Microcontroller: A key contributor in ES* (from here: session 5)

Amir Mahdi Hosseini Monazzah

Room 332,
School of Computer Engineering,
Iran University of Science and Technology,
Tehran, Iran.

monazzah@iust.ac.ir

Fall 2024

* Parts of slides are adopted from Prof. Prabal Dutta lectures, Department of Electrical Engineering and Computer Science, University of Michigan

Outline

- Different types of processors
  - Characteristics
- How dose ES affect the processor architecture?
  - ES tasks
  - Architectural differences

Processors

- Execute programs
  - Serial execution of instructions
  - Simple, universal
- Instruction execution engine: fetch/execute cycle
  - Flow of control determined by modifications to program counter
  - Instruction classes:
    - Data: move, arithmetic and logical operations
    - Control: branch, loop, subroutine call
    - Interface: load, store from external memory

Processors (cont.)

- Traditional architecture goal: Performance
  - Caches
  - Branch prediction
  - Multiple/Out of Order (OoO) issue

Processors

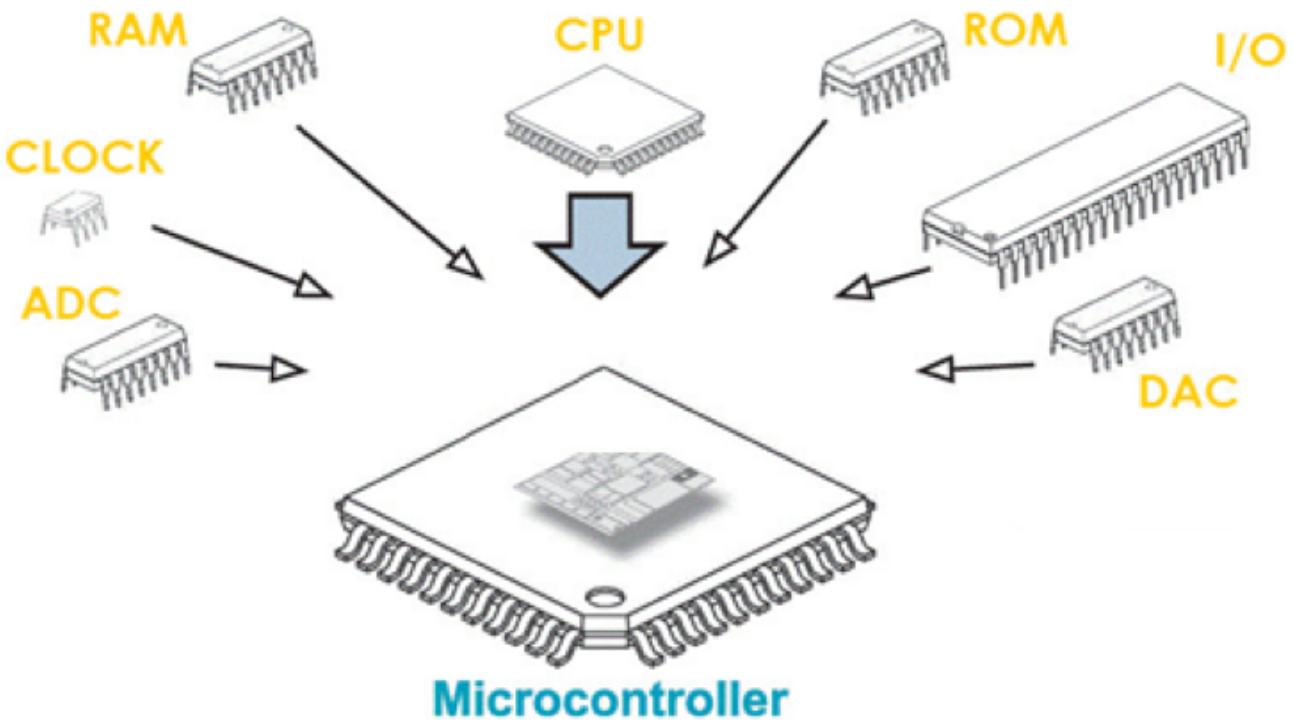
- Processor is a universal computing engine
  - Program can compute arbitrary functions
  - Use a processor for simple/specific tasks
- Advantages:
  - High-level language
  - Compilers/debuggers
  - Arbitrary control structures
  - Arbitrary data structures
- Disadvantages:
  - Cost / Size

