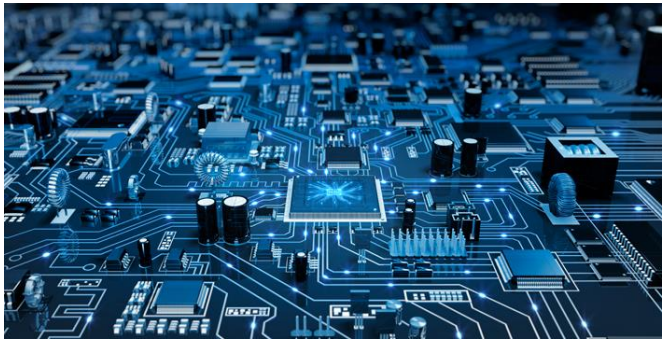
C Programming and Embedded Systems

Instructor: Tinoosh Mohsenin
Modified from slides by Alexander Nelson

What is an Embedded System?



- Loose Definition:
  - A system which is part of a larger system
- What does that actually mean
  - Hardware/Software co-design
    - Usually with specific purpose
    - Usually special set of constraints (size/power/time)
  - From Wiki:
    - An **embedded system** is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is *embedded* as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today.
- Can be based on microcontroller, DSP, ASICs, even standard Microprocessor unit(MPU)

Examples

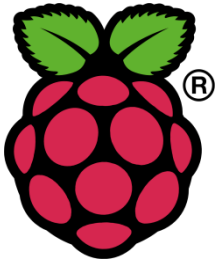


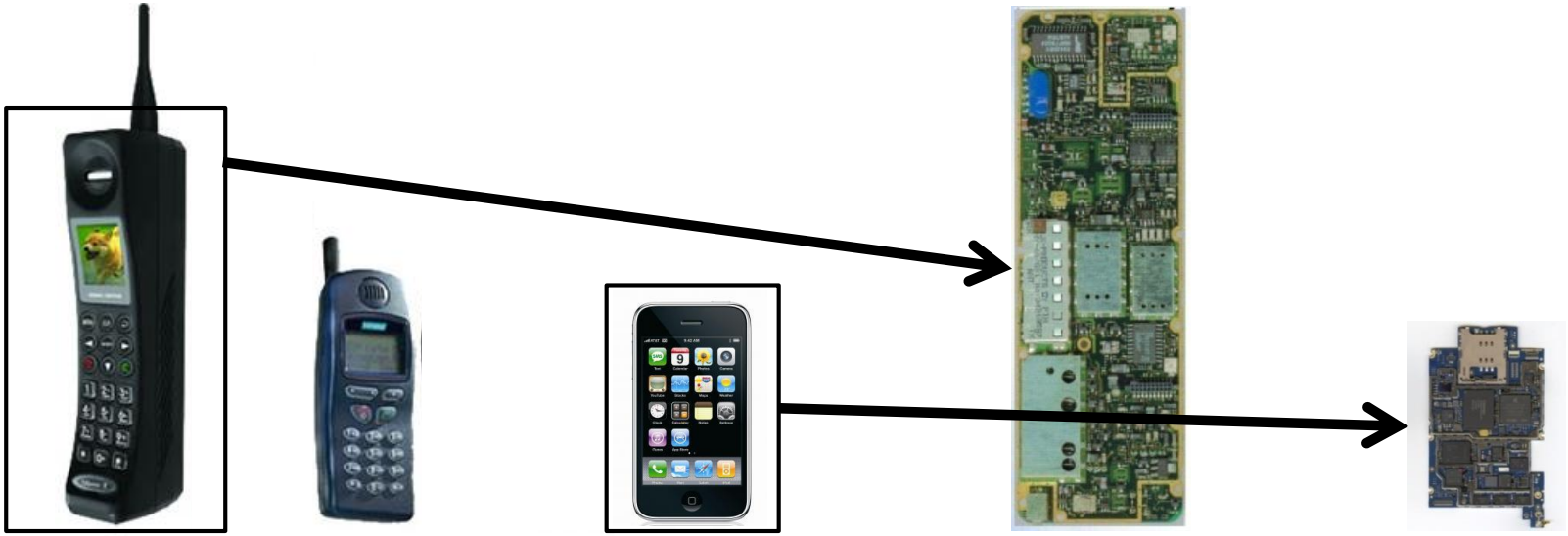
*Image credit to Fitbit, Arduino, Google, ford, we-online, Acorn Instruments, Raspberry Pi and used under Fair Use law. I do not claim ownership to any of these images

Market Share

- Embedded industry represents about $17 billion in revenue per year
  - Compare to $19 B for cellphone processors
  - Compare to ~$25 B for non-mobile, non-embedded MPUs
- Embedded Systems outnumber other MPUs by over 100:1

