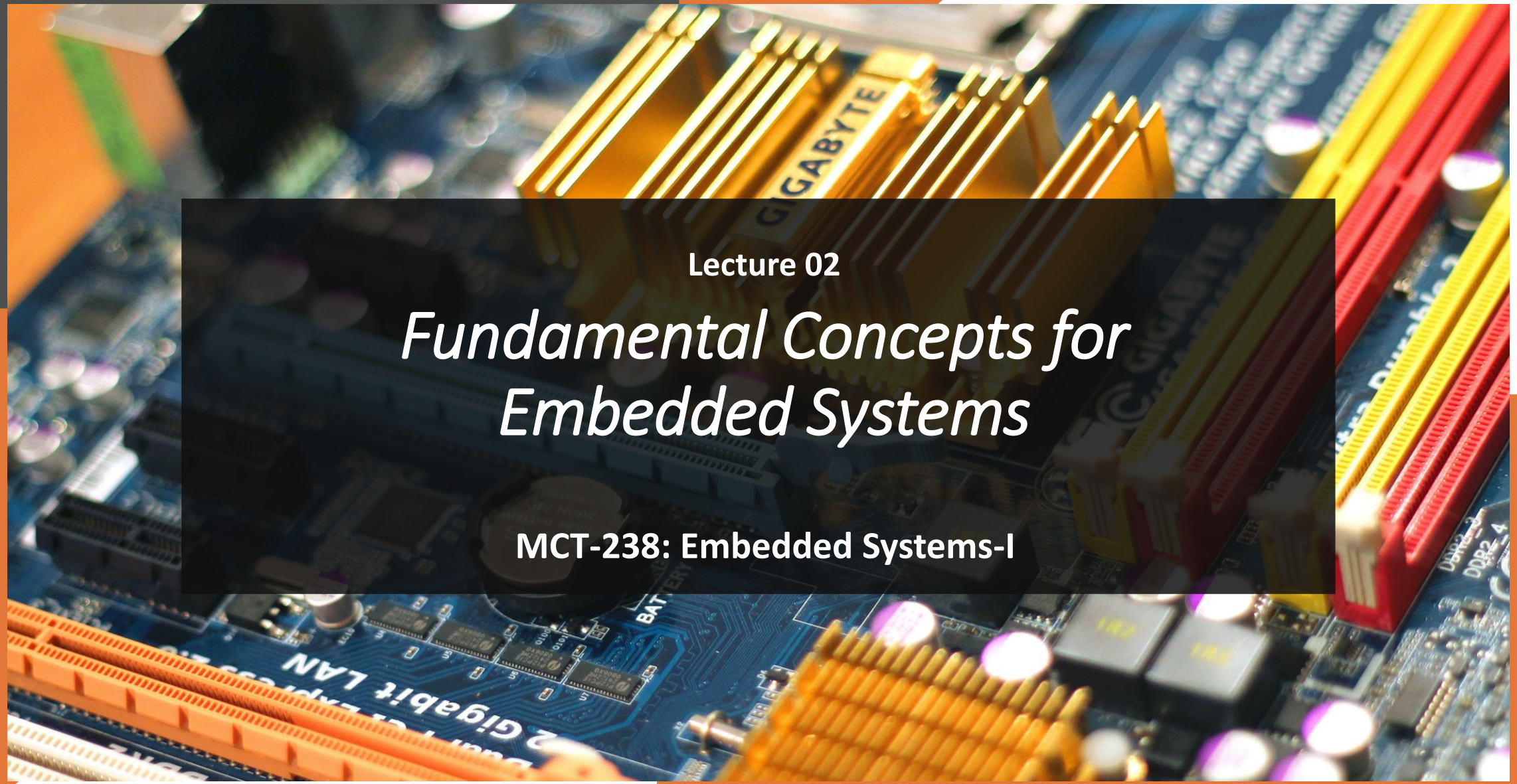
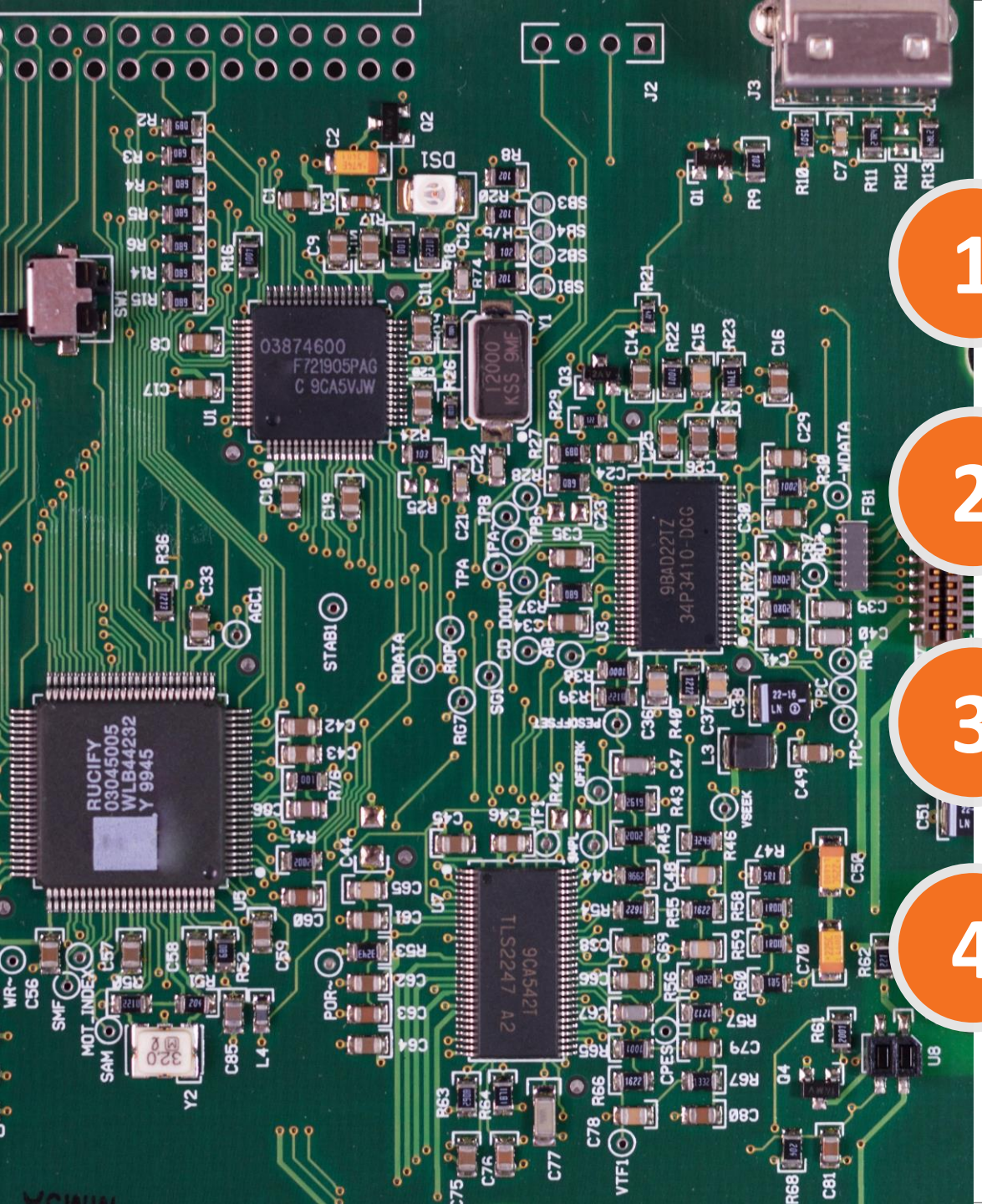