Embedded processor (microcontroller)

- Processor optimized for low cost
  - No cache
  - Small memory
  - No disks
  - 4 bit/8 bit/16 bit
  - No FP
  - No complicated datapath
  - Multicycle instruction interpretation
  - Simple/no operating system
  - Programs are static

Embedded processor (microcontroller)

- Low performance
  - 1 MIPS is enough if 1 ms is the time scale
- Integrate on a single chip



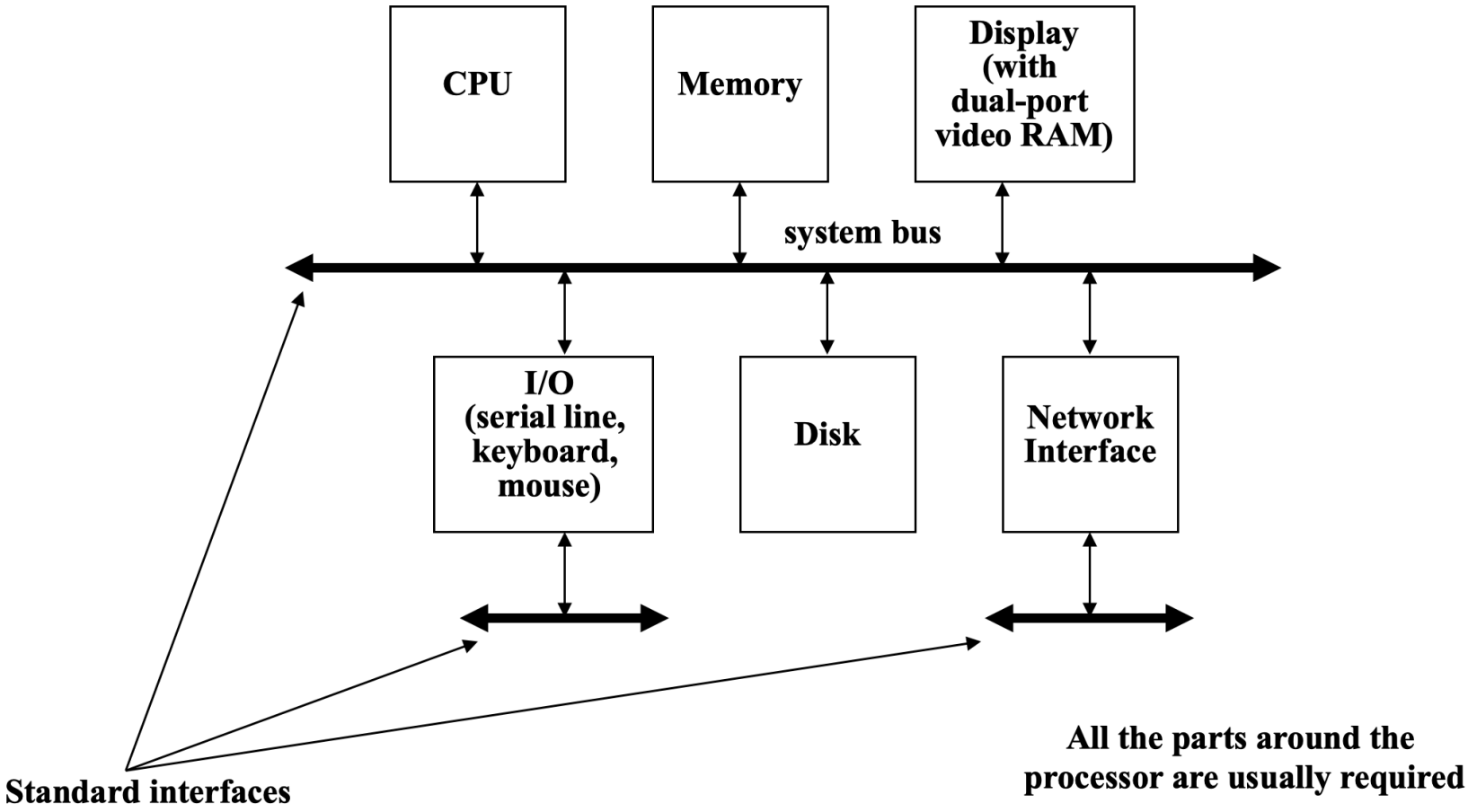
General-purpose processor

- Programmed by user
- New applications are developed routinely
- General-purpose
  - Must handle a wide ranging variety of applications
- Interacts with environment through memory
  - All devices communicate through memory
  - DMA operations between disk and I/O devices
  - Dual-ported memory (as for display screen)
- Oblivious to passage of time (takes all the time it needs)