Data pulled from icinsights.com

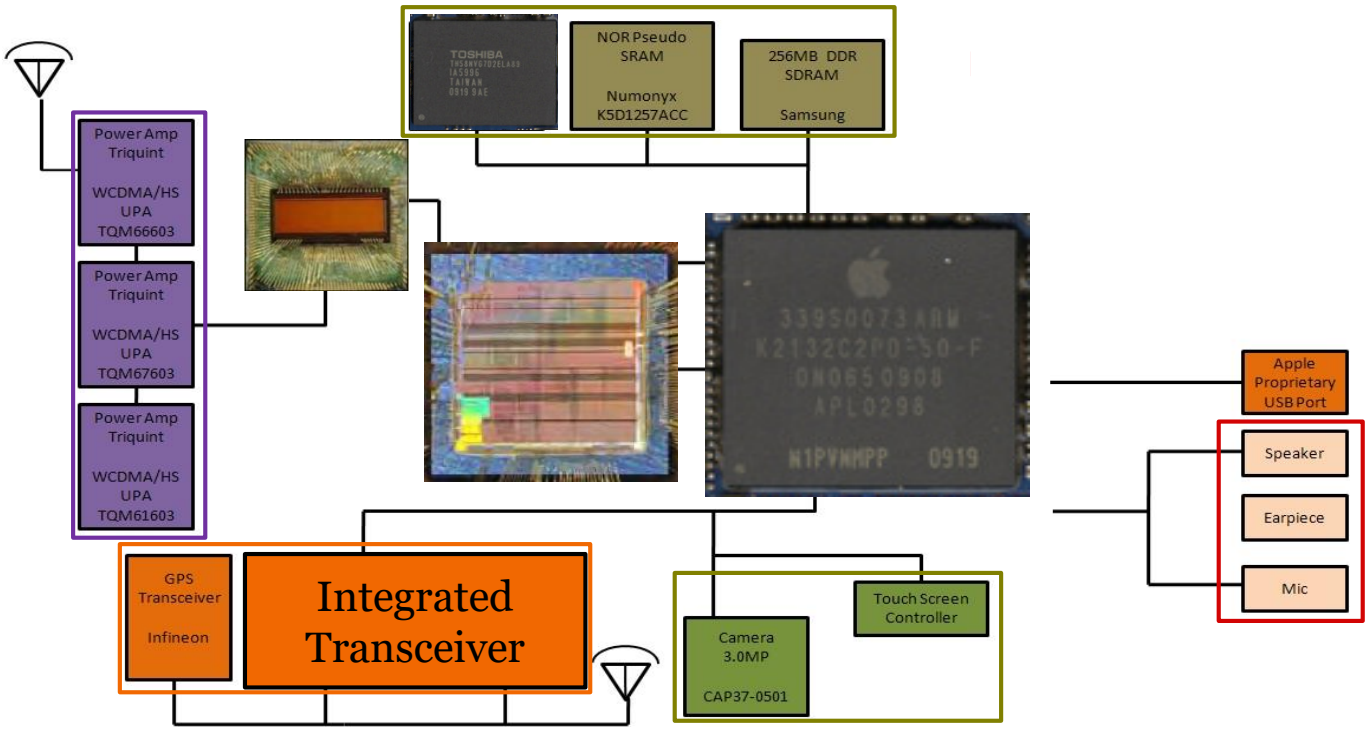
Trends in Cellphone Chip Integration



- Chip integration is increasing every generation
  - Cell phone size is decreasing
- Users want more features every generation
- Power budget is very limited

