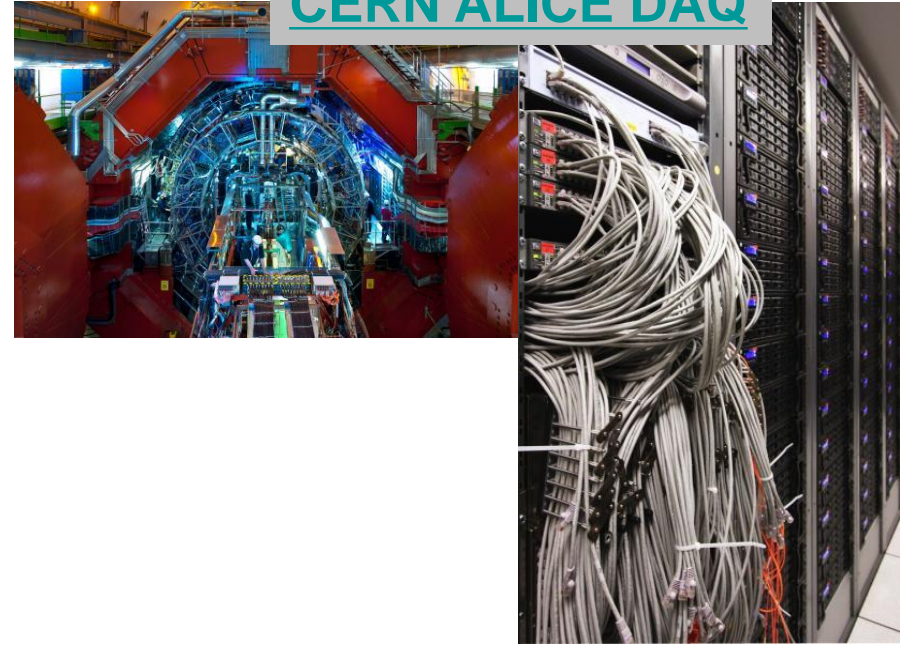
# Introduction to data acquisition & control

# Data acquisition and control - examples

Autonomous cars



CERN ALICE DAQ



Drones



CubeSat

# Unmanned Aircraft System (UAS) example

IMU: Inertial Measurement Unit

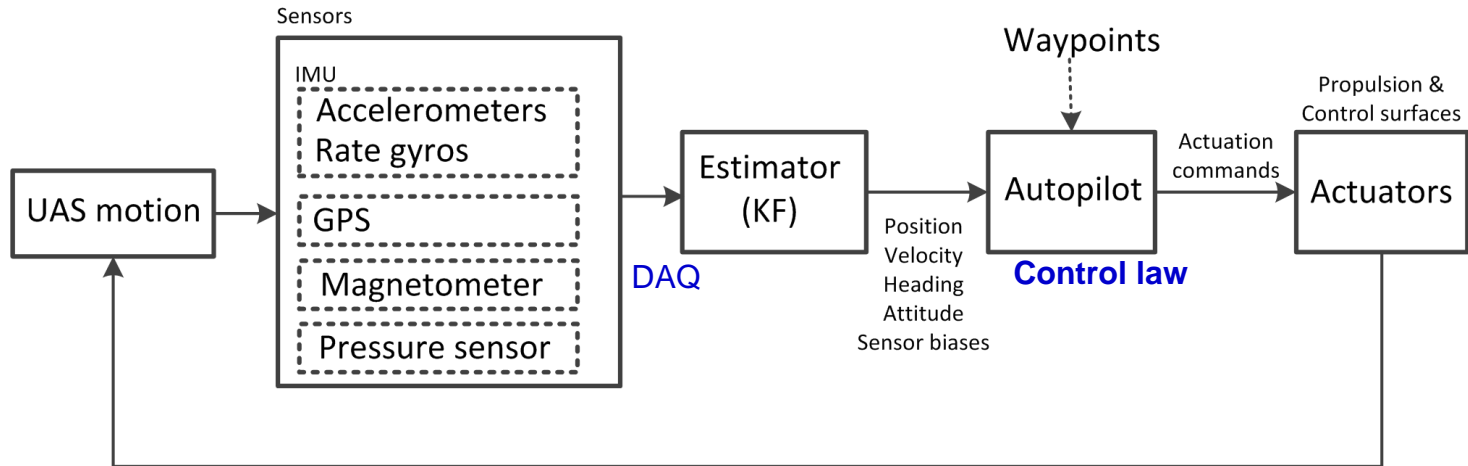
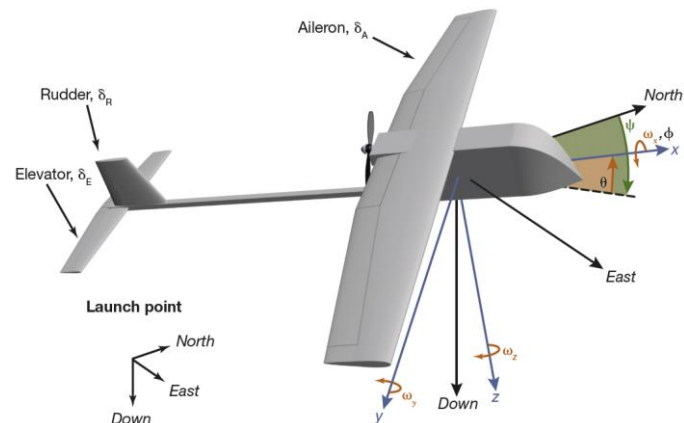
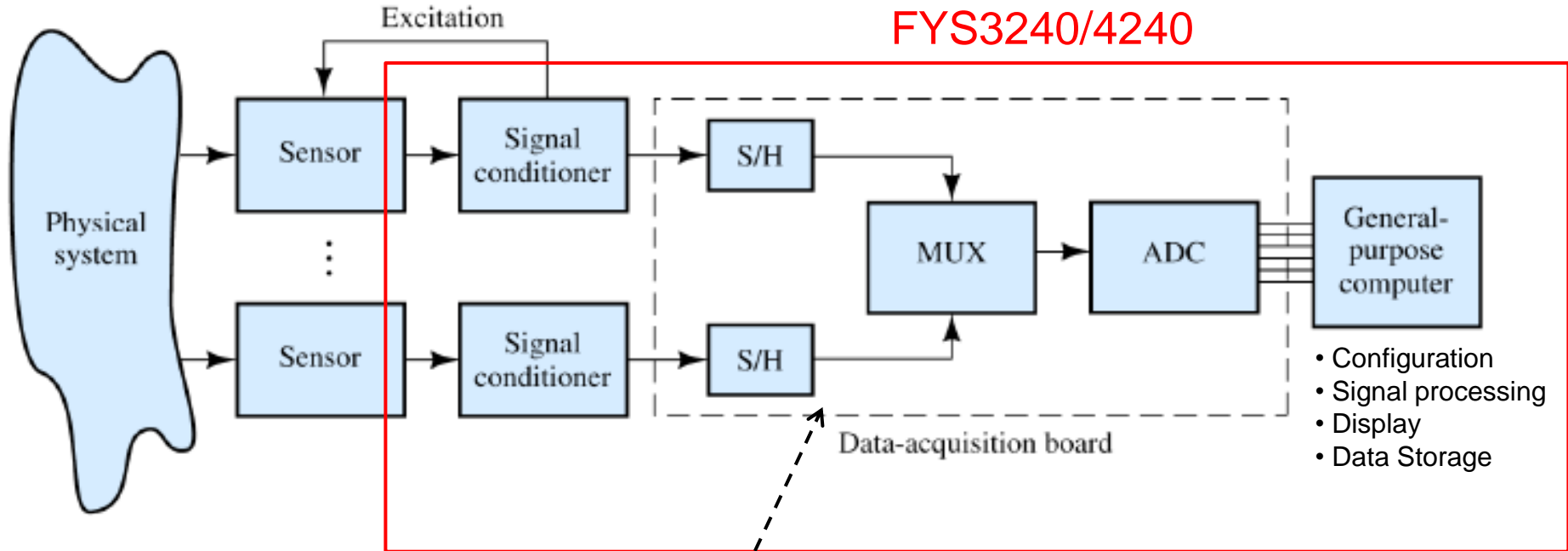


