Principles and Practices of Microcontroller (Embedded System Design I) -MCU Basics

Gang Chen (陈刚)

Associate Professor

Institute of Unmanned Systems
School of data and computer science
Sun Yat-Sen University

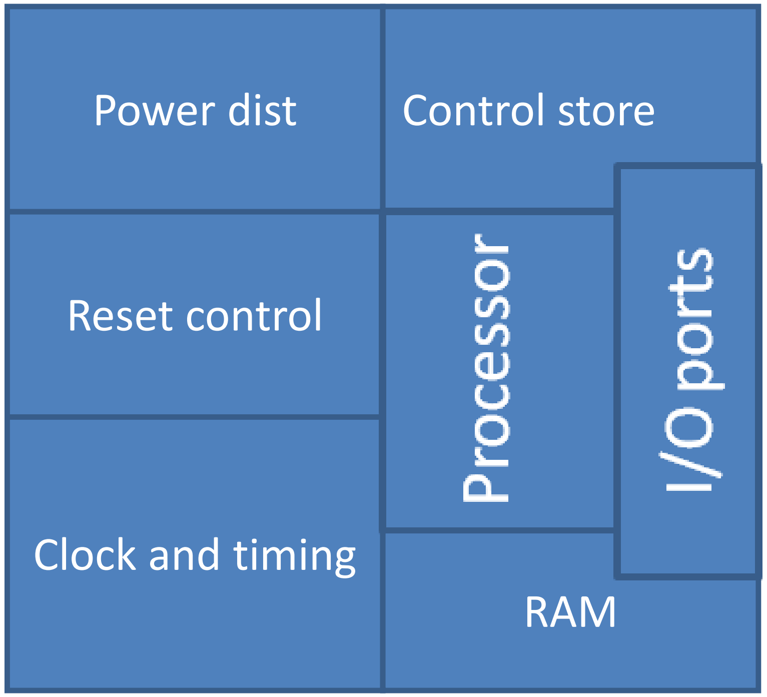
https://www.usilab.cn/team/chengang/

Recap of the last lecture

- **What distinguishes embedded systems?**
  - Application-specific
  - Resource-constrained
  - Real-time operations
  - Software runs “forever”
- **Technology scaling is driving “embedded everywhere”**
  - Microprocessors
  - Memory (RAM and Flash)
  - Imagers and MEMS sensors
  - Energy storage
- **Embedded platforms and software**
  - How does a phone boot? HW, SW, INT, and drivers
  - What are the components of a DSL modem? 18 major parts
  - Why the ARM architecture? 90%+ of 32-bit embedded CPUs
  - How do ARM licensees differentiate products? Peripherals

Microcontroller (MCU)

- **An embedded microcontroller is a chip which is a computer processor with all it’s support functions (clocking and reset), memory, and i/O built into the device.**

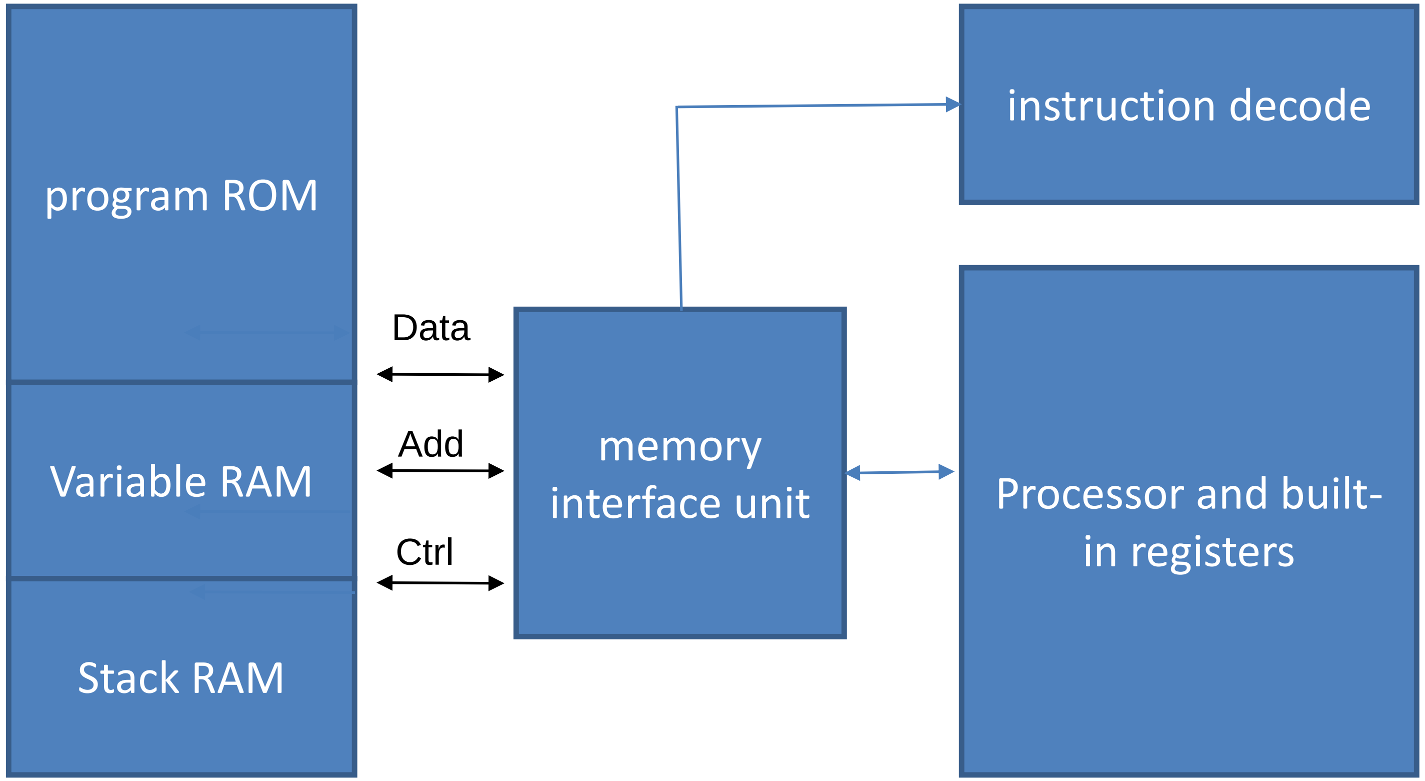


Processor Architecture

- **Harvard and Princeton**
  - Princeton provided ‘Von Neumann’ architecture where common memory space are used for storing program and data. Memory unit is responsible for arbitrary access to memory space between reading instructions and passing data back and forth with processor and its internal registers.
    - Advantages: simple memory interfacing and management.
  - Harvard proposes a design that used separate memory banks for program storage, the processor stack, and variable RAM.
    - Advantage: execute instruction in fewer cycles than Von Neumann