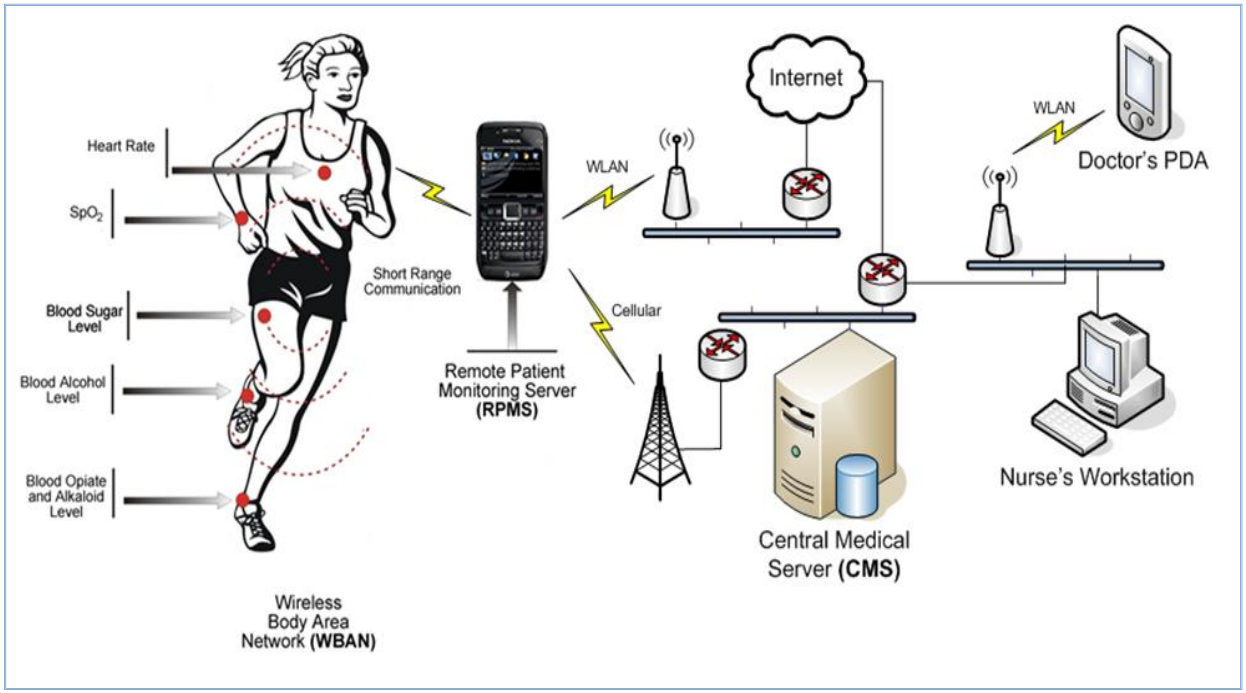
Y. Neuvo, ISSCC 2004

Cellphone Architecture Example



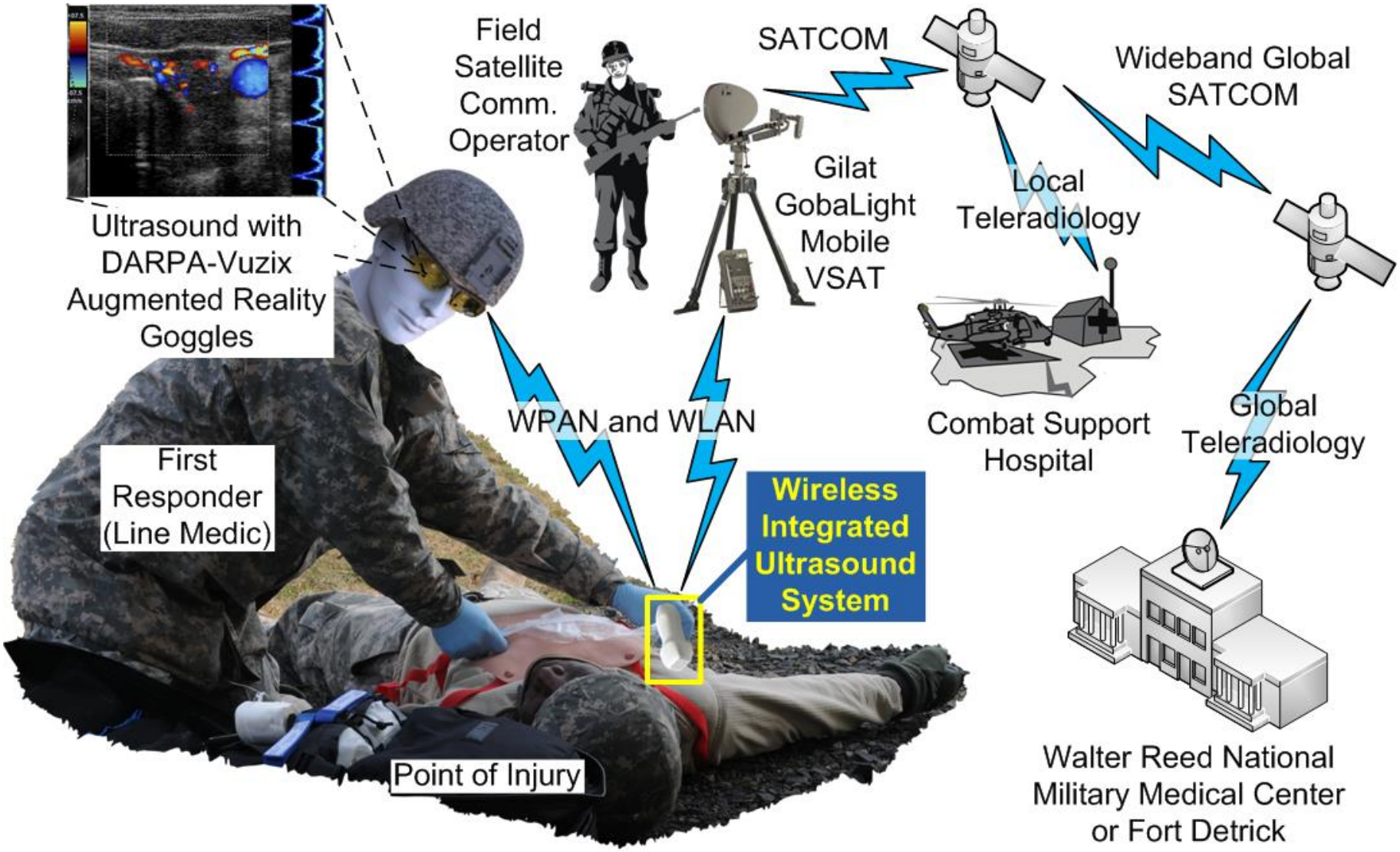
- Cellphone chips have multiple processing cores and support multiple applications and features
  - Ex: Integrated Transceiver: WiFi (802.11a/b/g), Bluetooth, FM