Illustration 5: Waypoints for flight over seattle





# Computer-based DAQ system

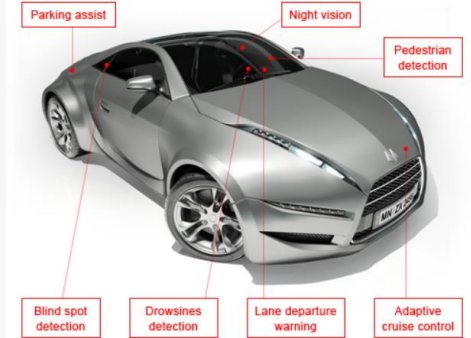
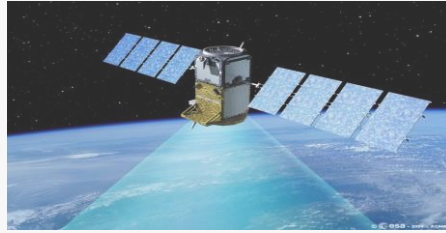


- PCI DAQ card
- USB DAQ box
- PXI DAQ system
- .....



# Embedded systems

## Aerospace & Defence



ABS Entertainment system



1968

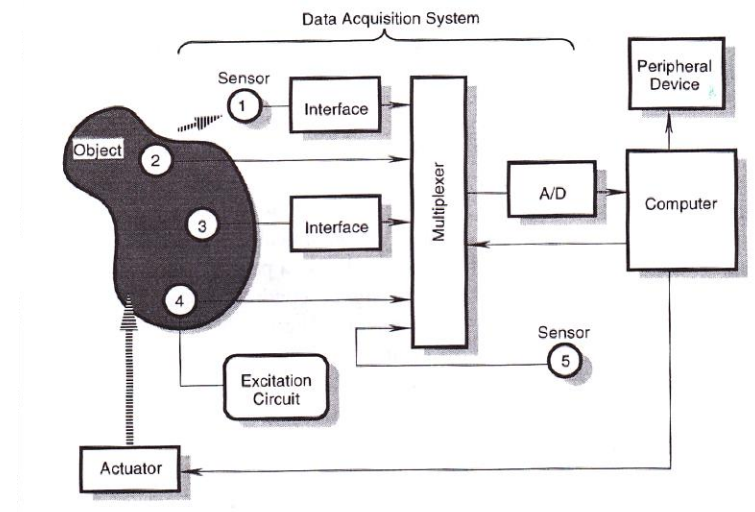


Apollo Guidance Computer

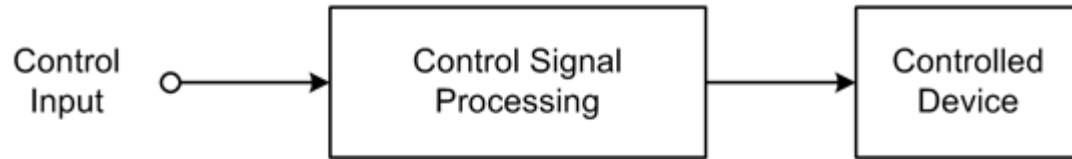
“The first embedded computer systems”

# Data Acquisition (DAQ) & Control Systems

- **The main application domain for Real-Time and Embedded computer systems is within Data Acquisition and Control.**
  - A dish washer system reads data from a timer, temperature and water level sensors, and controls the water valves and heater.
- **Data acquisition involves collecting signals from measurement sources and digitizing the signal for storage, analysis and presentation.**
  - For a closed-loop control system the acquired and processed data are also utilized for controlling the process through a feedback loop.