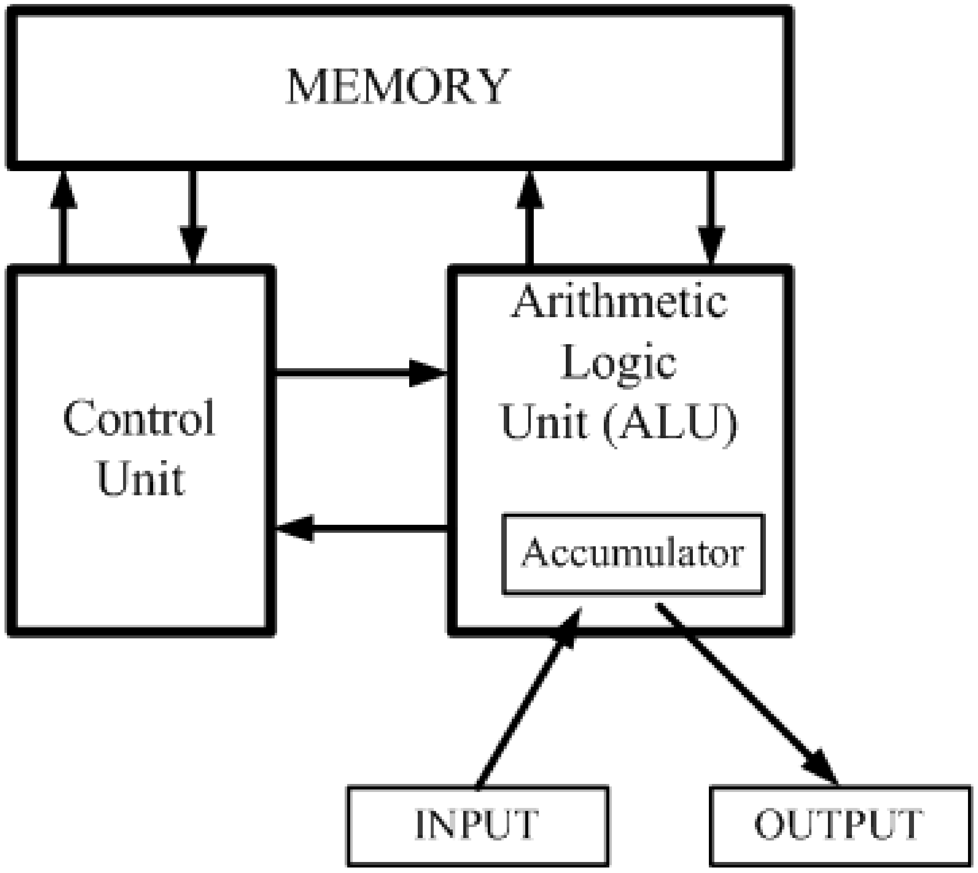
Princeton architecture block diagram

- **Also called von Neumann architecture**



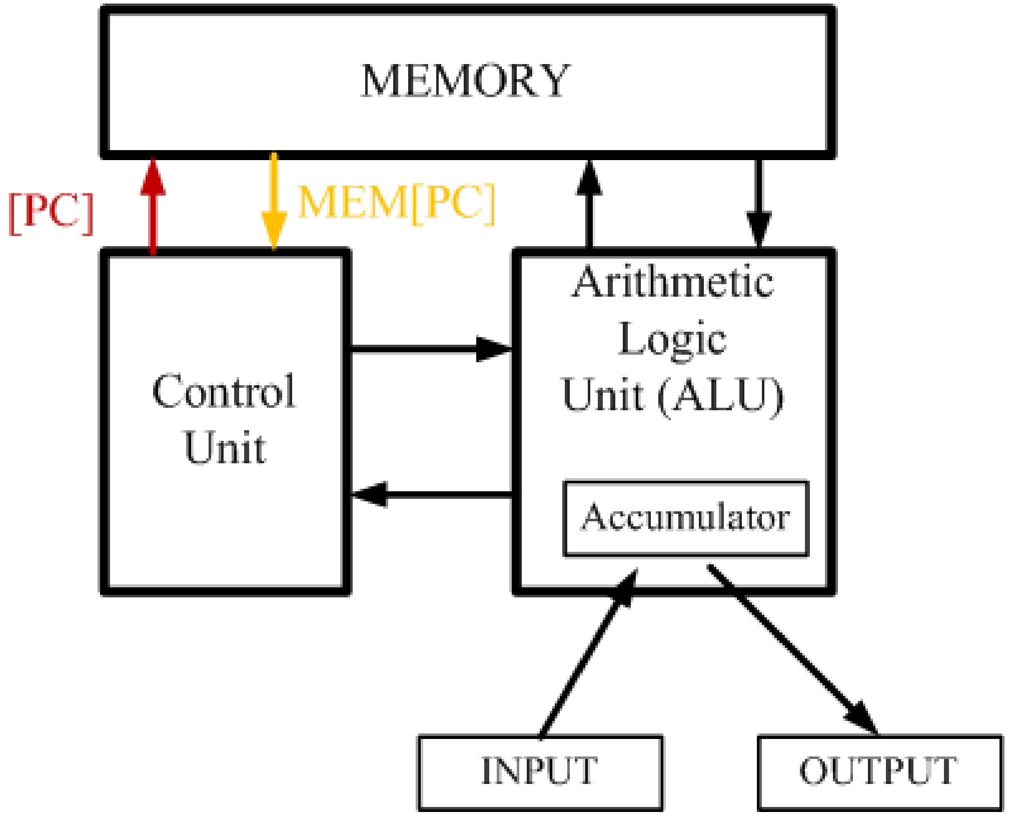
The von Neumann architecture – 1940s and 50s

- **The von Neumann architecture – 1940s and 50s**
    - A stored-program computer that uses a central processing unit and a single separate storage structure that hold both instructions and data.



