MECE336– Microprocessors I

Lecture 1 – Introduction and Background

Associate Prof. Dr. Klaus Werner Schmidt

Department of Mechatronics Engineering – Çankaya University

Compulsory Course in Mechatronics Engineering
Credits (3/2/4)

Course Webpage: http://MECE336.cankaya.edu.tr

Content and Structure

Content

- Embedded Systems and Microcontrollers
- PIC 16F84A Architecture
- Assembly Programming Basics
- Input/output Usage
- Arithmetic Operations
- Timers and Interrupts, Sleep Mode and Watchdog Timer
- Programming Examples

Structure

- 3 lecture hours: Monday 15:20 – 17:10, Tuesday 9:20 – 10:10
- 2 Lab hours: Wednesday 13:20 – 15:10; Wednesday 15:20 – 17:10
- Office hours: Tuesday 10:20 – 11:00

Grading and Literature

Grading

- 13 Laboratories (25%)
- 1 Midterm Exam (25%)
- 1 Final Exam (35 %)
- 1 Project (15 %)

Literature

- Wilmhurst, Tim: "Designing Embedded Systems with PIC Microcontrollers: Principles and Applications", Elsevier Ltd., 2010 (ISBN: 9-78-1-85617-750-4) (Main Textbook)
- Mazidi, Muhammad Ali, McKinlay, Rolin D., Causay, Danny: "PIC Microcontrollers and Embedded Systems", Pearson International Education, 2008 (ISBN: 0-13-600902-6)

Embedded Systems: Basics

Embedded System Definition

A system whose principal function is not computational, but which is controlled by a computer embedded with it

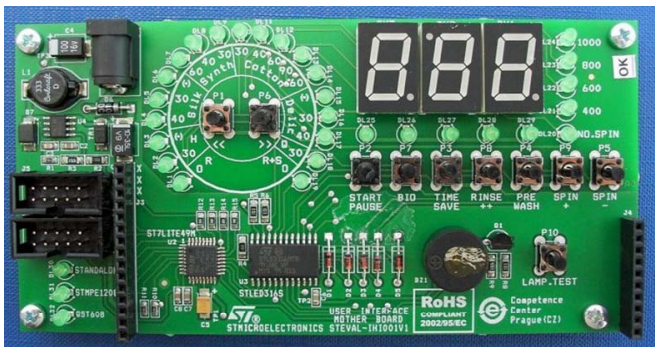
- Embedded system has functionality different from only computation
- Some part/functionality of an embedded system needs computations
- A computing device is embedded in the embedded system