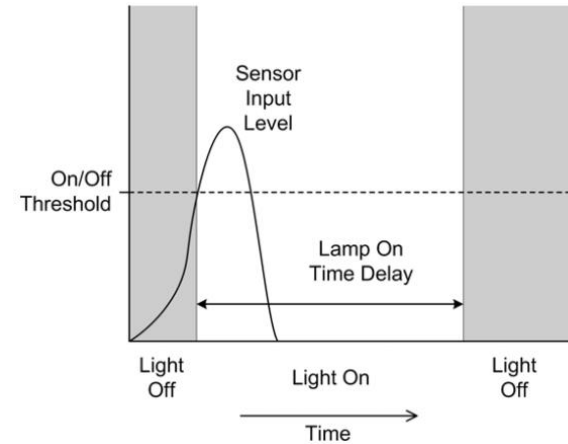
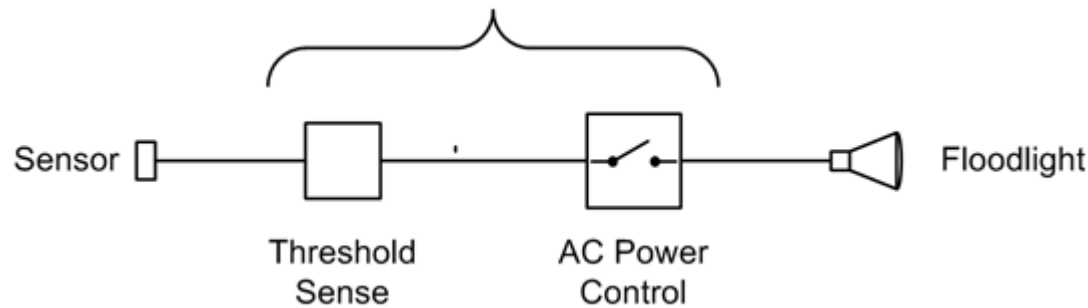
# Open-loop control



Possible Functions:

- Threshold (limit trip)
- Amplification
- Inversion
- Filtering
- Time Delay

Figure 1-3. Open-loop control



# Closed-loop control

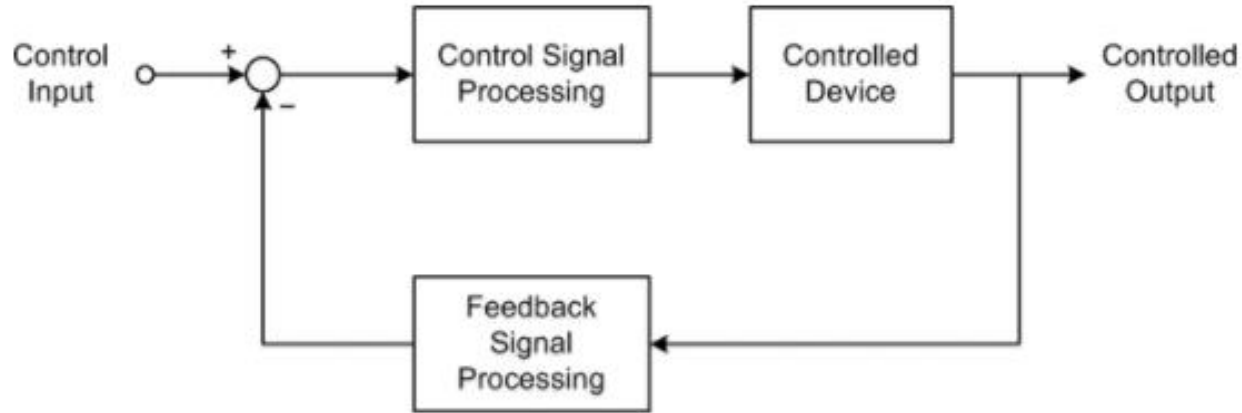


Figure 1-6. Closed-loop control

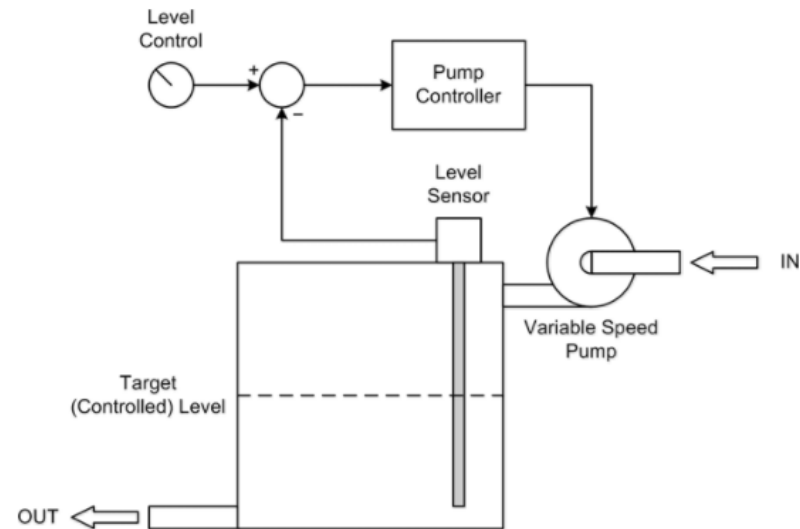


Figure 1-7. Closed-loop fluid level control

# Automated test setup

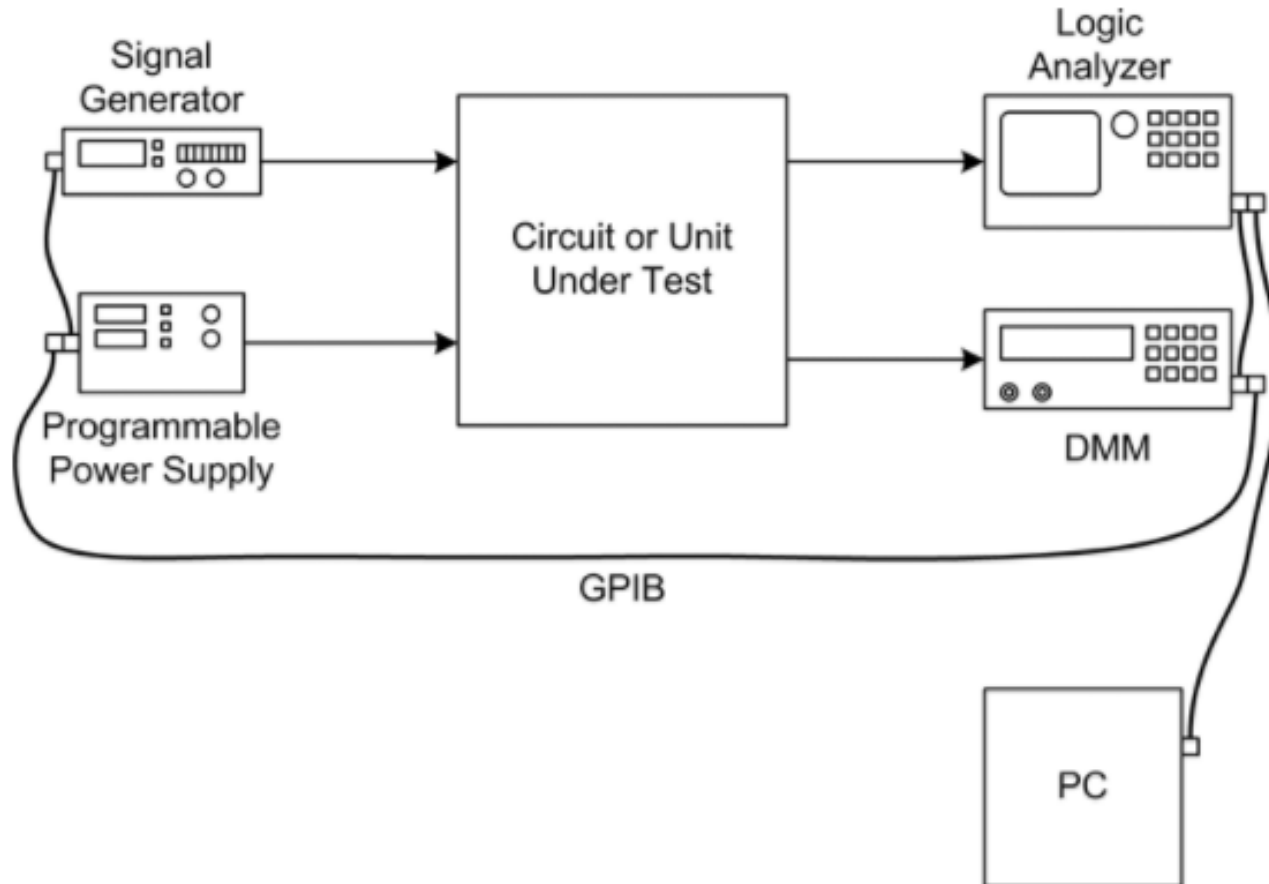
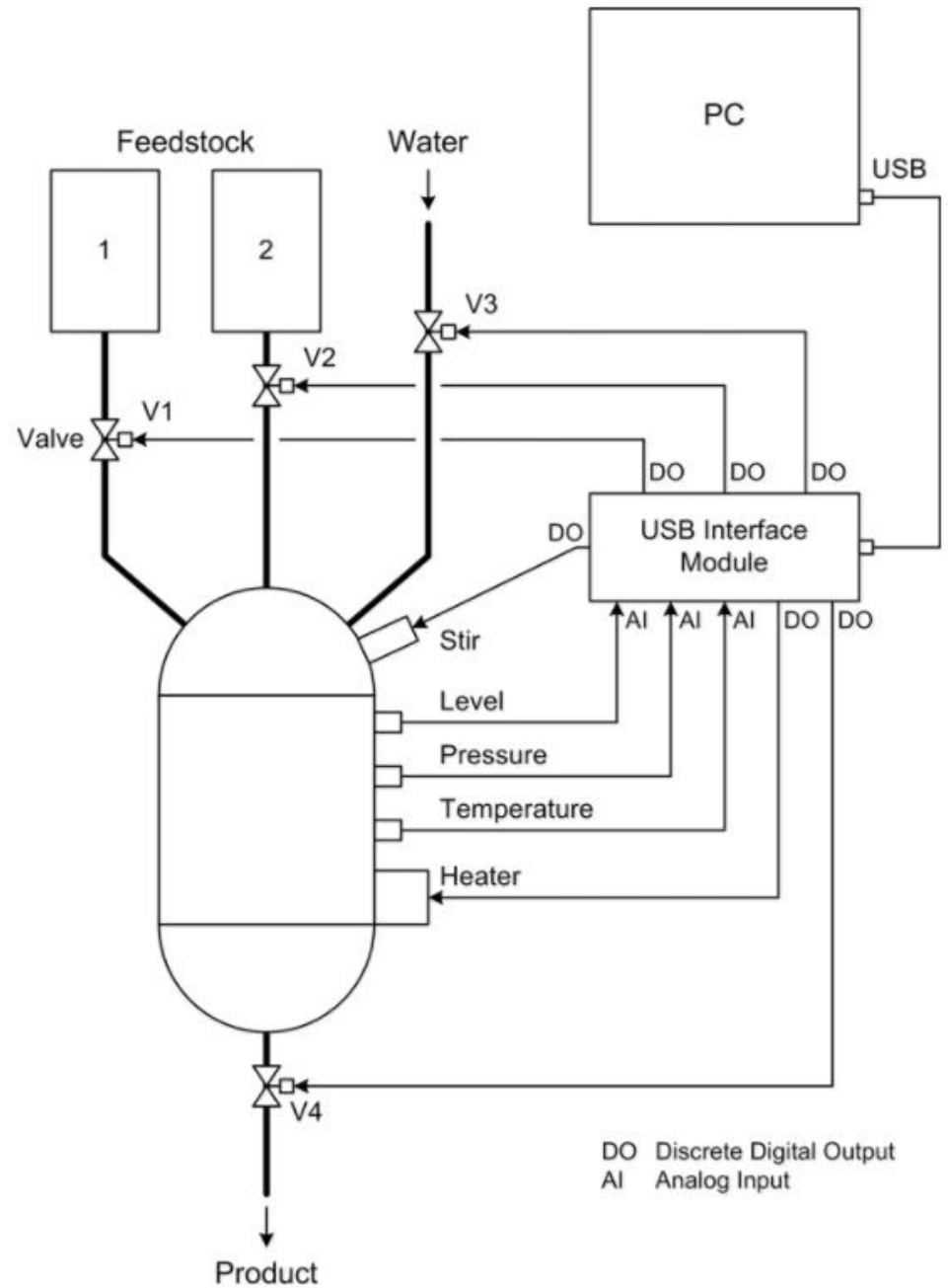