Lecture 02

Fundamental Concepts for Embedded Systems

MCT-238: Embedded Systems-I



1. BASIC ELECTRONICS REVIEW
2. DIGITAL INFORMATION AND LOGIC
3. DIGITAL REPRESENTATION OF NUMBERS
4. COMPUTER ARCHITECTURE

BASIC ELECTRONICS REVIEW

Electric Current, Voltage, Resistance, Ohm's Law, Electric Power, Energy, Switch, KCL, KVL, Current Divider, Voltage Divider

Electric Current, Voltage, Resistance

BASIC ELECTRONICS REVIEW

- **Electric Current** (***I***): flow of charge in unit time, measured in *ampere* (*A*)
  - 1 ampere (A) of current is 6.241×10^{18} electrons per second, or one coulomb per second
  - directional and measured at one point as the number of electrons travelling per second
  - has an amplitude and a direction
- **Voltage** (***V***): potential difference between two points, measured in *volt* (*V*)
  - the electromotive force or potential to produce current
- **Resistance** (***R***): opposition to flow of current through a media, measured in *ohm* (*Ω*)
  - Ideally, a conductor wire is simply a resistor with a resistance of 0 Ω
  - Normally, a good insulator has a resistance of order of 10^6 Ω
  - Color coded carbon resistors are most used

Ohm's Law, Electric Power, Energy

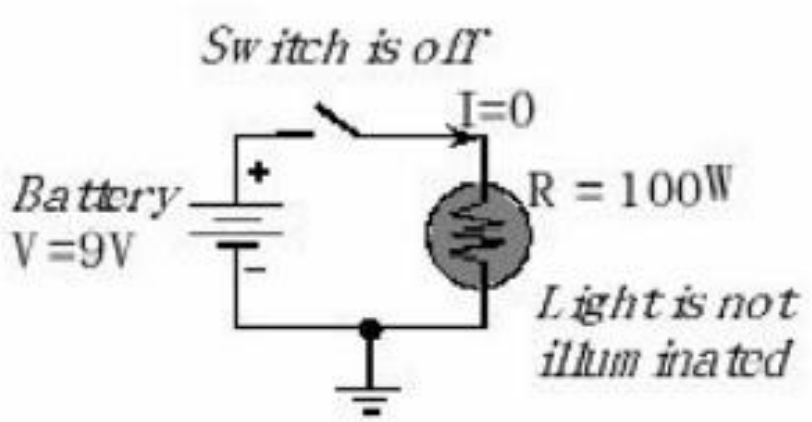
BASIC ELECTRONICS REVIEW

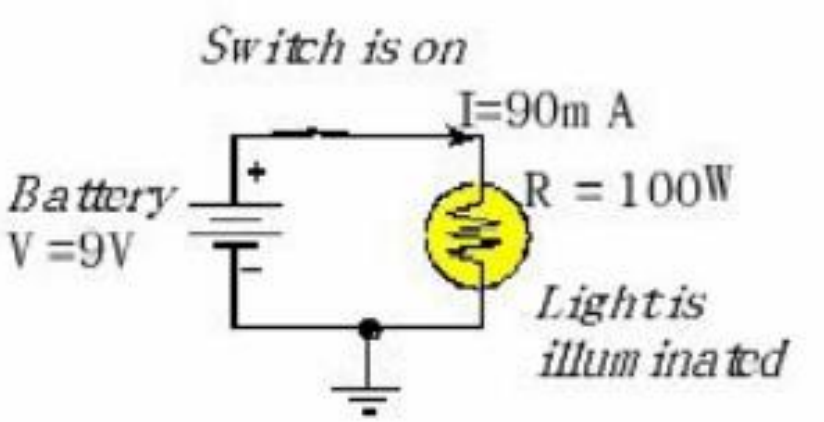
- **Ohm's Law**: The basic relation between voltage, current, and resistance for a resistor
  - Voltage = Electric Current × Resistance i.e., ***V*** = ***IR***
  - *There is 1 V across a resistor, and 2 mA is flowing. What is the resistance?*
  - *There is 5V across a 100 Ω resistor. How much current is flowing?*
  - *What happens if you place a wire directly from + terminal to the –terminal of a battery?*
- **Electric Power** (***P***): amount of energy dissipated in a resistor in unit time, measured in *watt* (*W*)
  - Power = Voltage × Current = Voltage^2 / Resistance = Current^2 × Resistance ($P = VI = V^2/R = I^2R$)
  - There is 1 V across a resistor, and 2 mA is flowing. How much power is being dissipated?
  - There is 5V across a 100 Ω resistor. How much power is being dissipated?
- **Electric Energy** (***E***):
  - Energy = Power × Time = P×t = V × I × t