Smart Health Monitoring: Analysis & Delivery



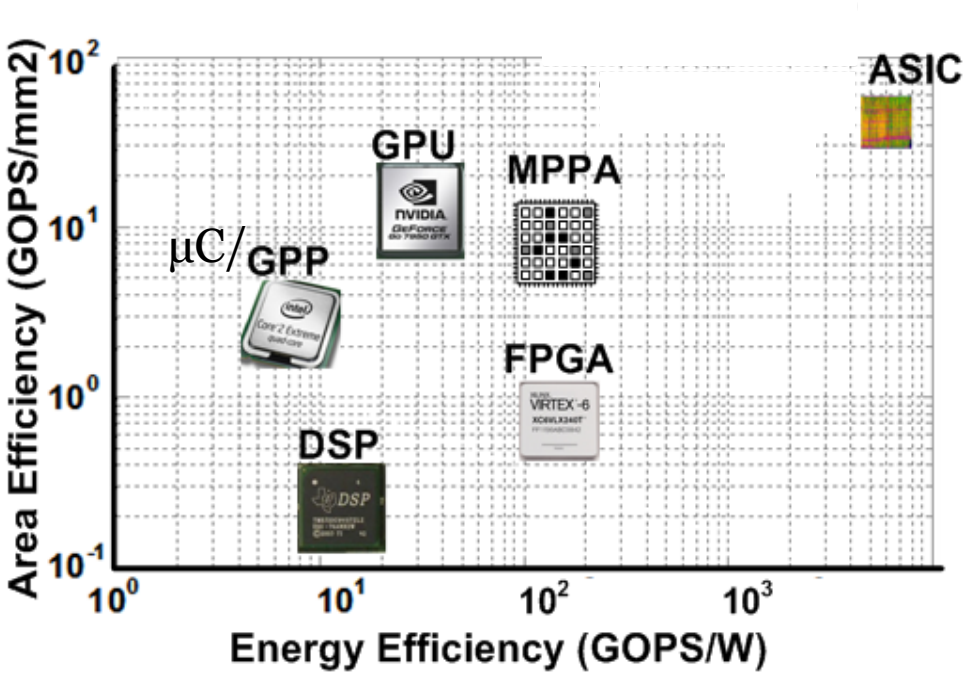
- Wearable medical monitoring systems
  - Reliable and seamless monitoring integrated into patients daily life routine
- Data analysis
  - Real-time data analysis and diagnosis for efficient healthcare delivery
- Data delivery
  - Real time data transmission to healthcare providers (e.g. nurses, primary care physicians, and first responders) through networks and immediate therapy through smart drug delivery