Usage of Embedded Systems

- Household, office, factories
- Cars, trains, other vehicles
- Hospitals
- Toys

Embedded Systems: Examples

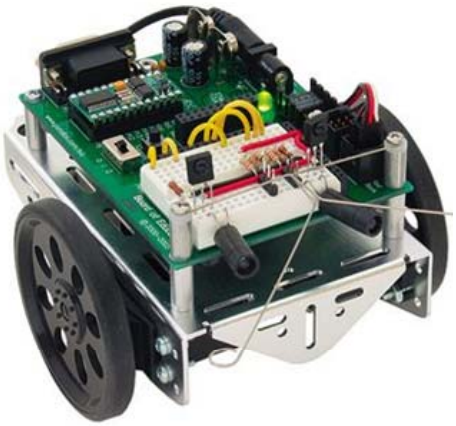
Washing Machine



Microwave

Embedded Systems: Examples

Autonomous Robot

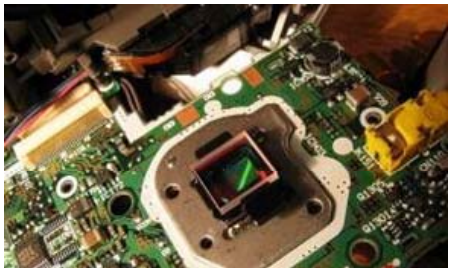


Digital Watch

Embedded Systems: Examples

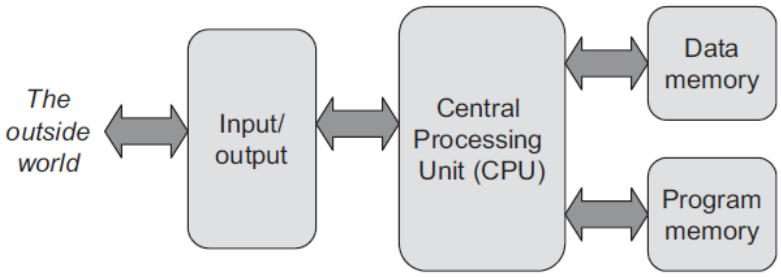
Cell Phone

Digital Camera



Embedded Systems: General Features

Overview



- CPU performs computations
- Program memory stores the instructions of the desired program
- Data memory stores application data values
- Input/output block for providing input data and reading output data
- Arrows represent data exchange

Embedded Systems: Description

Central Processing Unit (CPU)

- Perform arithmetic or logical computations
- Work through a series of instructions (program)
- Instructions perform simple tasks
- Combination of many instructions provide much computational power

Input/Output

- Interface to the outside world
- Sensors/actuators (Temperature sensor, button, motor, ...)
- Human input devices (Keyboard, mouse, touchpad, ...)
- Human output devices (Monitor, display, printer, ...)

Embedded Systems: Description

Program Memory

- Holds program to execute
- Needs to be permanent (also if there is no power)
- Program is ready to run as soon as power is applied

Data Memory

- Holds temporary data values (results of a computation, ...)
- Need not be permanent (can be erased if there is no power)

Data Paths

- I/O $\Leftrightarrow$ CPU
- Data Memory $\Leftrightarrow$ CPU
- Program Memory $\Leftrightarrow$ CPU

Embedded Systems: Instructions

Instruction Set

- Set of instructions recognized by a CPU
- Programs are build up from instructions
- Different instruction sets depending on complexity

CISC: Complex Instruction Set Computer

- Many and sophisticated instructions with different levels of complexity
- Simple instructions: usually 1 byte $\rightarrow$ quick execution
- Complex instructions: usually several bytes $\rightarrow$ slow execution

RISC: Reduced Instruction Set Computer

- Simple instruction set for fast operation
- Each instruction is contained in a single binary word
- Instruction includes code and address data

Embedded Systems: Memory Types

Volatile Memory

- Only works if device is powered and loses data if there is no power
  $\rightarrow$ temporary data storage that can be used as data memory
- Usually simple semiconductor technology (easy to write)
- Denoted as Random Access Memory (RAM)

Non-volatile Memory

- Retain stored value even if device is not powered
  $\rightarrow$ permanent storage that can be used as program memory
- Non-volatile semiconductor technology, more difficult to write
- Denoted as Read-only Memory (ROM)

Embedded Systems: Memory Organization

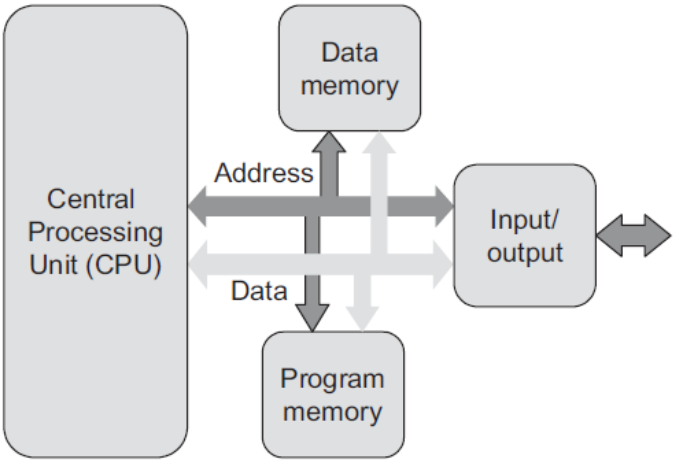
Memory Interaction

- Basic requirement: Read and/or write data
  → Determine memory address and move data
- Usage of buses for this purpose
  → Address bus for specifying memory address
  → Data bus for moving data