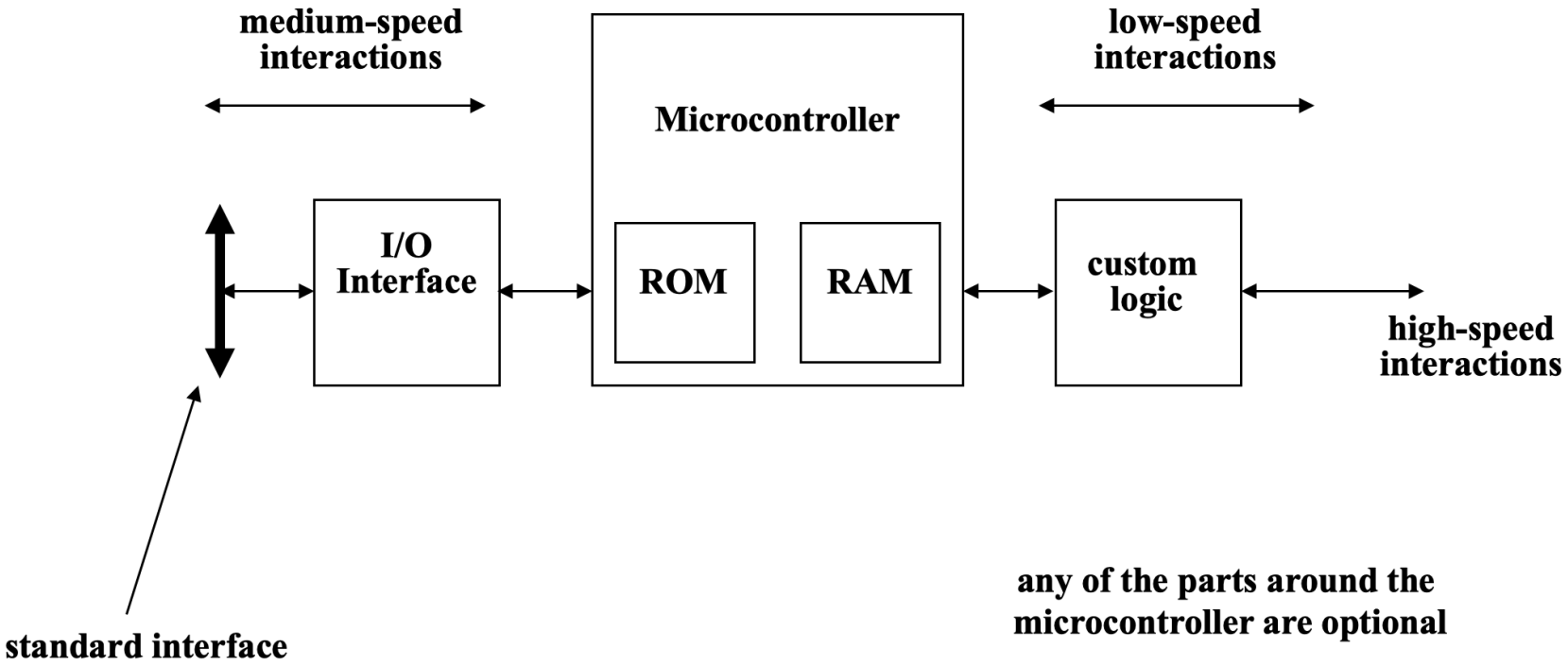
Embedded processors

- Programmed once by manufacturer of system
- Executes a single program (or a limited suite) with few parameters
- Task-specific
  - Can be optimized for specific application
- Interacts with environment in many ways
  - Direct sensing and control of signal wires
  - Communication protocols to environment and other devices
  - Real-time interactions and constraints

Typical general-purpose architecture



Typical Task-Specific Architecture



How does ES change things?

- Sense and control of environment
  - Processor must be able to “read” and “write” individual signal wires
  - Controls I/O devices directly
- Program has to do everything
  - Sense input bits
  - Change output bits
  - Do computation
  - Measure time
    - Many applications require precise spacing of events

How does ES change things?

- Problems with this
  - Precise timing?
  - Too slow?

Connecting to inputs/outputs

- Map external wire to a bit of a variable (memory or register)
  - if (a & 1) . . .
    X = 2

In-class assignment (from here: session 6)

- Write a C program that measures the acceleration
  - Accelerometer output has one wire!