Switch

BASIC ELECTRONICS REVIEW

- **Switch**: an element used to modify the behavior of the circuit
  - If the switch is pressed, its resistance is 0 (< 0.1 Ω in reality) and current can flow across the switch
  - If the switch is not pressed, its resistance is infinite (>100 MΩ in reality), and no current will flow.
  - If the switch is on, how much power is being dissipated in the bulb?
  - ***Current always flows in a loop***. e.g. when the switch is pressed, current flows out of the + side of battery, across the switch, through the light and back to the – side of the battery. When there is no loop, no current can flow.

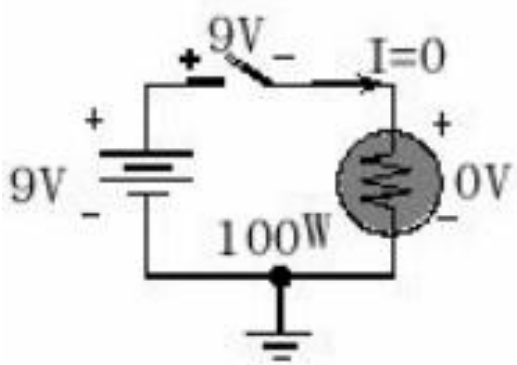


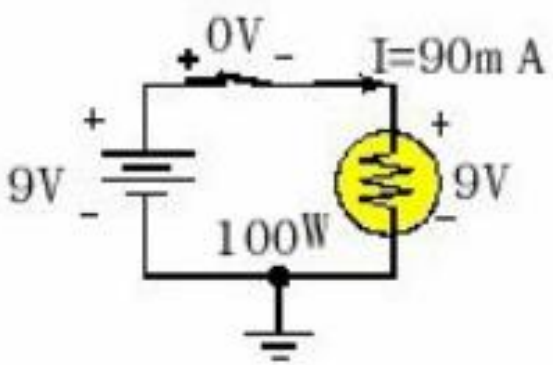


Kirchhoff's Voltage Law (KVL)

BASIC ELECTRONICS REVIEW

- There are a few basic rules that allow us to solve for voltages and currents within a circuit comprised with batteries, switches, and resistors
- **Kirchhoff's Voltage Law** (***KVL***): The sum of the voltages around the loop is zero
  - for a battery, we label the + and – sides exactly the way the battery is labeled
  - for a resistor, current enters the + label and exits the – label
  - draw arrows in the direction the currents actually flow, so the voltages will be positive
  - switch as a resistor of either 0 or infinity resistance, so it too can be labeled with a current arrow and a voltage polarity

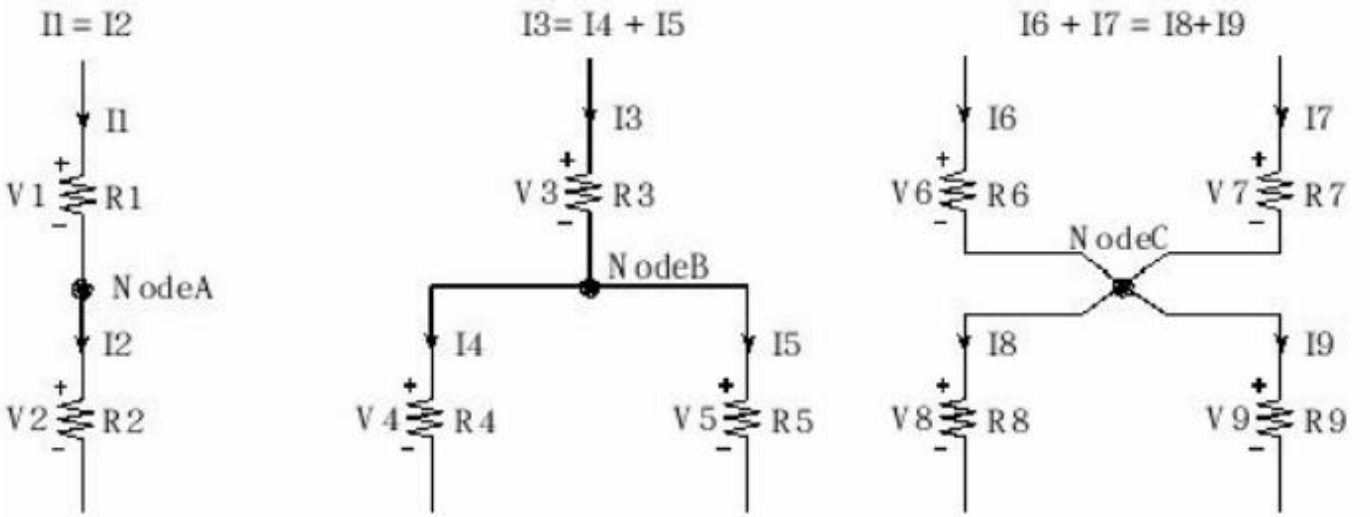




Kirchhoff's Current Law (KCL)