Figure 1-9. Test instrumentation example

# Process control



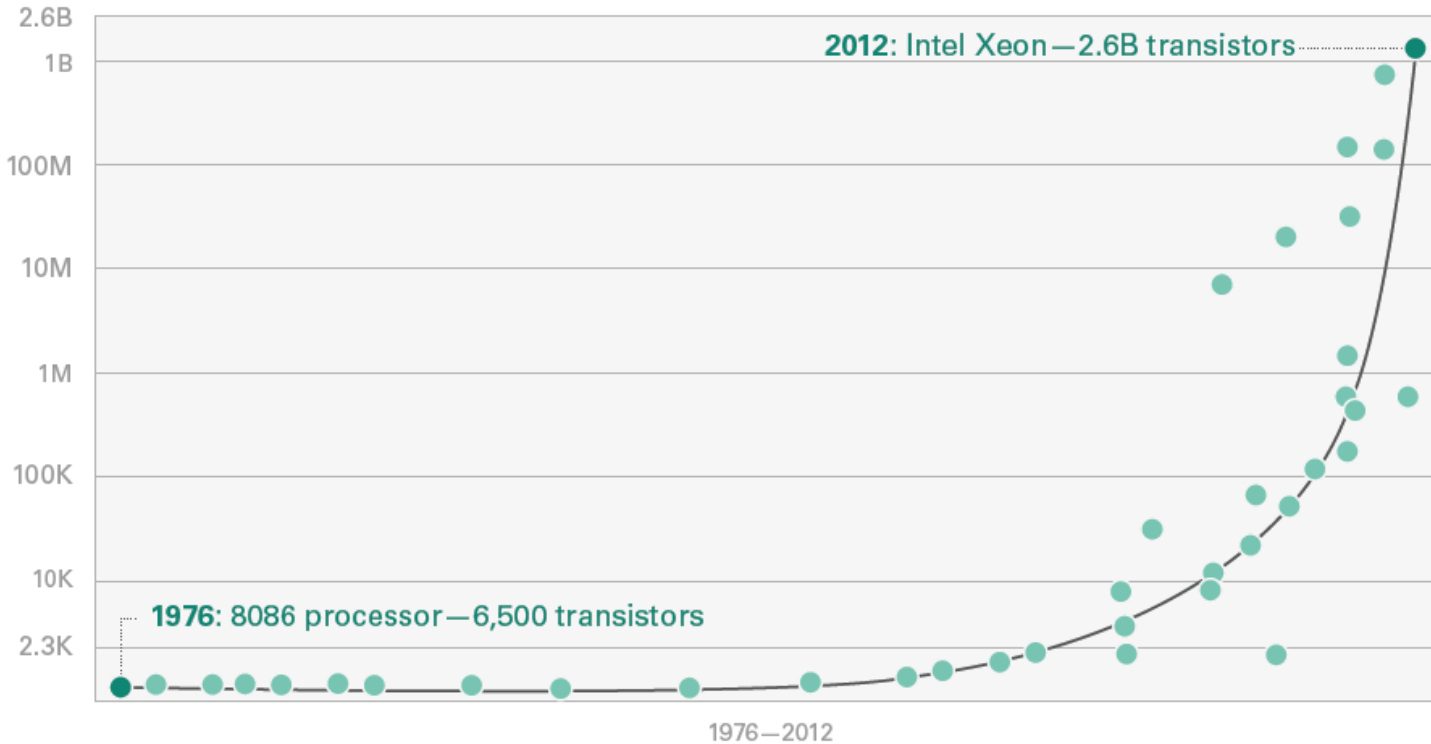
# **Trends, HW/SW I/O, Data transfer & Parallel processing**

- Selected background material



# Moore's law

TRANSISTOR COUNT BY DATE OF INTRODUCTION



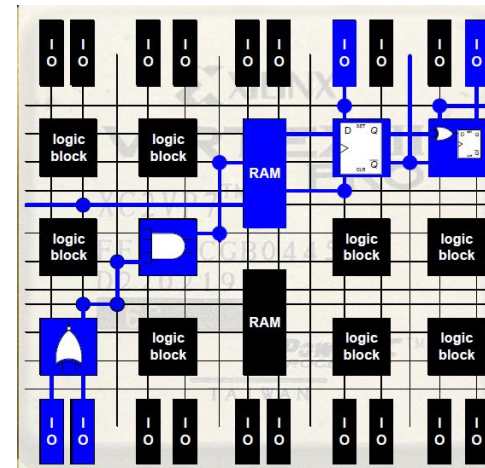
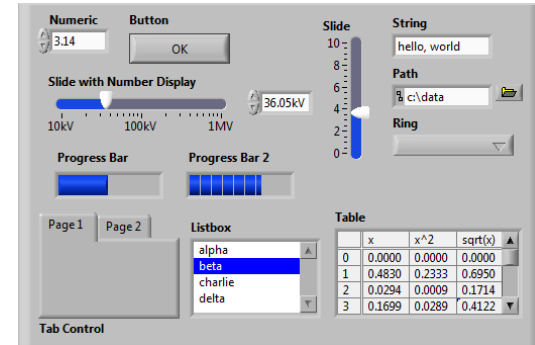
► Intel has been a key contributor to Moore's law over the past four decades, including its latest Xeon processors that contain 2.6 billion transistors.