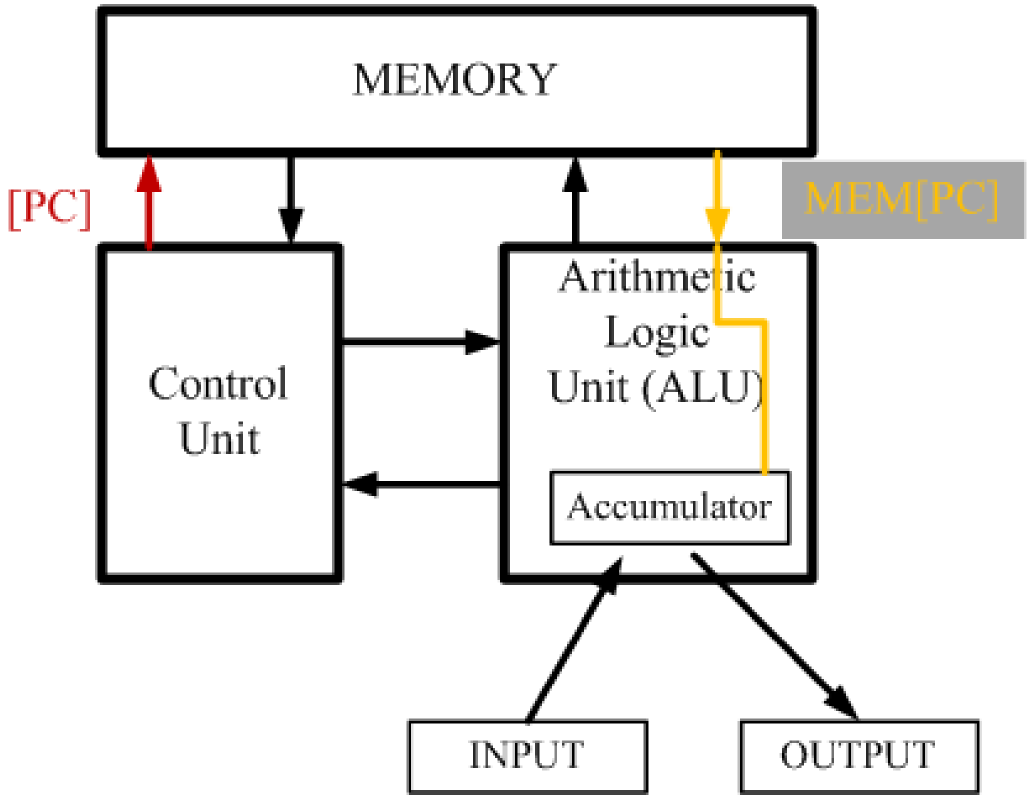
Basic operation of architecture

- **Instructions are executed in sequence**
- **First step during execution**
  - MEM(PC) ➔ IR
    - ◘ Send contents of PC (Program counter) to memory
    - ◘ Memory responds with the contents at that address placing it on the data bus.
  - Increment the PC (PC+1->PC)
  - The values on the data bus are loaded into the instruction register



Decode Instruction and execute

- **Say the instruction was a load immediate**
- **This means that the next word in the instruction stream is the data that we want loaded into the accumulator**
- **Operation is now**
  - MEM(PC)→ Accum
  - Also increment the PC



More von Neumann

- Earliest computers had fixed programs – such as a desk calculator
- The von Neumann architecture introduced the concept of a stored program. In fact, in early computers, they often wrote programs that self modified.
- Self-modifying code is now seen as a very bad programming practice (also, it really isn’t needed).
- von Neumann’s was very familiar with Alan Turing’s (1912-1954) work – the Turing Machine (1936).
- Both von Neumann and Turing wrote papers on stored program computers.