BASIC ELECTRONICS REVIEW

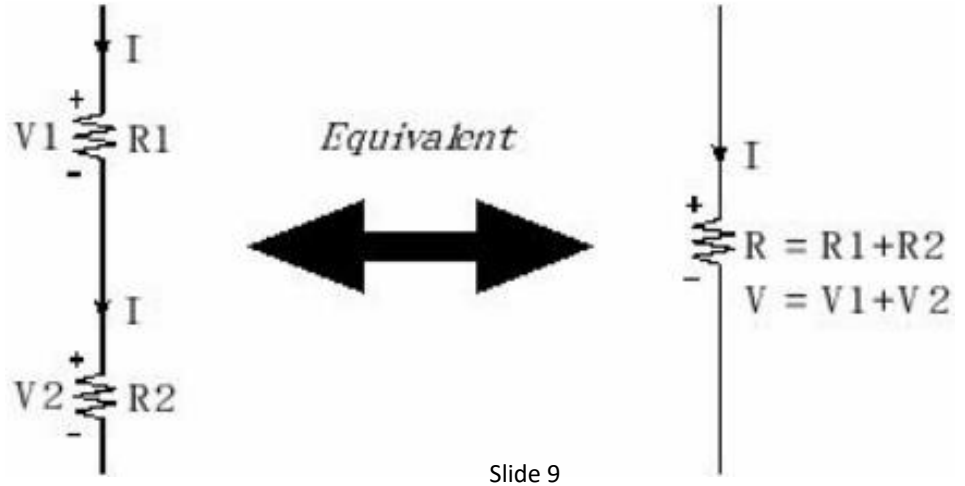
- **Kirchhoff's Current Law** (***KCL***): The sum of the currents into a node equal the sum of the currents leaving a node
  - the current arrow across a resistor goes from the + voltage to the – voltage
  - the current arrow across a battery goes from the – voltage to the + voltage



Voltage Divider Rule

BASIC ELECTRONICS REVIEW

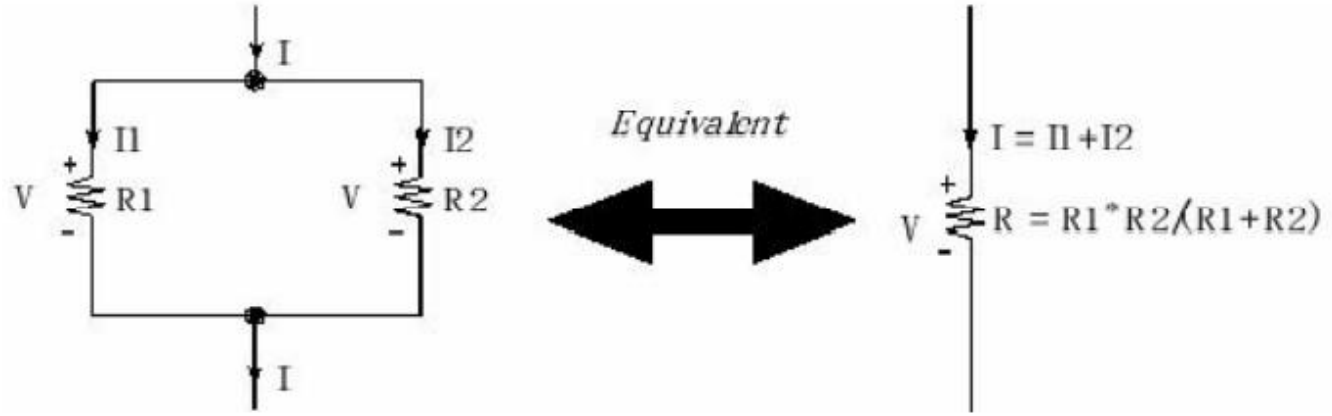
- **Series Resistance**: If resistor R1 is in series with resistor R2, this combination behaves like one resistor with a value equal to R1+R2.
  - two series resistors replaced with one resistor at R= R1+R2, the behavior will be same
  - The V equals V1+V2. By KCL, the currents through the two resistors are the same
  - These two facts can be used to derive the **voltage divider rule**
  - V2 = I×R2 = (V/R) × R2 = V × R2/(R1+R2)
  - Using Figure, assume I is 1mA, R1 is 1k Ω and R2 is 2k Ω , what is V?
  - Using Figure, assume V is 6V, R1 is 1k Ω and R2 is 2k Ω , what is V2?



Current Divider Rule

BASIC ELECTRONICS REVIEW

- **Parallel Resistance**: If resistor R1 is in parallel with resistor R2, this combination behaves like one resistor with a value equal to R = R1×R2/(R1+R2) = 1/(1/R1 + 1/R2)
  - two parallel resistors with one resistor at R= R1×R2/(R1+R2) , the behavior will be the same.
  - The I equals I1+I2. By KVL, the voltages across the two resistors are the same
  - These two facts can be used to derive the **current divider rule**
  - I1 = V/R1= (I × R)/R1 = I ×(R1 × R2/(R1+R2))/R1 = I × R2/(R1+R2)
  - Using Figure, assume I is 1mA, R1 is 2k Ω and R2 is 3k Ω , what is V?
  - Using Figure, assume V is 6V, R1 is 2k Ω and R2 is 3k Ω , what is I2?