Military & Aerospace Telemedicine

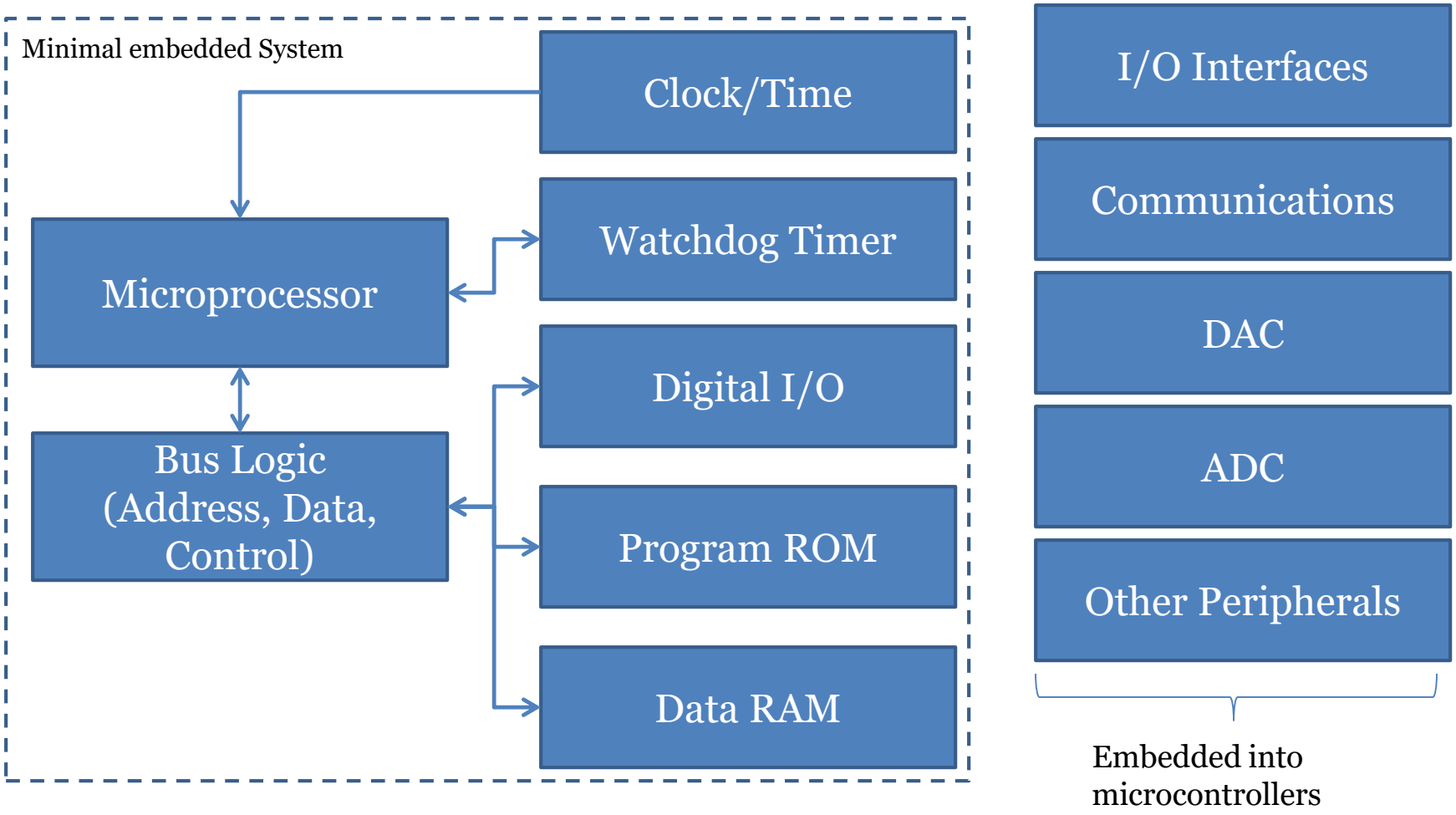


Key Objectives

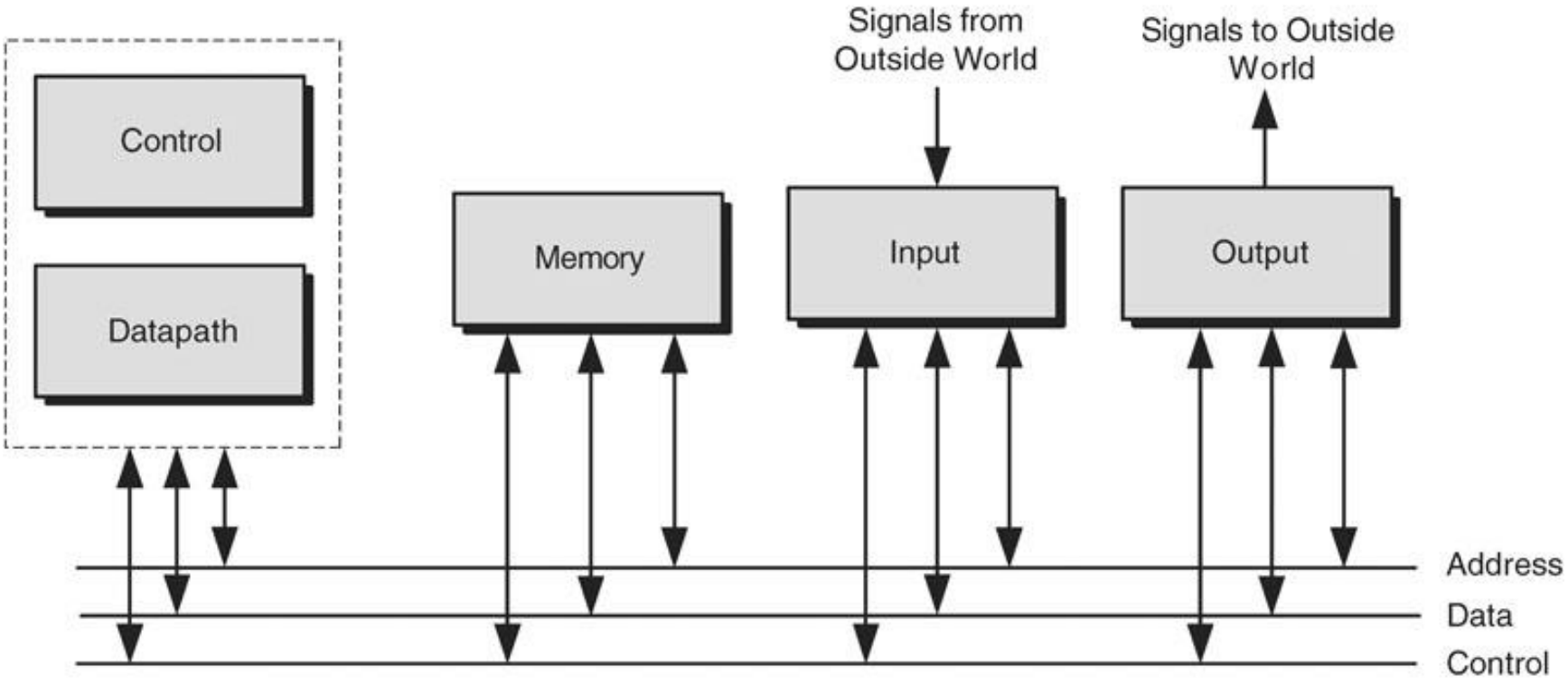
- High performance: 10-100 GOPS
- Energy efficiency: 100-1000 GOPS/W
- Area efficiency: 10-100 GOPS/mm²
- Programmability