**Moore's law:** the number of transistors per square inch on integrated circuits (or in general, the performance) doubles approximately every 18 months.



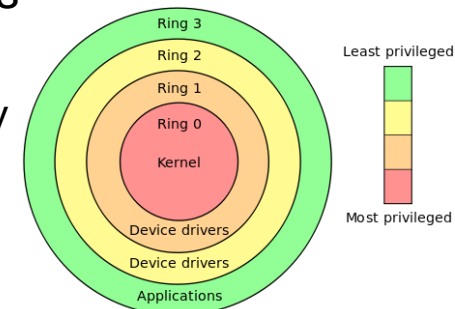
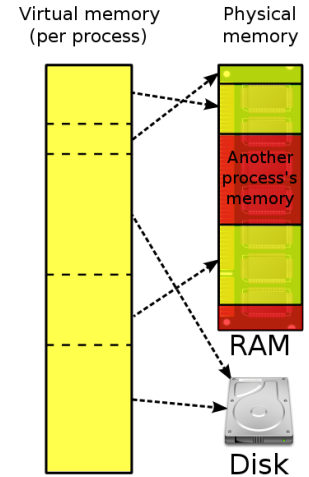
# Trends in Test and Measurement

- **Multicore CPUs and Parallel programming**
  - Increased CPU performance without increased clock rates
  
- **Software-Defined Instrumentation**
  - Can easily change functionality
  
- **FPGA-Enabled Instrumentation**
  - High performance, True parallelism, High determinism, High reliability, Reconfigurable
  
- **64 bit operating systems**
  - An “unlimited” address space allows much more RAM (Random Access Memory) in the computer
  
- **Network (Ethernet) connection**



# Protected mode

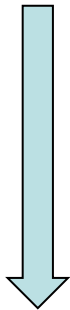
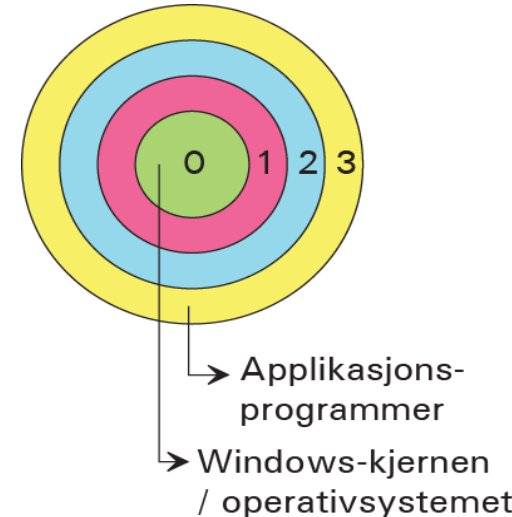
- In computing, **protected mode** is an operational mode of x86-compatible central processing units (CPU)
  - The term x86 refers to a family of instruction set architectures based on the Intel 8086 CPU
  - It allows system software to utilize features such as [virtual memory](#), [paging](#), [safe multi-tasking](#), and other features designed to increase an operating system's control over application software.
- In protected mode, there are four privilege levels or rings, numbered from 0 to 3, with ring 0 being the most privileged and 3 being the least. The use of rings allows for system software to restrict tasks from accessing data or executing privileged instructions.
- In most environments, the operating system and device drivers run in ring 0 and applications run in ring 3
  - most general-purpose systems (for example Windows 7) use only two rings, even if the hardware they run on provides more CPU modes.



# Windows architecture

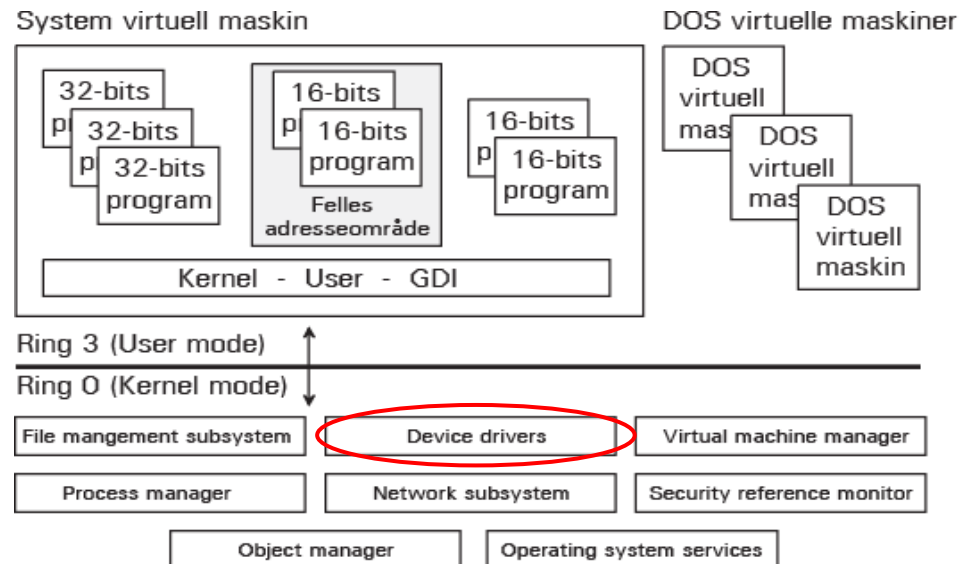
- Ring 0 : "all rights"
  - operating system (kernel) + device drivers
- Ring 3 : limited rights e.g. related to I/O
  - Applications (user programs)

## Privilege levels in Windows



• Need a device driver in order to allow hardware I/O operations from application programs

## 32 bit Windows XP



# Device Drivers

- In computing, a device driver or software driver is a computer program allowing higher-level computer programs to interact with a hardware device.
- A driver typically communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Drivers are hardware-dependent and operating-system-specific.