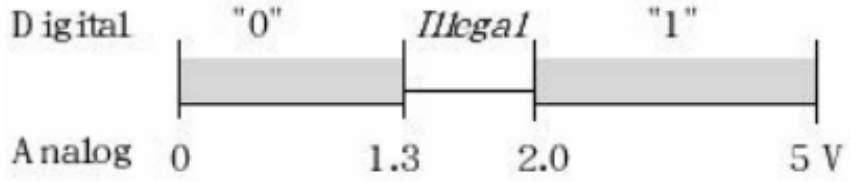
DIGITAL INFORMATION AND LOGIC

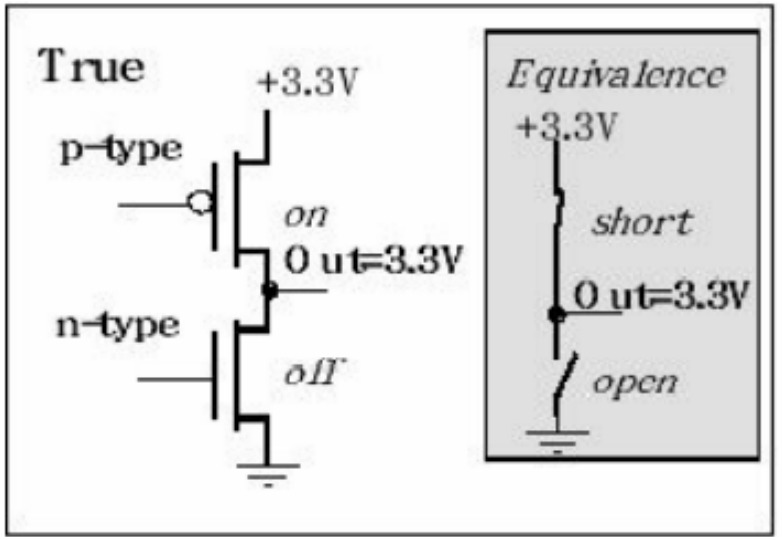
Binary Information and Digital Logic Implemented with MOS transistors, Kinds of Logic Operations,

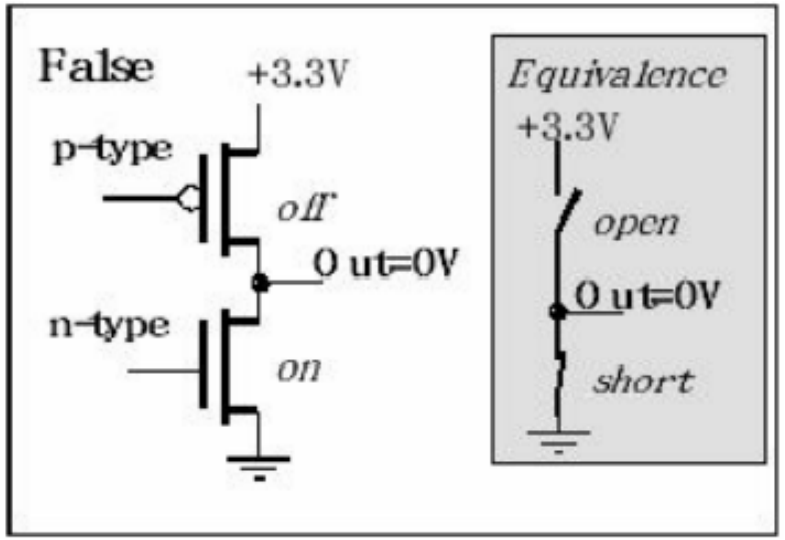
Binary Information Implemented with MOS transistors

DIGITAL INFORMATION AND LOGIC

- Information is stored on the computer in binary form
- A binary **bit** can exist in one of two possible states.
  - In positive logic, the presence of a voltage is called the '**1**', **true**, **asserted**, or **high** state
  - The absence of a voltage is called the '**0**', **false**, **not asserted**, or **low** state
- In digital circuits, each transistor is essentially on or off.
  - If the transistor is on, it is equivalent to a short circuit between its two output pins
  - Conversely, if the transistor is off, it is equivalent to an open circuit between its outputs pins
- Mapping between analog voltage and the corresponding digital meaning on the TM4C
  - What is the significance of 0.7 V separation between the two regions?



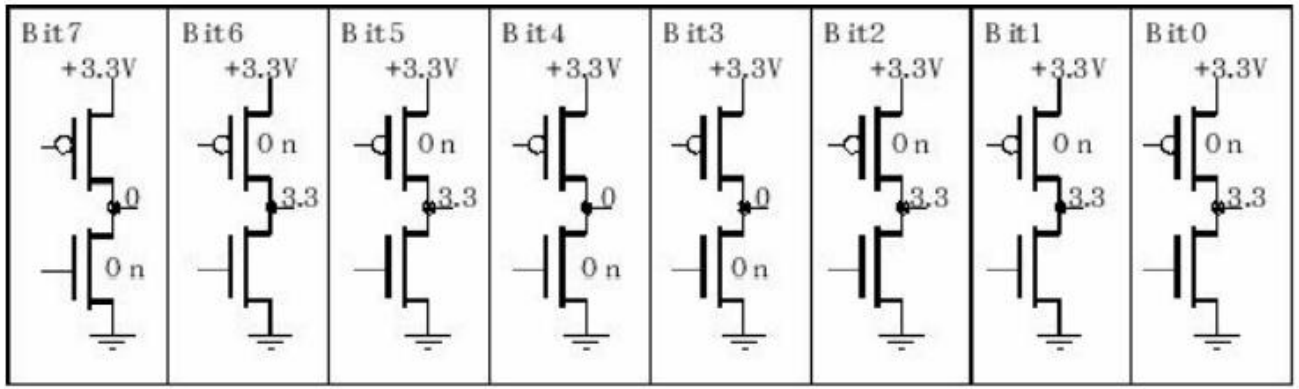




Binary Information Implemented with MOS transistors

DIGITAL INFORMATION AND LOGIC

- If the information we wish to store exists in more than two states, we use multiple bits.
  - A collection of 2 bits has 4 possible states (00, 01, 10, 11)
  - A collection of 3 bits (a **byte** or **octet**) has 8 possible states (000, 001, 010, 011, 100, 101, 110, 111)
- In general, a collection of n bits has 2^n states e.g., a byte contains eight bits, and is built by grouping eight binary bits into one object
  - Information can take many forms, e.g., numbers, logical states, text, instructions, sounds, or images
  - What bits mean depends on how the information is organized and more importantly how it is used
  - This figure shows one byte in the state representing the binary number 01100111