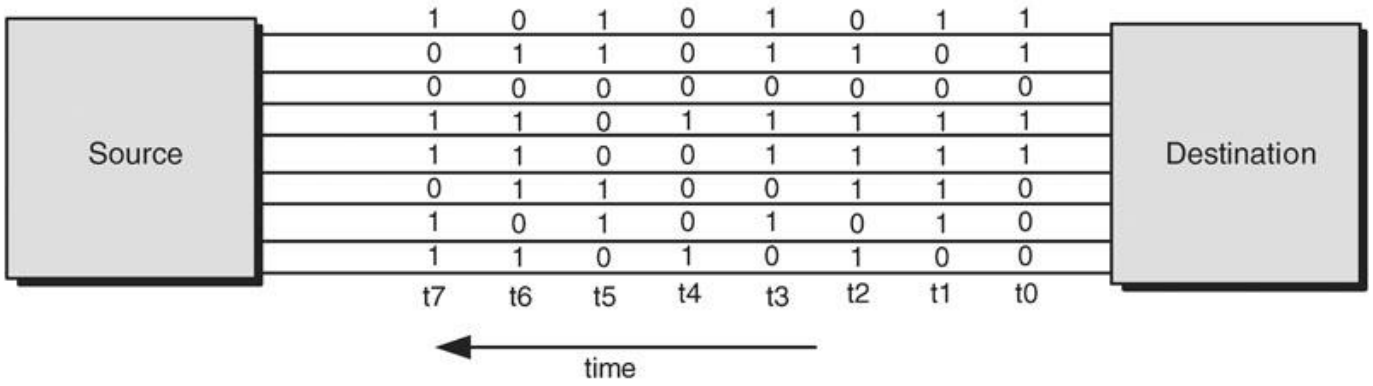
Microprocessor Based Embedded Systems



Typical BUS structure comprising Address, Data and control signals



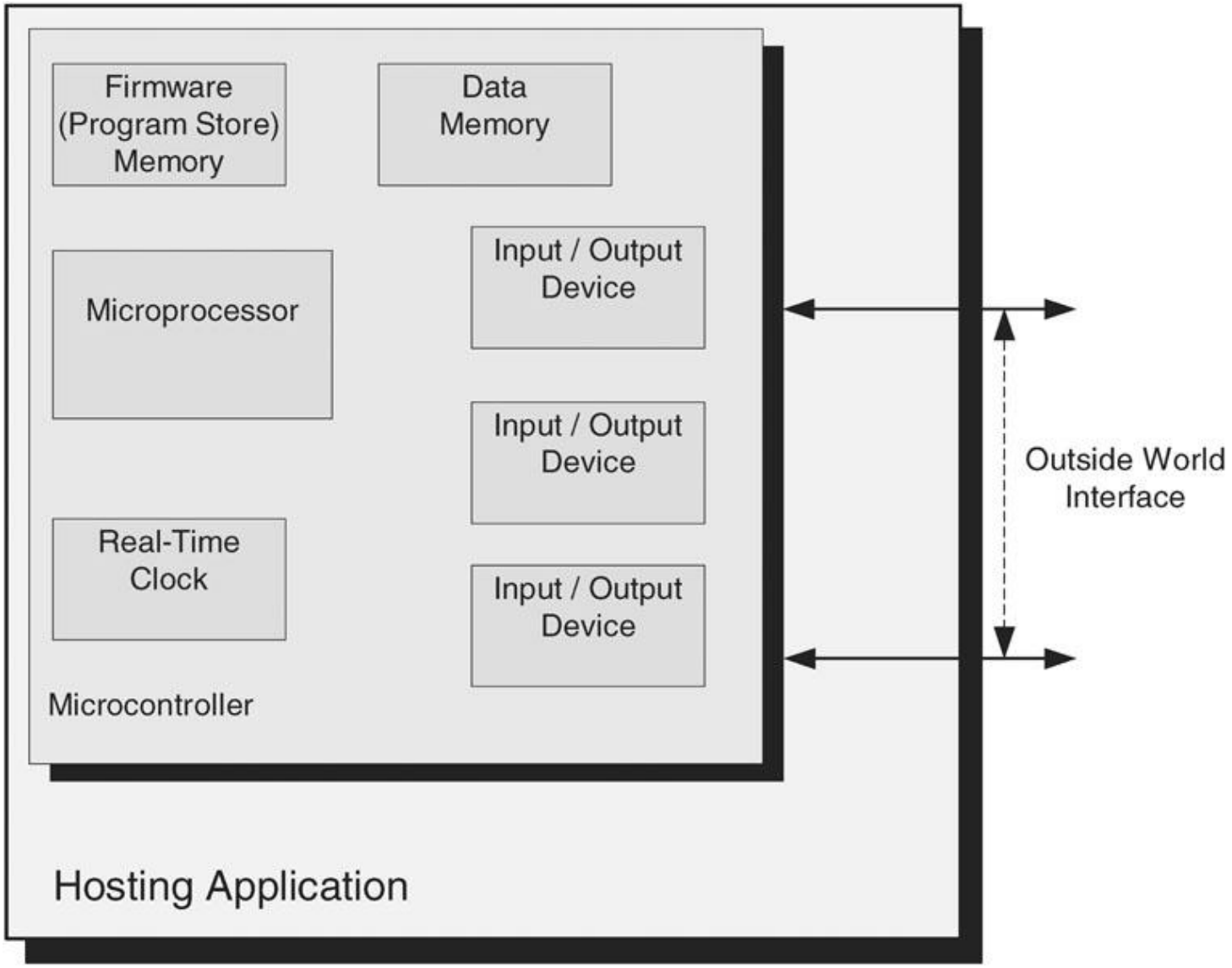
Data movement over an 8-bit Bus



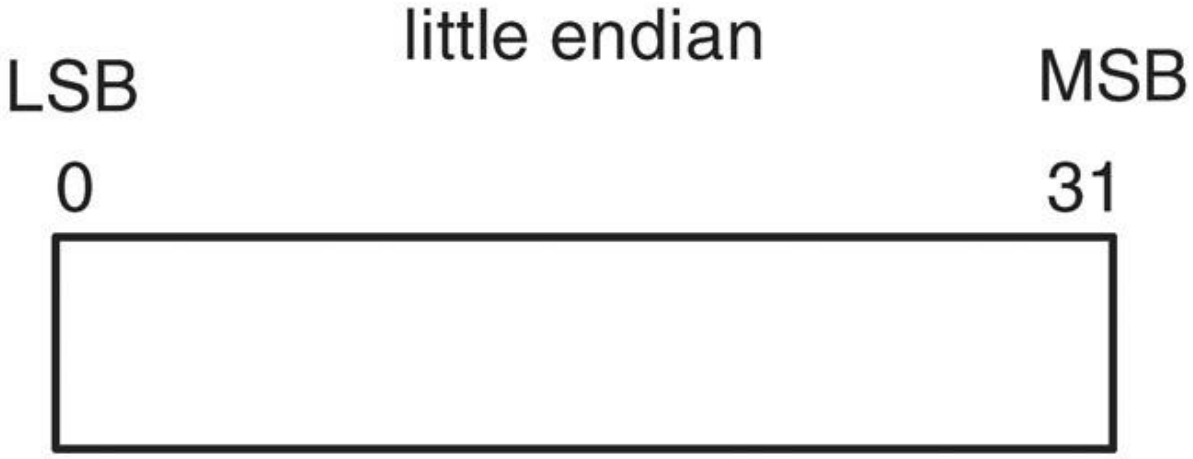
Example:

```
For (i=0; i<8; i++)
{
   printf ("%i", a[i]);
}
```

Microcontroller block diagram



Big Endian
In big endian, you store the most significant byte in the smallest address