# Using device drivers in LabVIEW

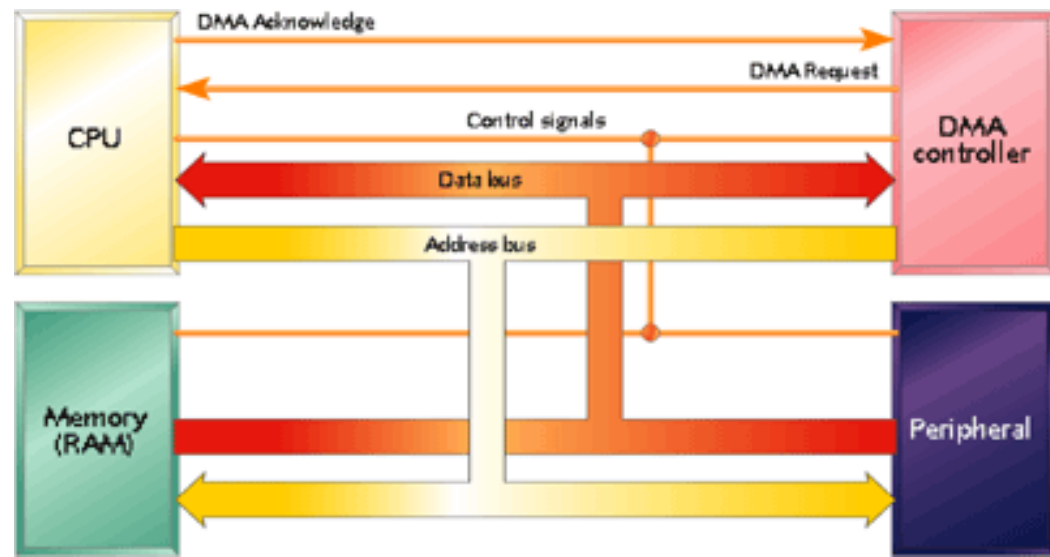
- **LabVIEW includes custom made drivers for serial communication, TCP, UDP, USB, DAQ etc.**
- All National instruments (NI) hardware is shipped with LabVIEW drivers
- Device drivers for specific instruments etc. can be downloaded from NI.com
- After your hardware driver software is installed, it is integrated into LabVIEW
- Most DAQ cards from other manufacturers also have LabVIEW drivers
- **Therefore, usually no device driver has to be written!**

# Interrupts

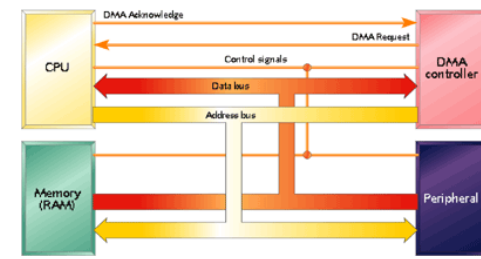
- One method to move acquired data to system memory (from a DAQ-card) is to generate an interrupt request (IRQ). signal. This signal can be generated when one sample is acquired or when multiple samples are acquired. The process of transferring data to system memory via interrupts is given below:
  - When data is ready for transfer, the CPU stops whatever it is doing and runs a special interrupt handler routine that saves the current machine registers, and then sets them to access the board.
  - The data is extracted from the board and placed into system memory.
  - The saved machine registers are restored, and the CPU returns to the original interrupted process.
- The actual data move is fairly quick, but there is **a lot of overhead time** spent saving, setting up, and restoring the register information.

# DMA introduction

- Direct memory access (DMA) is a system whereby samples are automatically stored in system memory while the processor (CPU) does something else.
- A computer usually supports several DMA channels.







# DMA (direct memory access) I

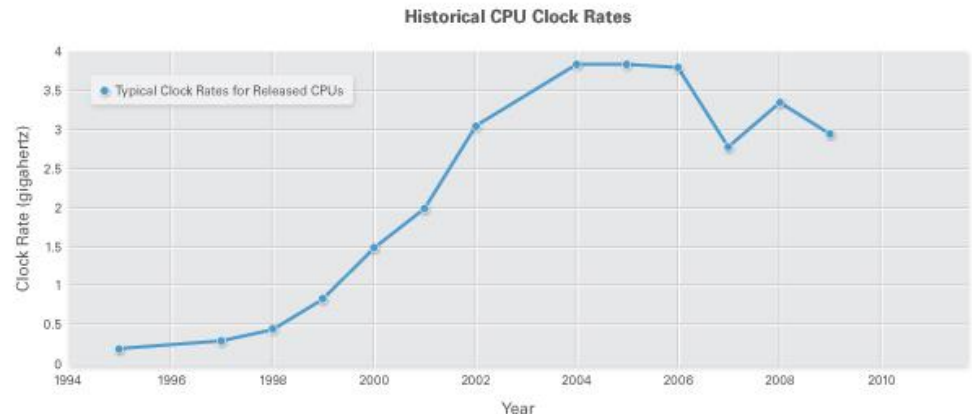
- DMA permits peripherals, such as a DAQ-card, to transfer data directly to or from memory without having each byte handled by the processor (CPU). Thus DMA enables more efficient use of interrupts, and increases data throughput.
- The process of transferring data via DMA is given below:
  - When data is ready for transfer, the DAQ-card notifies the DMA controller.
  - The DMA controller then asserts a DMA request signal to the CPU, asking its permission to use the bus (data bus, address bus, control bus).
  - The CPU completes its current bus activity, stops driving the bus, and returns a DMA acknowledge signal to the DMA controller.
  - The DMA controller then reads and writes one or more memory bytes, driving the address, data, and control signals as if it were itself the CPU.