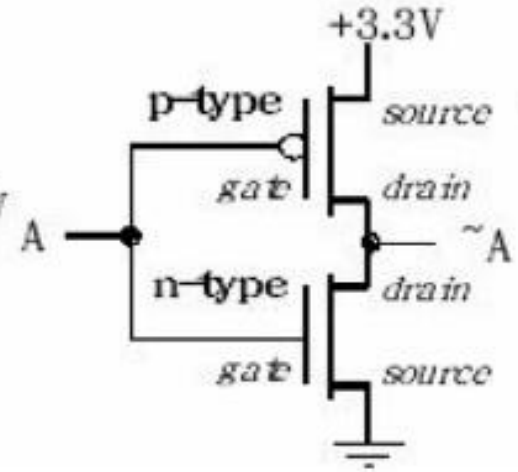
Digital Logic Implemented with MOS transistors

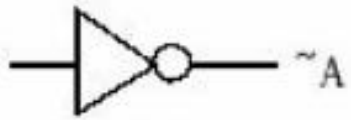
DIGITAL INFORMATION AND LOGIC

NOT GATE

- Each MOS transistors can be thought of as voltage-controlled switch
- In general, for p-type MOS transistor
  - current can flow from source to drain when active
  - the switch will be closed (transistor active) if its gate is low
- Can you guess the behavior of n-type MOS transistor?

A	p-type	n-type	~A
0 V	active	off	+3.3V
+3.3V	off	active	0V





74HC04

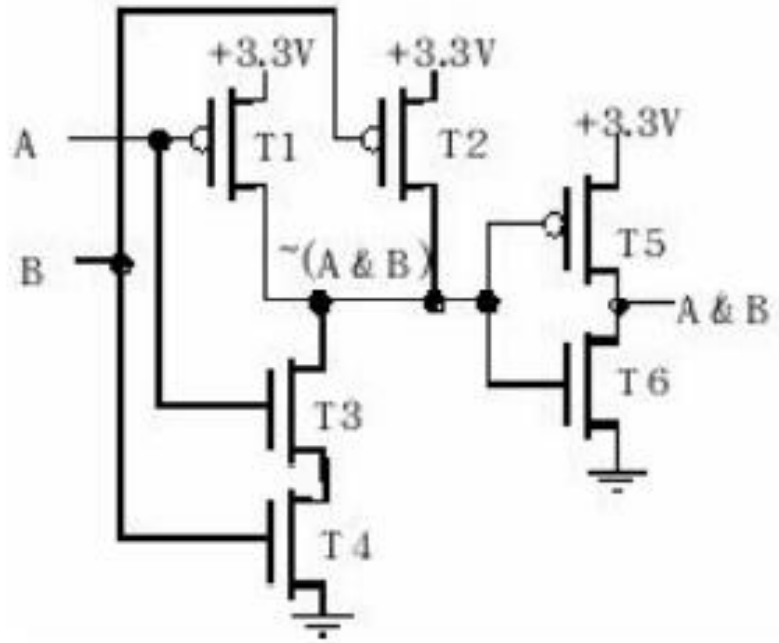
A	~A
0	1
1	0

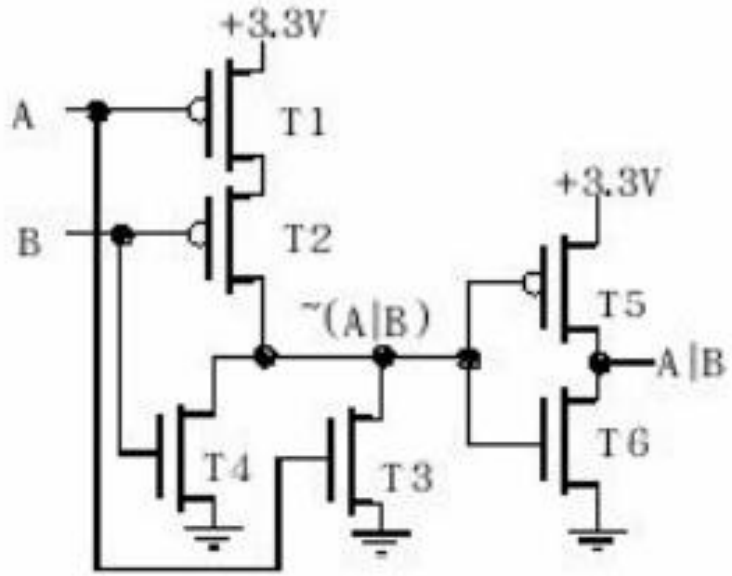
Digital Logic Implemented with MOS transistors

DIGITAL INFORMATION AND LOGIC

Multi Input Logic Gates

- The AND, OR, EOR digital logic takes two inputs and produces one output
- Can you guess which logic gates are implemented by the given circuits?





- Draw CMOS transistors implementation for NAND, NOR, XOR & XNOR gates.