Early Microprocessors

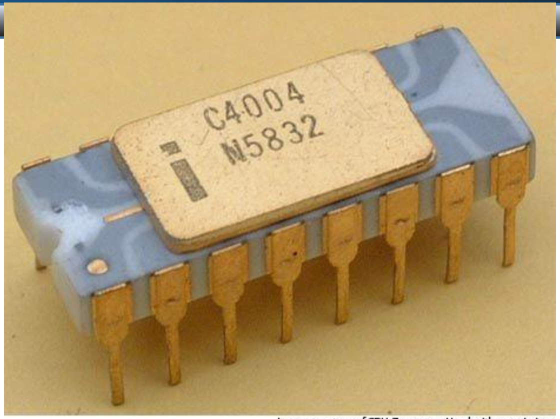
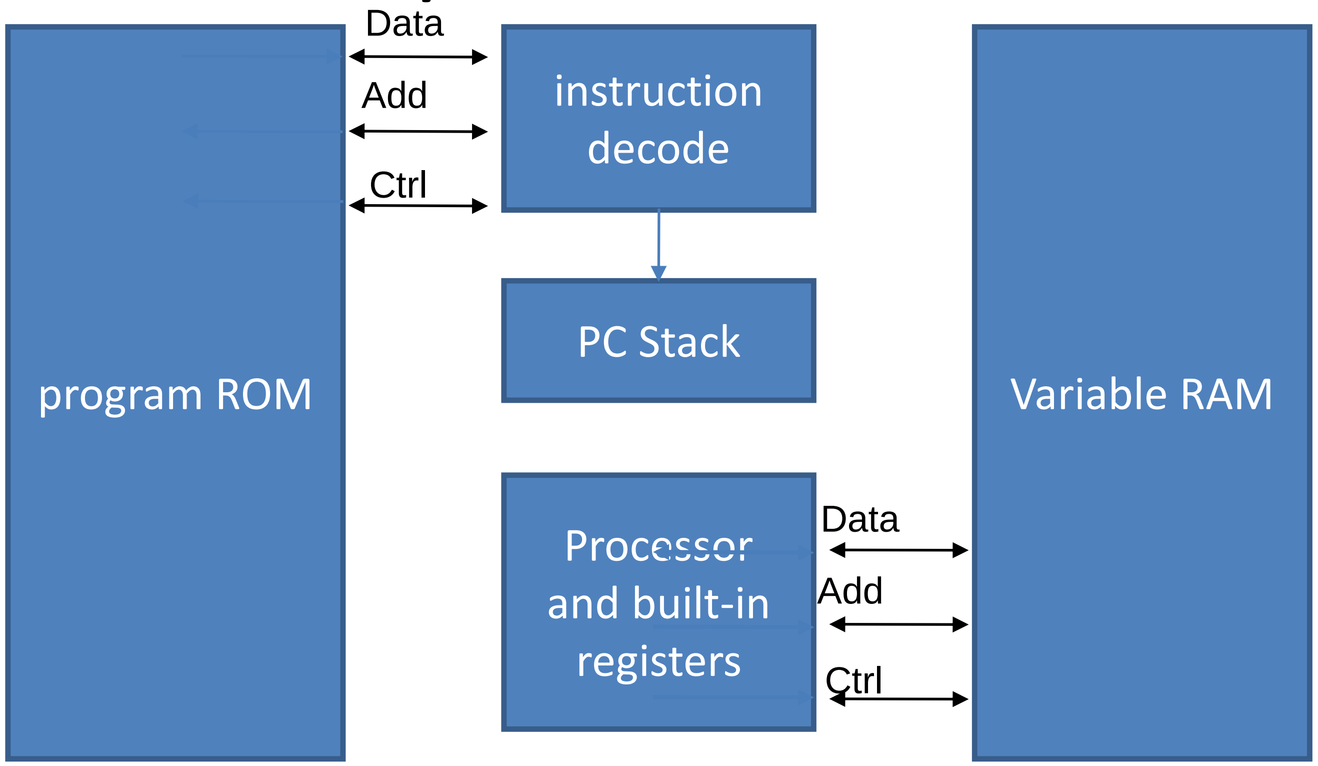


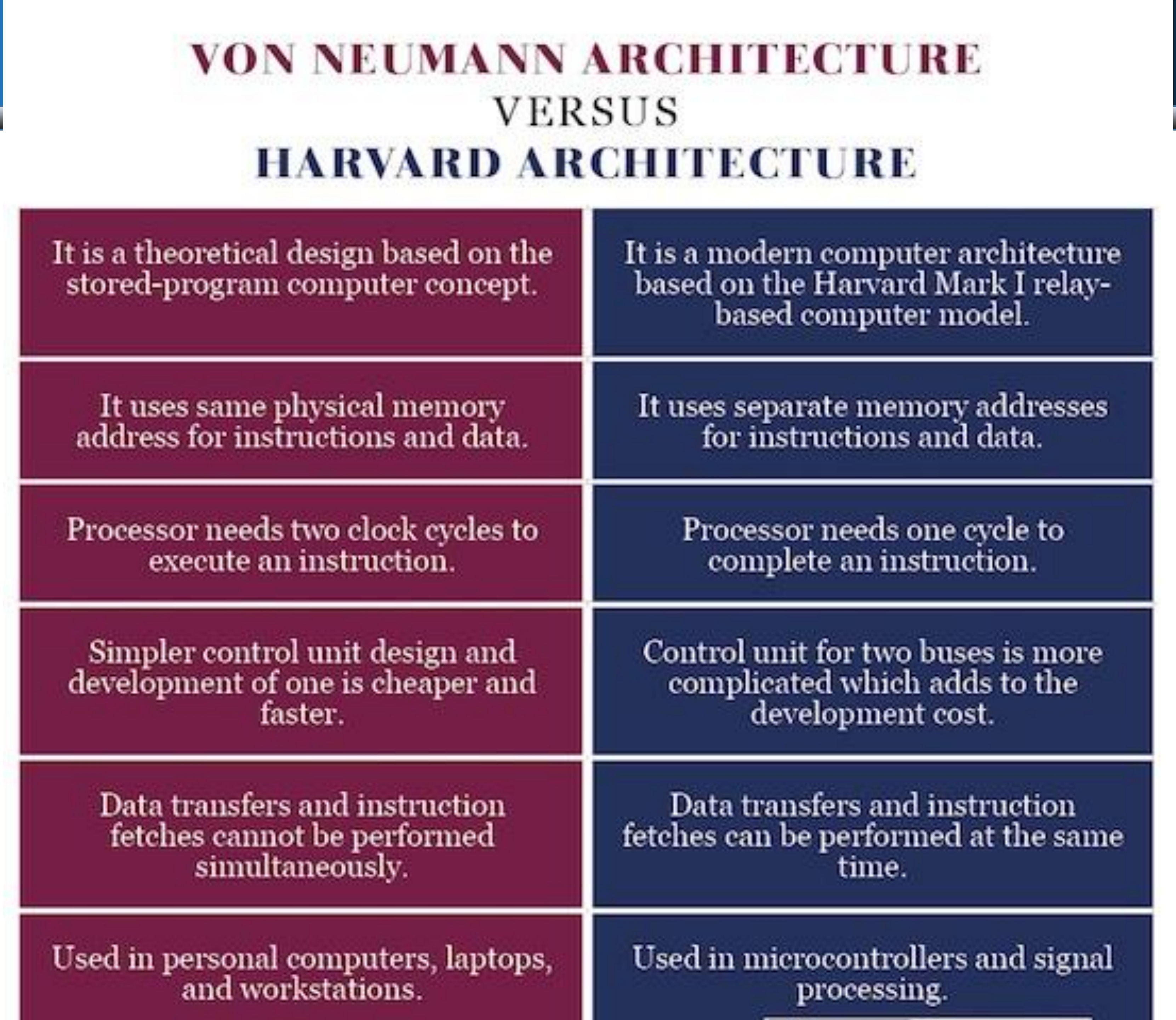
Image courtesy of CPU-Zone.com. Used with permission.