    - Acceleration coded as the duty cycle
    - Pulse-width/cycle-length
      » 20% = -128
      » 80% = +127
    - Cycle time = 10ms
  - Input is low-order bit of variable X
  - Assign result to variable Z
  - Make up whatever you need

In-class assignment: answer

- Answer by Mr. Arshia Moghimi (SUT)

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>

#define CYCLE_LENGTH 10

int main() {
    int X= 0, Z = 0, counter = 0, counter1 = 0;
    float step = 256 / 0.6;
    struct timeval tv;
    gettimeofday(&tv, NULL);
    unsigned long start = 1000000 * tv.tv_sec + tv.tv_usec; // time in microsecond

    while (1) {
        scanf("%d", &X);
        counter++;
        counter1 += (X & 1);
        gettimeofday(&tv, NULL);
        if (1000000 * tv.tv_sec + tv.tv_usec - start >= 1000000 * CYCLE_LENGTH) {
            Z = (1.0 * counter1 / counter - 0.2) * step - 128;
            printf("acceleration = %d\n", Z);
            gettimeofday(&tv, NULL);
            start = 1000000 * tv.tv_sec + tv.tv_usec;
            counter1 = 0;
            counter = 0;
        }
    }

    return 0;
}
```

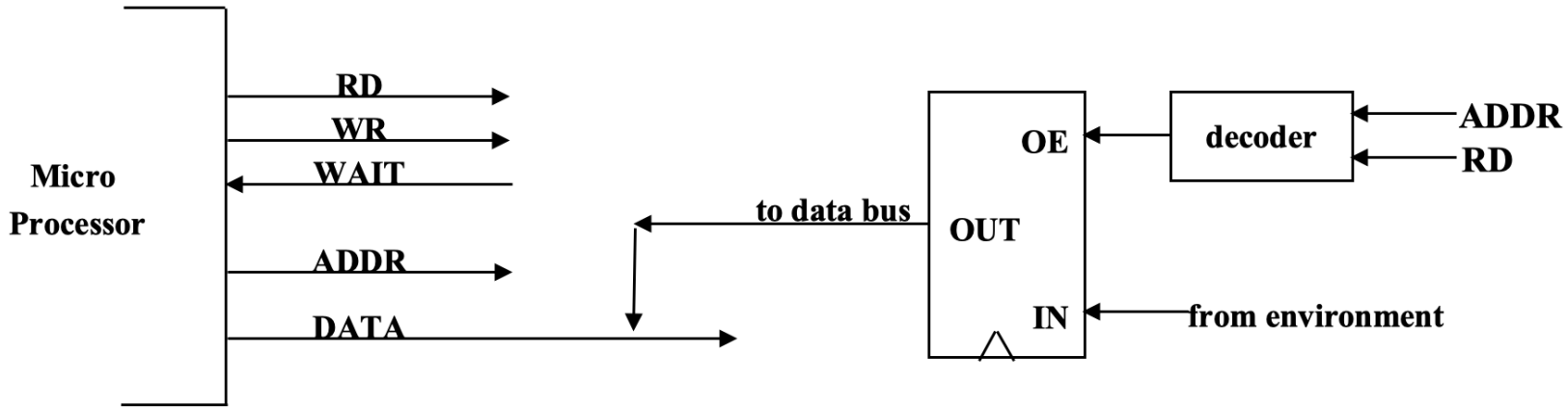
In-class assignment: answer

- Answer by Mr. Mohammad Javad Mirshekari Haghighi (IUST)