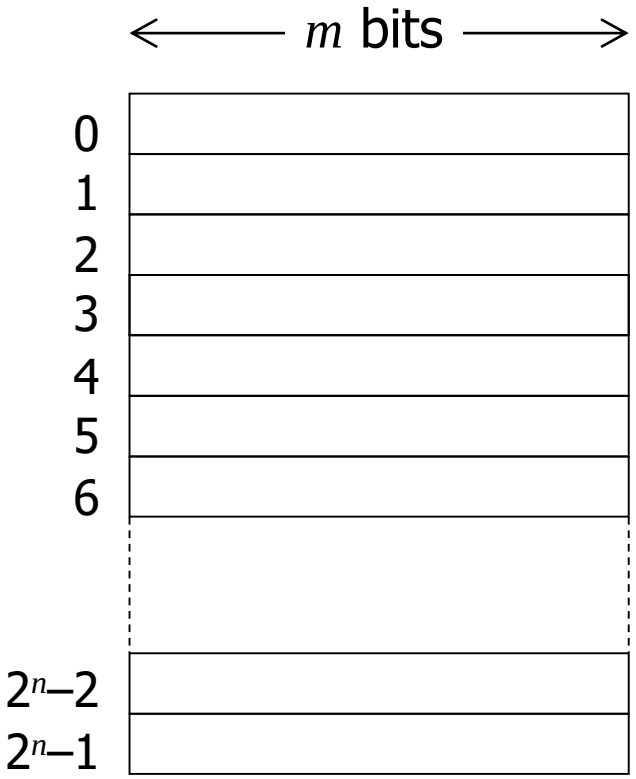


Little Endian
In little endian, you store the *least* significant byte in the smallest address

fig_01_11

Memory-General Concepts

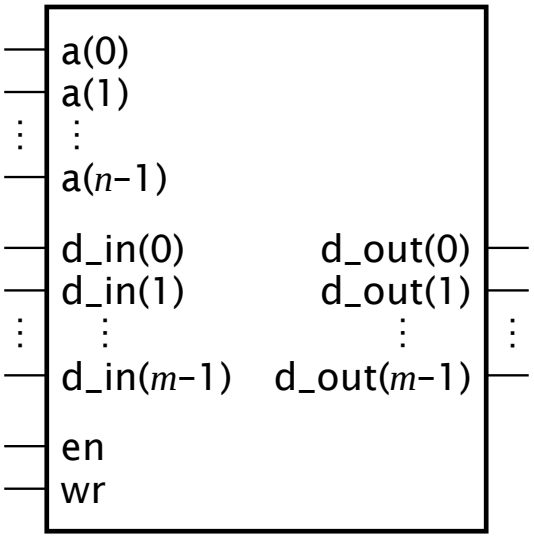
- A memory is an array of storage locations
  - Each with a unique address
  - Like a collection of registers, but with optimized implementation
- Address is unsigned-binary encoded
  - n address bits ⇒ 2^n locations
- All locations the same size
  - $2^n \times m$ bit memory