- The Intel 4004 – 1971
- 16-pin DIP package
- 92,000 instructions per sec
  - 10.8 microseconds per instruction
- Processor had a small address space for data and a small address space for instructions
- Designed for use in calculators
- Was the core element for the early electronic calculators – early calculators did basic arithmetic.
- Early microprocessors were often programmed in assembler or machine code. Compilers and many modern high level programming languages just didn’t exist.

Harvard architecture block diagram

- In the traditional von Neumann architecture memory holds both programs and data
- In the Harvard Architecture you have separate memory spaces for data and programs. (term that came into use during the late 1990s)



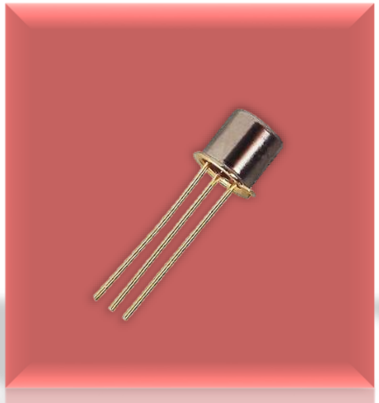


VON NEUMANN ARCHITECTURE VERSUS HARVARD ARCHITECTURE

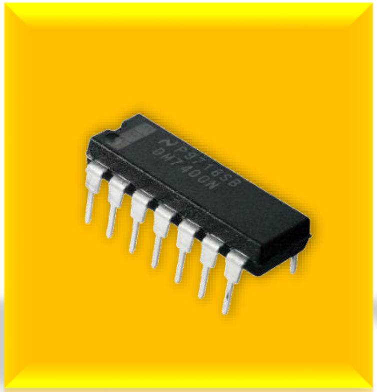
It is a theoretical design based on the stored-program computer concept.	It is a modern computer architecture based on the Harvard Mark I relay-based computer model.
It uses same physical memory address for instructions and data.	It uses separate memory addresses for instructions and data.
Processor needs two clock cycles to execute an instruction.	Processor needs one cycle to complete an instruction.
Simpler control unit design and development of one is cheaper and faster.	Control unit for two buses is more complicated which adds to the development cost.
Data transfers and instruction fetches cannot be performed simultaneously.	Data transfers and instruction fetches can be performed at the same time.
Used in personal computers, laptops, and workstations.	Used in microcontrollers and signal processing.