Kinds of Logic Operations

DIGITAL INFORMATION AND LOGIC

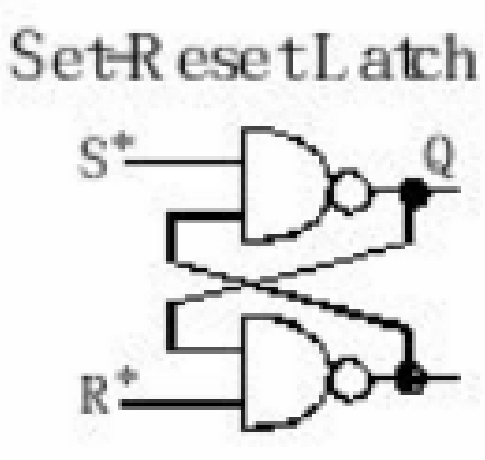
- When writing software, we will have two kinds of logic operations.
- When operating on numbers (collection of bits) we will perform logic operations bit by bit. i.e., **bitwise logic operation**. In other words, the operation is applied independently on each bit.
  - In C, the logic operator for AND is & . For example, if number A is 01100111 and number B is 11110000 then A&B = (01100111)&(11110000) = 01100000
- The other type of logic operation occurs when operating on Boolean values instead of bits. i.e., **logic operation**.
  - In C, the condition false is represented by the value 0, and true is any nonzero value.
  - In this case, if the Boolean A is 01100111 and B is 11110000 then both A and B are true. Applying Boolean AND (&&) operator, i.e., A&&B = (01100111)&&(11110000) = (1)&&(1) = 1

Digital Memory Storage Units

DIGITAL INFORMATION AND LOGIC

SET RESET LATCH

- S* and R* are the set and reset inputs respectively and Q is the output.
  - If S*=0 and R*=1, then the Q will be 1.
  - Conversely, if S*=1 and R*=0, then the Q will be 0.
- How SR-Latch works?
  - Normally, we leave both the S* and R* inputs high.
  - We make the signal S* go low, then back high to set the latch, making Q=1.
  - Conversely, we make the signal R* go low, then back high to reset the latch, making Q=0.
  - If both S* and R* are 1, the value on Q will be remembered or stored.
- This latch enters an unpredictable mode when S* and R* are simultaneously low.

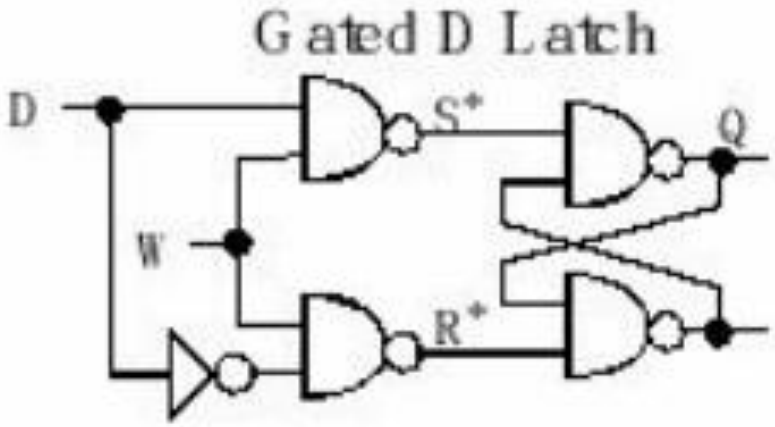


Digital Memory Storage Units

DIGITAL INFORMATION AND LOGIC

GATED D LATCH

- It take a data input, D, and a control signal, W, and produce the S* and R* commands for the set-reset latch
  - if W=0, then the latch remember the value on Q that was previously written
  - if W=1, then the data input, D is stored into the latch
    - if D=1 and W=1, then S*=0 and R*=1, making Q=1
    - if D=0 and W=1, then S*=1 and R*=0, making Q=0
- How Gated D-Latch works?
  - put the data on the D input, next make W go high, and then make W go low
  - This causes the data value to be stored at Q
  - After W goes low, the data does not need to exist at the D input anymore
  - If the D input changes while W is high, then the Q output will change correspondingly
  - However, the last value on the D input is remembered or latched when the W falls

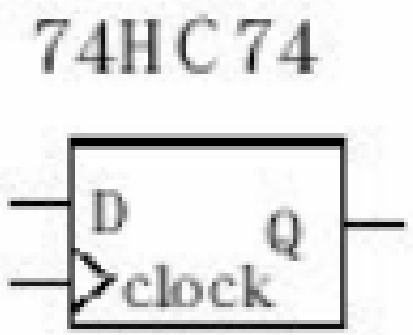


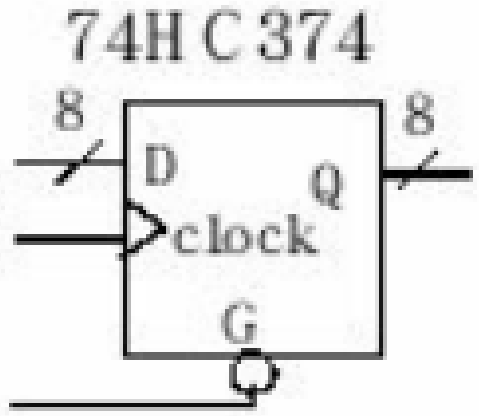
Digital Memory Storage Units

DIGITAL INFORMATION AND LOGIC

D Flip-Flop

- D flip-flops are the basic building block of RAM and registers on the computer
- The D flip-flops are edge-triggered, meaning that changes in the output occur at the rising edge of the input clock
- How D Flip-Flop works?
  - First place the digital value on the D input, give a rising edge to the clock input
  - After the clock rising edge, the value is available at the Q output, and the D input is free to change
- The 74HC374 is an 8-bit D flip-flop, such that all 8 bits are stored on the rising edge of a single clock.