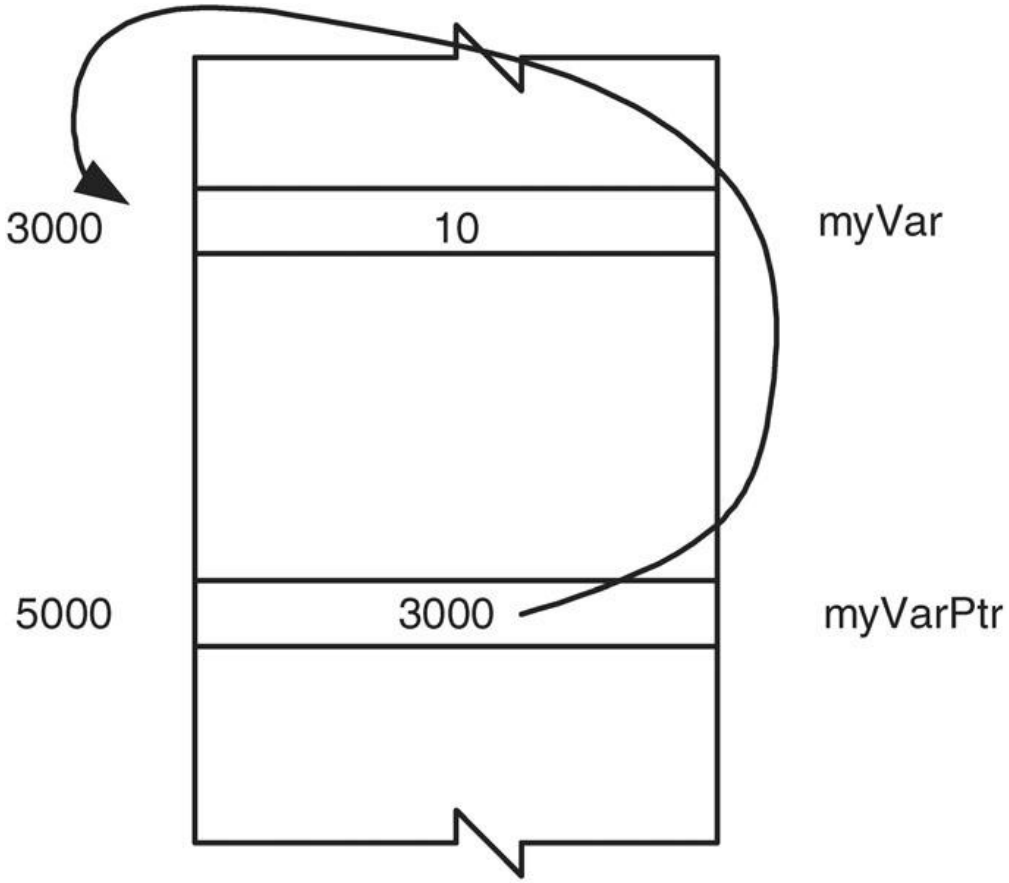
Memory Sizes

- Use power-of-2 multipliers
  - Kilo (K): $2^{10} = 1{,}024 \approx 10^3$
  - Mega (M): $2^{20} = 1{,}048{,}576 \approx 10^6$
  - Giga (G): $2^{30} = 1{,}073{,}741{,}824 \approx 10^9$
- Example
  - 32K × 32-bit memory
  - Capacity = 1,024K = 1Mbit
  - Requires 15 address bits
- Size is determined by application requirements

Basic Memory Operations



- a inputs: unsigned address
- d_in and d_out
  - Type depends on application
- Write operation
  - en = 1, wr = 1
  - d_in value stored in location given by address inputs
- Read operation
  - en = 1, wr = 0
  - d_out driven with value of location given by address inputs
- Idle: en = 0



The example places the integer value 10 in binary into some location e.g address 3000

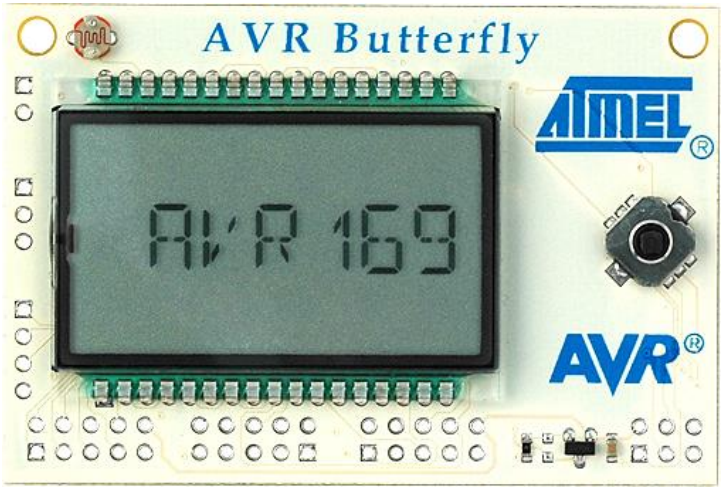
```
Int myVar=10
Int* myVarPtr=&myVar // take the
//address of myVar assign it to the
//pointer variable myVarPtr
```

When interpreted by the system, the code directs the system to set aside another memory word to hold the address

Why Microcontrollers?

- Peripheral loaded
  - ADC, DAC, GPIOs, Serial Interfaces
- Cheap
  - ~$1 for 8-bit processor
- Relatively Simple and Low Power
  - ~300μA operation (1 AA battery for 275 days, depending on the application)
  - <1μA sleep (1 AA battery for 225 years)
- Programmable
  - Assembly or C

Our Microcontroller



- AVR Butterfly
  - ATMEGA 169PV chip
  - Built-in peripherals
    - 120 segment LCD Screen
    - Joystick
    - Piezo element – sounds
- Programmer
  - AVR Dragon
- These boards will be used for projects and discussions