# DMA (direct memory access) II

- When the transfer is complete, the DMA controller stops driving the bus and deasserts the DMA request signal. The CPU can then remove its DMA acknowledge signal and resume control of the bus.
- In **single-cycle mode**, the DMA controller gives up the bus after each transfer. This minimizes the amount of time that the DMA controller keeps the processor off of the memory bus, but it requires that the bus request/acknowledge sequence be performed for every transfer. This overhead can result in a drop in overall system throughput if a lot of data needs to be transferred.
- In **burst mode**, the DMA controller keeps control of the bus until all the data buffered by the requesting device has been transferred to memory (or when the output device buffer is full, if writing to a peripheral).

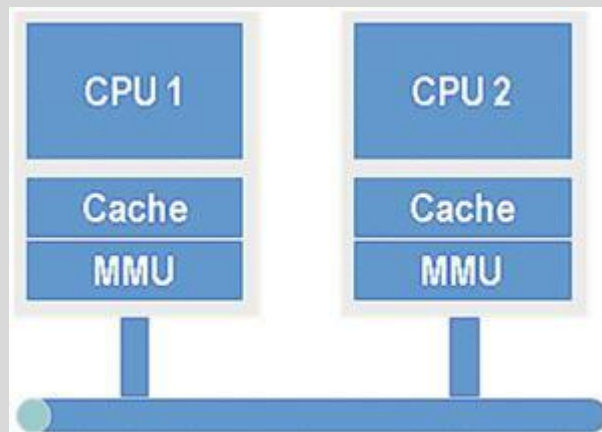
# Need for parallel programming

- Before 2005 innovations in processor technology resulted in computers with CPUs that operate at higher clock rates.
- However, as clock rates approached their theoretical physical limits, companies developed new processors with multiple processing cores.
  - **$P = ACV^2f$**  (P is consumed power, A is active chip area, C is the switched capacitance, f is frequency and V is voltage).
- With multicore processors the best performance and highest throughput is achieved by using parallel programming techniques



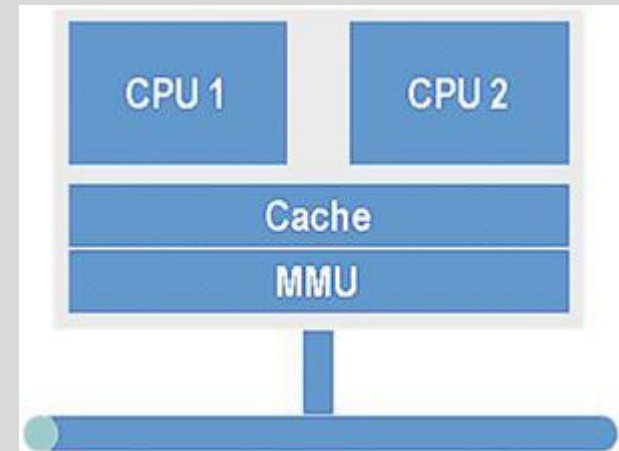
# Multiprocessors & Multicore Processors

Multiprocessor systems contain multiple CPUs that are not on the same chip



The multiprocessor system has a divided cache with long-interconnects

Multicore Processors contain any number of multiple CPUs on a single chip



The multicore processors share the cache with short interconnects

**Lowest inter-processor communication latency**  
→ **Fastest program execution time**

# Multicore Programming Goals

- Increase code execution speed
  - execution time: time from start to completion of task (response time)
- Maintain rate of execution but increase data throughput
  - throughput is the amount of work that can be done in a given time
- Evenly balance tasks across available CPUs (fair distribution of processing load)
- Dedicate time-critical tasks to a single CPU

In computing, **FLOPS** (floating-point operations per second) is a measure of a computer's performance

## Computer performance

Name	FLOPS
yottaFLOPS	$10^{24}$
zettaFLOPS	$10^{21}$
exaFLOPS	$10^{18}$
petaFLOPS	$10^{15}$
teraFLOPS	$10^{12}$
gigaFLOPS	$10^9$
megaFLOPS	$10^6$
kiloFLOPS	$10^3$

# Multicore Programming Challenges

- Thread (loop) Synchronization
  - Data loss (too slow data read vs. data write)
  - Reading the same data point multiple times
- Race Conditions
  - Uncertain result due to shared resources
- Deadlocks
  - Two or more processes are each waiting for the other to finish

# Hyper-threading

- Hyper-threading is a technology that was introduced by Intel, with the primary purpose of improving support for multi-threaded code.
- **Under certain workloads hyper-threading technology provides a more efficient use of CPU resources by executing threads in parallel on a single processor.**
- Hyper-threading works by duplicating certain sections of the processor.
- A hyper-threading equipped processor (core) pretends to be two "logical" processors to the host operating system, allowing the operating system to schedule two threads or processes simultaneously.
- E.g. Xeon, Core i5 and Core i7 Intel processors implement hyper-threading.

# Hardware acceleration

- Normally, processors (CPUs) are sequential, and instructions are executed one by one.
- In computing, **hardware acceleration** is the use of computer hardware to perform some function faster than is possible in software running on the general-purpose CPU.
- The main difference between hardware and software is **concurrency**, allowing hardware to be much faster than software. Hardware accelerators are designed for computationally intensive software code
- Examples of hardware accelerators includes graphics processing units (**GPUs**) and field-programmable gate array (**FPGA**)