MCU History

Vacuum Tube
1939



Transistor
1947



Logic Gate
1960

Microcontroller
1971

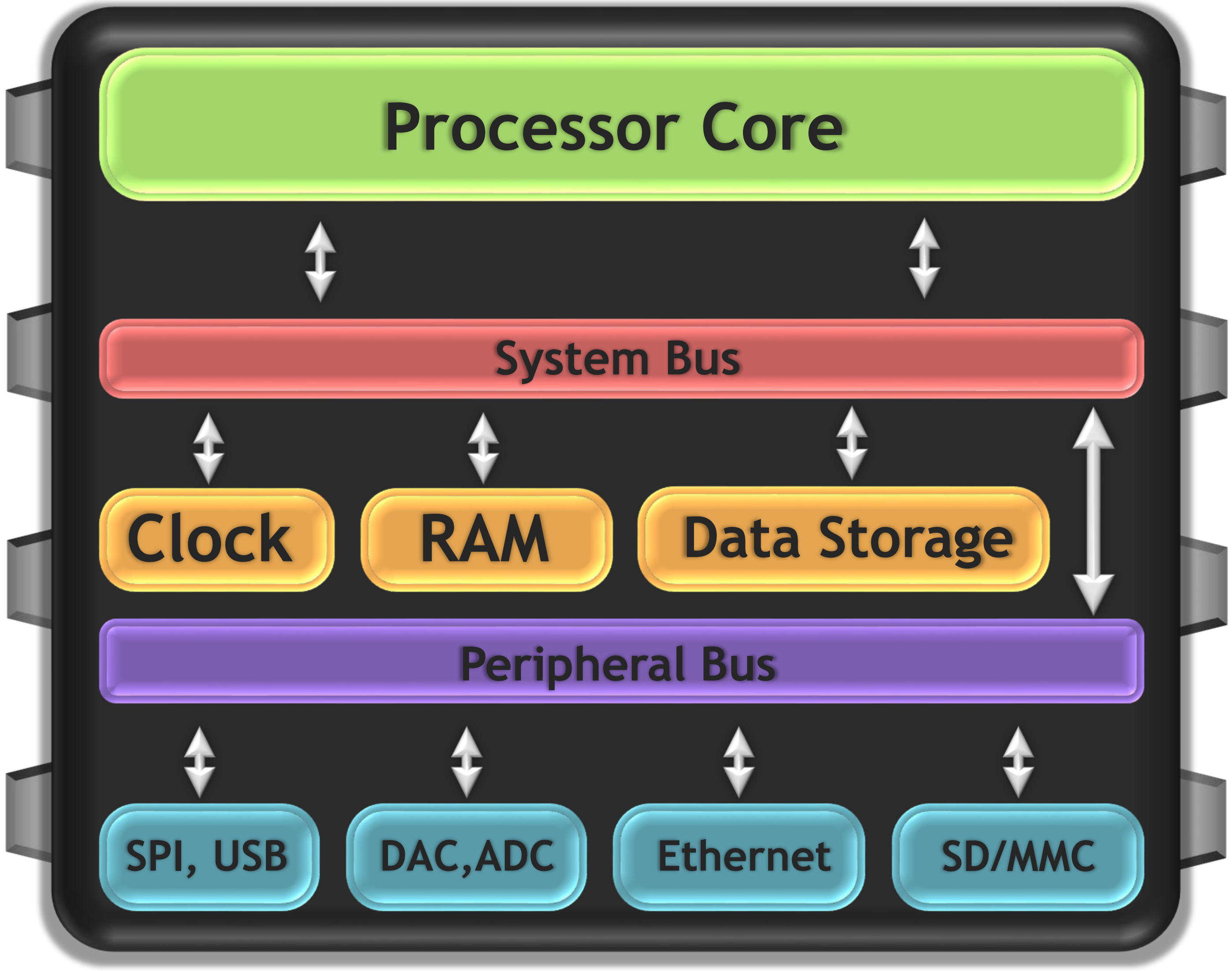
Microprocessor –vs- Microcontroller

- You may have heard of the term “microprocessor” or just “processor” before.
- You may ask, “Is there a difference between a microprocessor and microcontroller?”
- Yes there is, they are two different things!

Microprocessor –vs- Microcontroller

	Microprocessor	Microcontroller
Applications	General computing (i.e. Laptops, tablets)	Appliances, specialized devices
Speed	Very fast	Relatively slow
External Parts	Many	Few
Cost	High	Low
Energy Use	Medium to high	Very low to low
Vendors	intel AMD ARM	ATMEL ST Texas Instruments Microchip

Microcontroller Overview



Basic Principles of Operation

- Microcontrollers are used for specific applications.
- They do not need to be powerful because most applications only require a clock of a few MHz and small amount of storage.
- A microcontroller needs to be programmed to be useful.
- A microcontroller is only as useful as the code written for it. If you wanted to turn on a red light when a temperature reached a certain point, the programmer would have to explicitly specify how that will happen through his code.

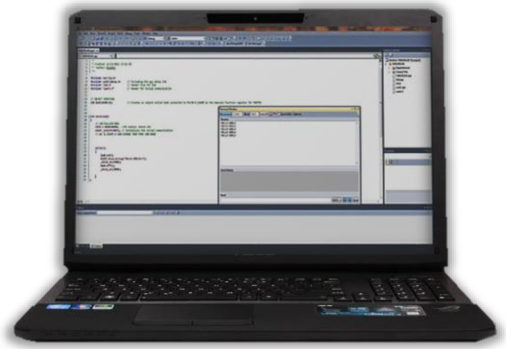
Microcontroller Programming

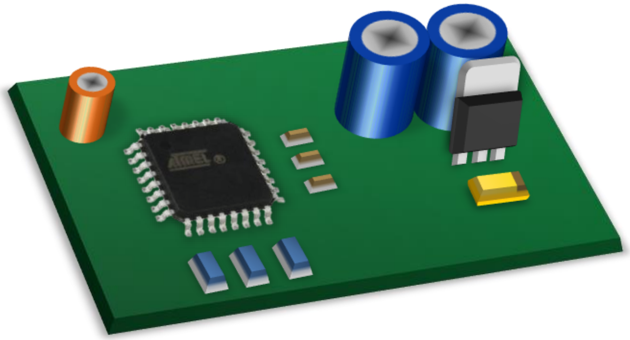
- Code is written for the microcontroller in an integrated development environment, a PC program. The code is written in a programming language. (e.g. C, BASIC or Assembly).

- The IDE debugs the code for errors, and then compiles it into binary code which the microcontroller can execute.

- A programmer (a piece of hardware, not a person) is used to transfer the code from the PC to the microcontroller. The most common type of programmer is an ICSP (In-circuit serial programmer).

Microcontroller Programming





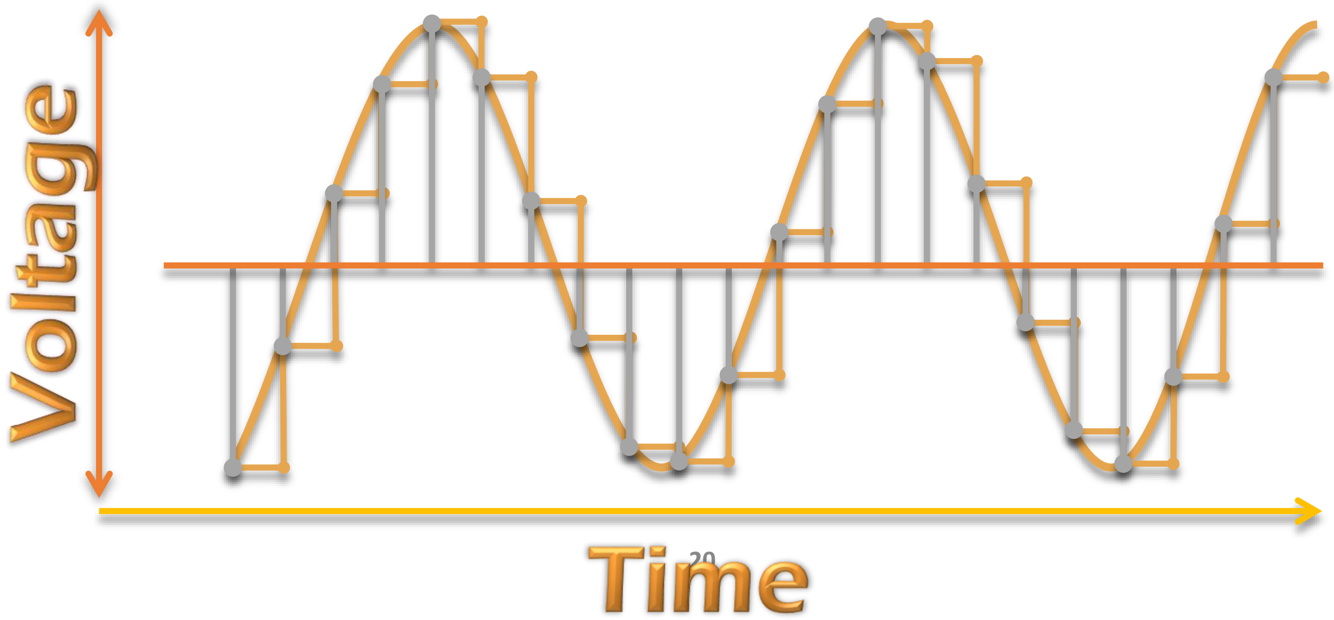
Presto!