Illustration

Embedded Systems: Memory Architectures

Von Neumann Architecture



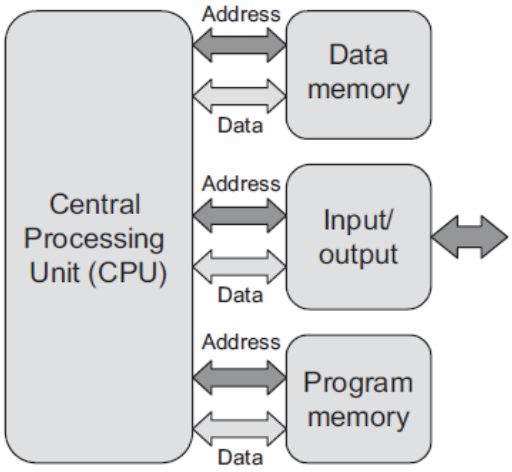
Description

- One address bus
- One data bus
  → Each bus serves program memory, data memory and I/O

- Advantage: Flexible division of memory in program and data memory
- Disadvantage: Shared ⇒ only one component can access at a time
- Disadvantage: Same bus for all memory sizes (small/large words)

Embedded Systems: Memory Architectures

Harvard Structure



Description

- Every memory area has own data bus
- Every memory area has own address bus
  → Each bus only serves a single memory area

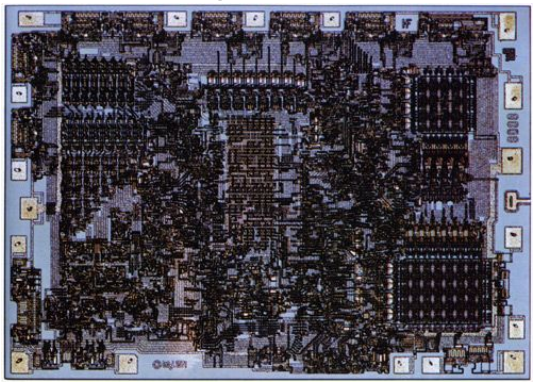
- Advantage: Flexibility in bus size depending on memory area
- Advantage: Simultaneous access of different memory areas
- Disadvantage: Program and data memory must be separate

Embedded Systems: Microprocessors (MPs)

Development

- MP is meant as a computing device
- First MP developed by Intel (1970)
  → 4-bit, 2300 transistors, 60000 operations/sec
  → significant processing power at low cost and small space
- Included only CPU
- Functionality such as memory, I/O outside MP
- Current MPs also include some memory on CPU

First Microprocessor



Embedded Systems: Microcontrollers (MCs)

Necessity of Microcontrollers

- Notice that computing power can also be used for control
  → Use microprocessors to control products such as embedded application examples
- Properties of such applications
  - Small/moderate computing power requirement at possibly low price
  - No/less need for a developed human interface
  - Has to work in harsh environments (hot, cold, impact)
  - Need input/output interface for sensors/actuators

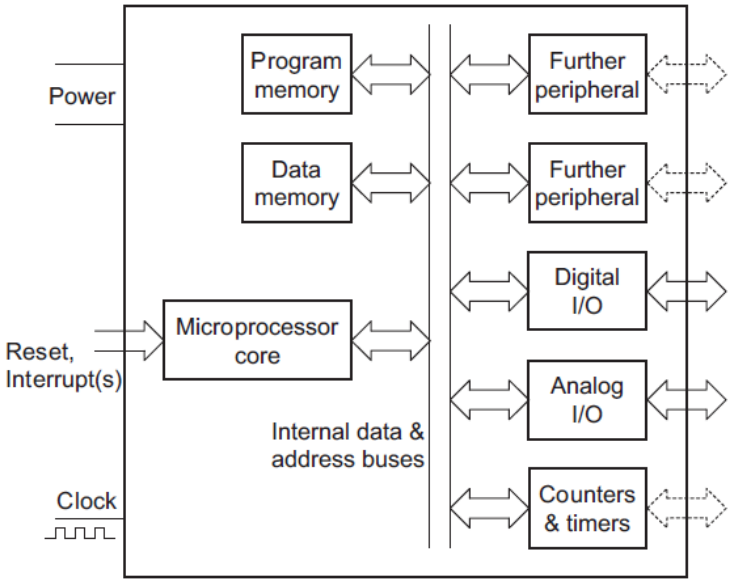
  ⇒ Compact computing device including memory and interfacing is needed