```
#include <iostream>
using namespace std;

char X = 1; // input port, update auto
char OUT = 0; // output port, acceleration
char MAX_OUT = 127; // 'b01111111

struct TimePoint
{
    int ms; // time in milisecond
}; // sample time struct

TimePoint time(){
    // return current time as TimePoint
    struct TimePoint now;
    return now;
}

int main(){
    struct TimePoint start_1;//   _____________
    struct TimePoint end_1; //____|             |____ positive pulse

    struct TimePoint start_0;//___               _____ negative pulse
    struct TimePoint end_0;//    |_____________|

    bool value = false; // value of the pulse
    bool is_high = false; // true, if value is true
    float duty_cycle = 0; // duty cycle

    while(true) { // read input port continuously
        // get pulse value
        value = (X >> 0) & 0x1;
        // set, update duty cycle
        if(value == true && !is_high){
            is_high = true;
            start_1 = time();
            end_0 = start_1;
        }
        else if(value == false && is_high) {
            is_high = false;
            end_1 = time();
            start_0 = end_1;
        }

        // calculate duty cycle
        duty_cycle = (end_1.ms - start_1.ms) / ( end_0.ms - start_0.ms ) + (end_1.ms - start_1.ms);
        // set OUT, according to the range [0.2, 0.8] of duty cycle
        OUT = (MAX_OUT * (duty_cycle - 0.5)) / 0.3;
    }
    return 0;
}
```

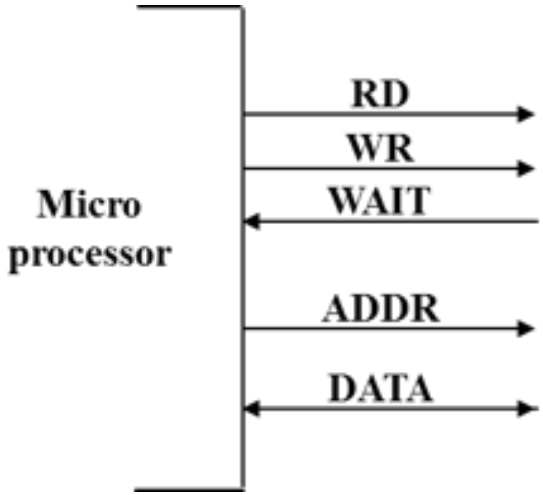
Memory-mapped inputs

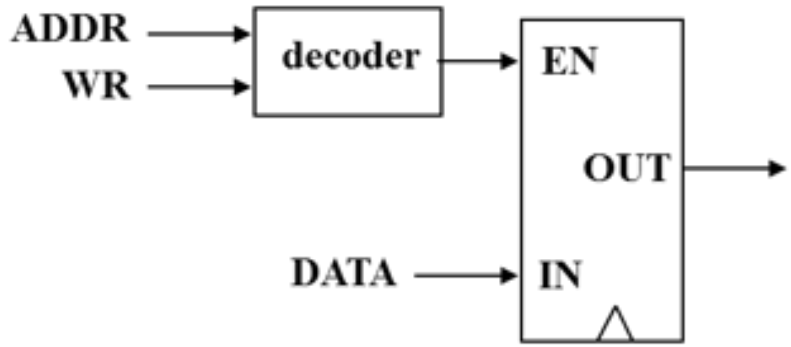
- Map external wire to a bit in the address space of the processor
- External register buffers values coming from environment