MCU Basics

- **Digital and Analog Circuit**
  - Digital Circuit
    - only 0 and 1 logic
    - Driven by synchronized clock
  - Analog Circuit: continuous signal
- **MCU is digital circuit**
  - GPIO
    - 1 stands for VCC, e.g., 5V or 3.3V
    - 0 stands for GND, i.e., 0V
  - Require crystal oscillator
    - As synchronized clock
- **How about analogy signal?**

The Analog to Digital Converter (ADC)

- Just about every modern microcontroller contains an ADC(s).
- It converts analog voltages into digital values.
- These digital representations of the signal at hand can be analyzed in code, logged in memory, or used in practically any other way possible.



Microcontroller Applications

- This is the controller board for a washing machine. If a button is pushed or if a knob is turned, the microcontroller knows how to react to the event.

- Ex. If "start" is pushed, the microcontroller knows to switch a relay which starts the motor.

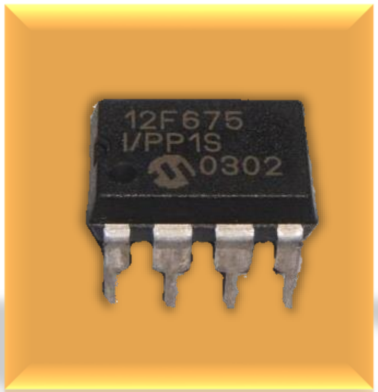
Microcontroller Applications

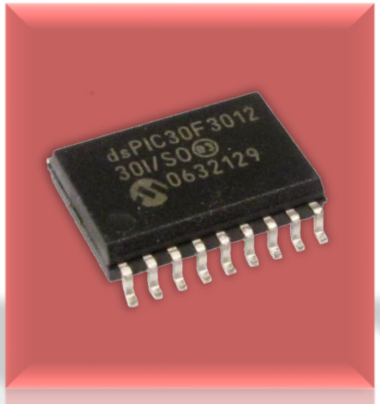
- Many robots use microcontrollers to allow robots to interact with the real world.

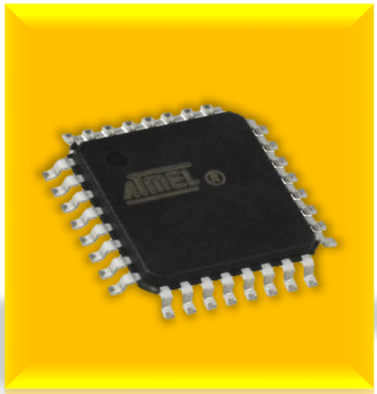
- Ex. If a proximity sensor senses an object near by, the microcontroller will know to stop its motors and then find an unobstructed path.

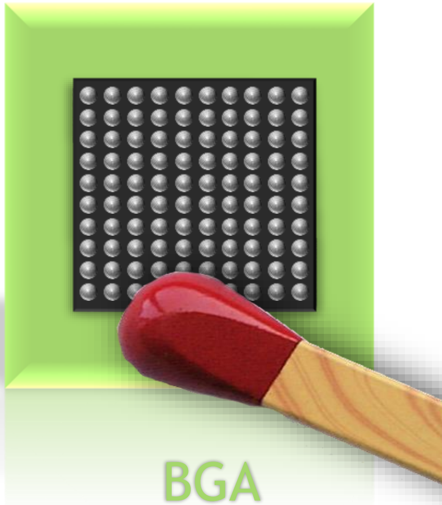
- Motor control and Communication

Microcontroller Packaging









DIP	SOIC	QFP	BGA
(Dual Inline Package)	(Small Outline IC)	(Quad Flat Package)	(Ball Grid Array)
Through hole	Surface Mount	Surface Mount	Surface Mount
8 pins	18 pins	32 pins	100 pins
9mm x 6mm	11mm x 7mm	7mm x 7mm	6mm x 6mm
0.15pins/mm^2	0.23pins/mm^2	0.65pins/mm^2	2.78pins/mm^2

Couse Share Folder

https://pan.baidu.com/s/1OxsPGJfzmDvwedKuzNc5AA

Binary number system

- Number systems
- To and from base 10
- Addition
- Subtraction (made easy)
- Multiplication

We live in a base 10 world

- **Why base 10?**
  - Could have been base 5 or base 20.
  - We can thank Ug! the caveman.
- **In base 10 we have 10 symbols**
  - 0 1 2 3 4 5 6 7 8 9
- **In any number base system you have n symbols**
  - Base 2 – 0 1
  - Base 8 - 0 1 2 3 4 5 6 7
  - Base 16 – 0 1 2 3 4 5 6 7 8 9 A B C D E F

Other number bases

- Number system base
- <u>Base 10</u>
- 0 1 2 3 4 5 6 7 8 9 10 11 12 13
- <u>Base 2</u>
- 0 1 10 11 100 101 110 111 1000 1001 1010

- <u>Base 5</u> (would have digits 0 to 4)
- 0 1 2 3 4 10 11 12 13 14 20 21 22 23
- Base 8 (octal)
- 0 1 2 3 4 5 6 7 10 11 12 13 14 15

To and from base

- **Base 10 to binary (base 2)**
- **Number in Base 10**
- **19_{10} = ?**
- **Procedure (Integer division)**
  - Divide by 2 19/2 = 9 r 1
  - 9/2 = 4 r 1
  - 4/2 = 2 r 0
  - 2/2 = 1 r 0
  - 1/ 2 = 0 r 1

  So the binary of 19_{10} is 1 0 0 1 1
    - (In general for any number base you divide by the number system base and use the remainders)
- **More examples?**

To and from base

- How about 139_{10} = ?
- Again divine by 2 each time

number	div 2	remainder
139	69	1
69	34	1
34	17	0
17	8	1
8	4	0
4	2	0
2	1	0
1	0	1

- So have 1 0 0 0 1 0 1 1
- More Examples?