  ⇒ Microcontrollers are developed for this purpose

Embedded Systems: Microcontroller Organization

Components

- Simple MP core
- Data/program memory
- Peripherals for interfacing
- Power, clock



MC Examples

- Microchip (PIC), Atmel, Infinion, NXP, STMicroelectronics, Analog Devices, Texas Instruments
  → This lecture: PIC 16F84A microcontroller

Numbering Systems: Decimal, Binary, Hex

Decimal System (Base 10)

- 10 symbols: $0, 1, \ldots, 9$

Binary System (Base 2)

- 2 symbols: $0, 1$

Hexadecimal System (Base 16)

- 16 symbols: $0, 1, \ldots, 9, A, B, C, D, E, F$

Conversion from Decimal to Binary

- Use weight 2^i of the i-th bit in a binary number

Example

bit	6	5	4	3	2	1	0
number	1	0	0	1	1	0	1
weight	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$\rightarrow 1001101_2 = 2^6 + 2^3 + 2^2 + 2^0 = 64 + 8 + 4 + 1 = 77_{10}$

Numbering Systems: Decimal, Binary, Hex

Conversion from Binary to Decimal

- Find coefficients for each weight 2^i of the i-th bit in a binary number

Example

- $45 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
  $\rightarrow 101101_2 = 45_{10}$

Conversion from Binary to Hexadecimal

- Note: 4 bits of a binary number correspond to a hexadecimal number
  $\rightarrow$ Separate binary number into 4 bit pieces

Example

bit	12	11	10	9	8	7	6	5	4	3	2	1	0
number	1	0	1	0	0	1	1	0	0	0	1	1	0
weight	1	4				C				6			

$\rightarrow 1010011000110_2 = 14C6_{16}$

Numbering Systems: Decimal, Binary, Hex

Conversion from Hexadecimal to Binary

- Write each hexadecimal symbol as 4 binary bits

Example $2A7E_{16}$

Conversion between Decimal and Hexadecimal

- Analogous to conversion between Decimal and Binary with base 16

Examples

Number Systems: ASCII

Description

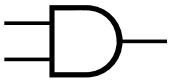
- American Standard Code for Information Interchange
- Binary patterns (7 bit) for numbers, letters, punctuation marks
  $\rightarrow$ Information can be shared among computers

Examples

Symbol	ASCII	Symbol	ASCII	Symbol	ASCII	Symbol	ASCII
0	30	A	41	a	61	$\star$	2A
1	31	B	42	b	62	$+$	2B
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$-$	2D
8	38	Y	59	y	79	$\backslash$	2F
9	39	Z	5A	z	7A	?	3F

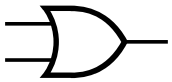
Logic Operations: AND, OR, XOR, NAND, NOR,

Logic AND



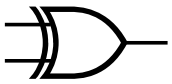
Input		Output
A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

Logic OR



Input		Output
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Logic XOR

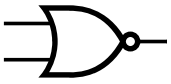


Input		Output
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Logic NAND

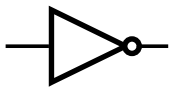
Input		Output
A	B	$\overline{A \& B}$
0	0	1
0	1	1
1	0	1
1	1	0

Logic NOR



Input		Output
A	B	$\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

Logic NOT



Input	Output
A	$\overline{A}$
0	1
1	0

Logic Operations: Examples