  - Map register into address space
  - Decoder needed to select register for reading
  - Output enable (OE) so that many registers can use the same data bus



Memory-mapped outputs

- Map external wire to a bit in the address space of the processor
- Connect output of memory-mapped register to environment
  - Map register into address space
  - Decoder need to select register for writing (holds value indefinitely)
  - Input enable (EN) so that many registers can use the same data bus





On-chip support for communication (up to here: session 5)

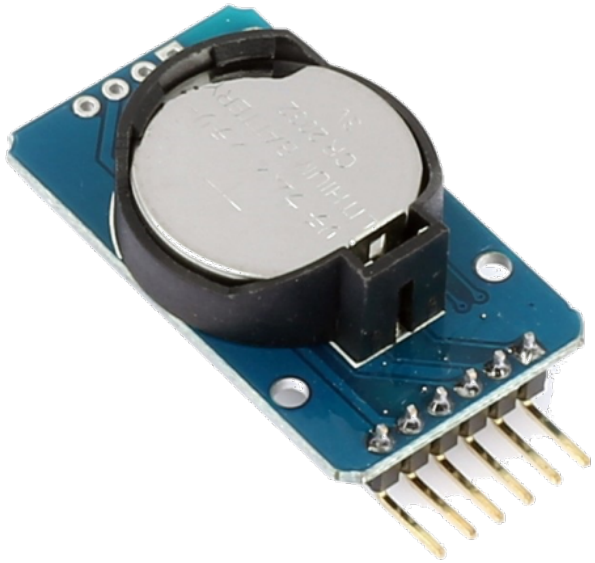
- Processor may not be fast enough
  - Offload standard protocols
- Built-in device drivers
  - For common communication protocols
  - Serial-line protocols most common as they require few pins
    - e.g. RS-232 serial interface - Special registers in memory space for interaction
- Increases level of integration
  - Pull external devices on-chip
    - Must be standard
  - Eliminate need for shared memory or system bus

Measuring time

- Keep track of detailed timing of each instruction's execution
  - Highly dependent on code
  - Hard to use with compilers
  - Not enough control over code generation
  - Interactions with caches/instruction-buffers
- Loops to implement delays
  - Keep track of time in counters
  - Keeps processor busy counting and not doing other useful things

Measuring time (cont.)

- Real-time clock
  - Sample at different points in the program
  - Simple difference to measure time delay



Timers

- Separate and parallel counting unit(s)
  - Co-processor to microprocessor
  - Does not require microprocessor intervention