Base 2 to base 10

- In first example have 1 0 0 1 1
- In binary the digits have the following weight
- $\text{Value}_{10}=a_4*2^4+a_3*2^3+a_2*2^2+a_1*2^1+a_0*2^0$
- $\text{Value}_{10}=a_4*16+a_3*8+a_2*4+a_1*2+a_0*1$
- So here
- 1 0 0 1 1 = 1*16 +0*8 +0*4 +1*2 +1*1
-                = 19

2nd example

- **Had 1 0 0 0 1 0 1 1 (written msb to lsb)**
- **Value of positions is (lsb to msb) 1,2,4,8,16,32,64,128,256,512,…**
  - The powers of 2
- **Value of the number above**
- **= 128 + 8 + 2 + 1**
- **= 139**

Binary addition

- **Follow the same rules as base 10**
  - carries -> 1 1 1 1
  - 1 1 0 0 1 　　　　0 0 1 1 1 (7)
  - 0 0 0 1 0 　　　　0 1 0 1 1 (11)
  - 1 1 0 1 1 　　　　1 0 0 1 0 (18)

- **More examples?**

Binary subtraction

- **Set the problem**

$$\begin{array}{rcccccc} & 1 & 0 & 0 & 0 & 0 & \quad 16 \\ - & 0 & 0 & 0 & 1 & 1 & \quad 3 \\ \hline \end{array}$$

- **And we need to borrow – step 1**

$$\begin{array}{rcccccc} & & 2 & & & & \\ & \cancel{1} & 0 & 0 & 0 & 0 & \quad 16 \\ - & 0 & 0 & 0 & 1 & 1 & \quad 3 \\ \hline \end{array}$$

Binary Subraction

- The next steps work through to

$$\begin{array}{rccccclr}
 & & 1 & 1 & 1 & & & \\
 & & 2 & 2 & 2 & 2 & & \\
 & 1 & 0 & 0 & 0 & 0 & & 16 \\
- & 0 & 0 & 0 & 1 & 1 & & 3 \\
\hline
\end{array}$$

- To allow answer to be done

$$\begin{array}{rccccclr}
 & & 1 & 1 & 1 & & & \\
 & & \cancel{2} & \cancel{2} & \cancel{2} & 2 & & \\
 & \cancel{1} & 0 & 0 & 0 & 0 & & 16 \\
- & 0 & 0 & 0 & 1 & 1 & & 3 \\
\hline
 & 0 & 1 & 1 & 0 & 1 & & \\
\end{array}$$

Binary multiplication

- Just like multiplication in base 10

```
          1 0 0 0 0       16
      x   0 0 0 1 1        3
    -----------------------
          1 0 0 0 0
        1 0 0 0 0   .
        -------------
        1 1 0 0 0 0       48
```

- More examples?

2’s complement

- **For 4 bits can represent values -8 to 7**

0	0 0 0 0	-0	0 0 0 0
1	0 0 0 1	-1	1 1 1 1
2	0 0 1 0	-2	1 1 1 0
3	0 0 1 1	-3	1 1 0 1
4	0 1 0 0	-4	1 1 0 0
5	0 1 0 1	-5	1 0 1 1
6	0 1 1 0	-6	1 0 1 0
7	0 1 1 1	-7	1 0 0 1
		-8	1 0 0 0

- **In general can represent**
  - $-2^{n-1} \leq n \leq 2^{n-1} - 1$

Arithmetic with 2's complement

- **Add 5 and -3**

```
   1   1
   0 1 0 1     5
   1 1 0 1    -3
 ---------------
   0 0 1 0
Carry out = 1
```

Finding the 2's complement

- **To generate the 2's complement of n**
- **Say n is 3**
  - 3 in binary (4 bits) is 0011
  - Procedure
  - 1) Take the 1's complement, then add 1
    - 1's complement – complement all bits
    - 0011 → 1100 +1 = 1101
  - 2) Starting at the lsb (rightmost) bit
    - Keep the 1st 1 and then complement the rest of the bits. Can easily see on previous example

Subtraction via 2’s complement

- 14 – 6
- (need 5 bits to represent in 2’s complement form)
- 01110 – 00110 or

- 01110 + 11010 (i.e. 14 + (-6))

- 01110
- <u>+11010</u>
- 01000 and a carry out of 1 value is 8

Operating in 2's complement

- **In general can represent**
  - $-2^{n-1} \leq n \leq 2^{n-1} - 1$
- **So with 4 bits can represent values of $-8 \leq n \leq +7$**
- **So with 8 bits can represent values of $-128 \leq n \leq +127$**
- **So with 16 bits can represent values of $-32768 \leq n \leq +32767$**

Link to MCU control

- **How to set one specific bit in a register?**
  - Turn on one LED in Led arrays
- **How to clear?**
- **How to read one specific bit in a register?**
- **Write C code now.**

Assignment 1

- **What is a Turing machine?**
- **Write up what a Turing machine is and how a Turing Machine executes a program.   Write 2/3 to 1 pages.**
- **Print it out and hand it to me in two weeks.**
- **There are many sources for this assignment**
  - Google web search
  - Wikipedia
  - Library
- **Send Emails: lingyh6@mail2.sysu.edu.cn**
- **CC: gangchen1170@foxmail.com**
- **With Subject:** 单片机课程第**X**次作业**+**姓名**+**学号