  - In simple case, like a real-time clock
    - Set timer/read timer
  - Interrupt generated when expired
- More interesting timer units
  - Self reloading timers for regular interrupts
  - Pre-scaling for measuring larger times
  - Started by external events

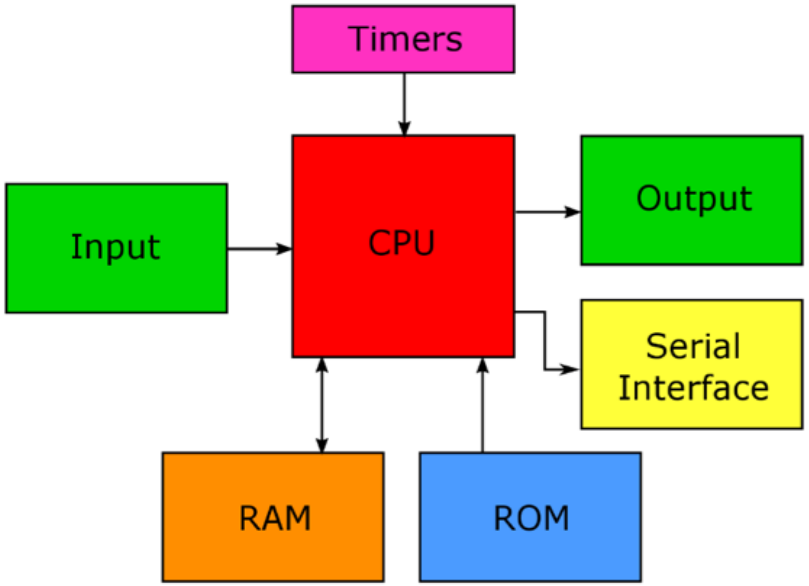
Timer: Input/output events

- Input capture
  - Record precise time when input event occurred
  - To be used in later handling of event
- Output compare
  - Set output to happen at a point in the future
  - Reactive outputs – set output to happen a pre-defined time after some input
  - Processor can go on to do other things in the meantime

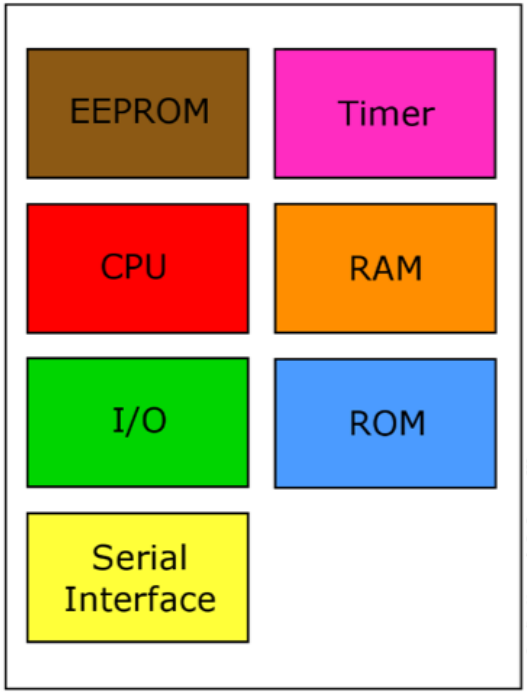
Microprocessor vs microcontroller

- What is the main difference between microcontroller and microprocessor?

Microprocesser: CPU
and several supporting chips.

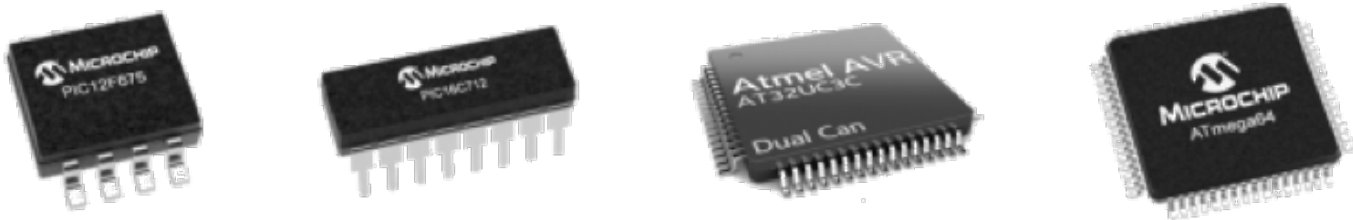


Microcontroller: CPU
on a single chip.



What is a microcontroller?

- A **microcontroller** (**MCU** for microcontroller unit, or **UC** for μ-controller)
  - A small computer on a single integrated circuit.
  - It is similar to, but less sophisticated than, a system on a chip (SoC)
    - SoC may include a microcontroller as one of its components.



8Bit Microcontroller | 16Bit Microcontroller | 32Bit Microcontroller | 64Bit Microcontroller

What is a microcontroller? (cont.)

- A microcontroller contains
  - One or more CPUs (processor cores)
  - Memory
    - Ferroelectric RAM
    - NOR flash
    - OTP ROM
    - RAM
  - Programmable input/output peripherals.
- Microcontrollers are designed for embedded applications


The End