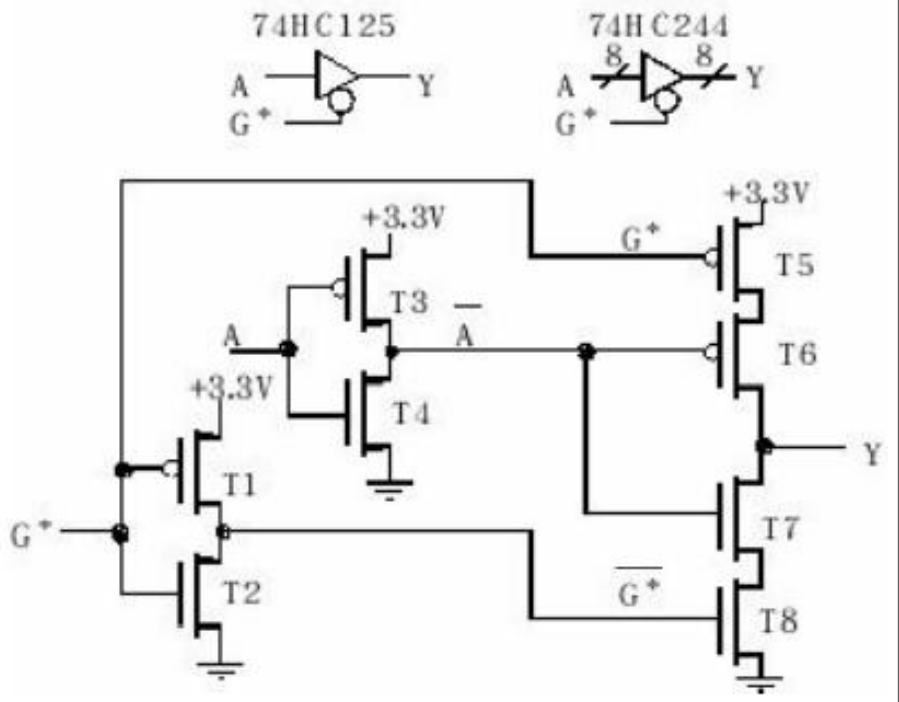


Digital Memory Storage Units

DIGITAL INFORMATION AND LOGIC

Tri-State Driver

- Essential component of computers, the tristate driver can be used dynamically control signals within the computer.
- It is called **tristate** because there are three possible outputs: **high**, **low**, and **HiZ** or high impedance (floating state)
- How tristate driver works?
  - To activate the driver, make its gate (G*) low.
  - When the driver is active, its output (Y) equals its input (A).
  - To deactivate the driver, make its G* high.
  - When the driver is not active, its output Y floats independent of A
- The HiZ output means the output is neither driven high nor low

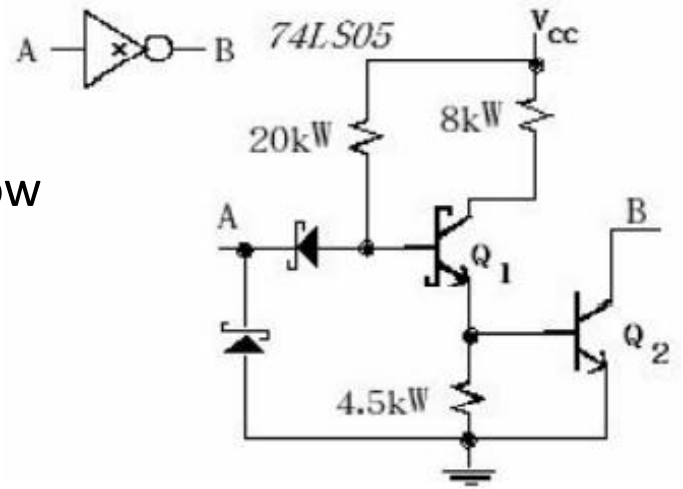


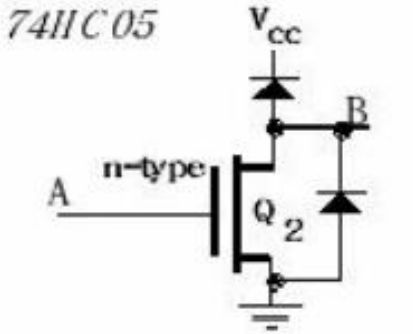
Digital Memory Storage Units

DIGITAL INFORMATION AND LOGIC

Open Collector NOT Gate

- The output of an open collector gate, drawn with the '×', has two states low (0V) and HiZ (floating)
- How open collector gate works?
  - If A is high (+3.3V), the transistor is active, and the output is low (0V)
  - If A is low (0V), the transistor is off, and the output is neither high nor low
- In general, we can use an open collector NOT gate to switch current on and off to a device, such as a relay, a light emitting diode (LED), a solenoid, or a small motor.
- The 74HC05, the 74LS05, the 7405, and the 7406 are all open collector NOT gates.
- 74HC04 is high-speed CMOS and can only sink up to 4 mA when its output is low.
- Since the 7405 and 7406 are transistor-transistor-logic (TTL) they can sink more current.
- In particular, the 7405 has a maximum output low current (IOL) of 16 mA, whereas the 7406 has a maximum IOL of 40 mA.





A	Q_2	B
Low	off	HiZ
High	active	Low

DIGITAL REPRESENTATION OF NUMBERS

Number Systems and Conversions, Signed Numbers, IEC Prefixes

Number Systems

DIGITAL REPRESENTATION OF NUMBERS

- To solve problems using a computer we need to understand numbers
- In general, each digit in a **number system** has a place, equal to power of **Base** N, and the **value** selected from the set starting 1 to N-1 i.e., {0, 1, 2, 3, …, N-1}
  - **Binary Number System**: Base N = **2**, Set of values = {**0, 1**}
  - **Decimal Number System**: Base N = **10**, Set of values = {**0, 1, 2, 3, 4, 5, 6, 7, 8, 9**}
  - **Hexadecimal Number System**: Base N = **16**, Set of values = {**0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**}
- For example, a 4-digit decimal number $1984 = 1\times10^3 + 9\times10^2 + 8\times10^1 + 4\times10^0$
- For decimal number, $273.15 = 2\times10^2 + 7\times10^1 + 3\times10^0 + 1\times10^{-1} + 5\times10^{-2}$
- What is the numerical value of the 8-bit binary number 11111111_2?
- What is the numerical value of the 16-bit hexadecimal number 0x12AD?

Number Systems

DIGITAL REPRESENTATION OF NUMBERS

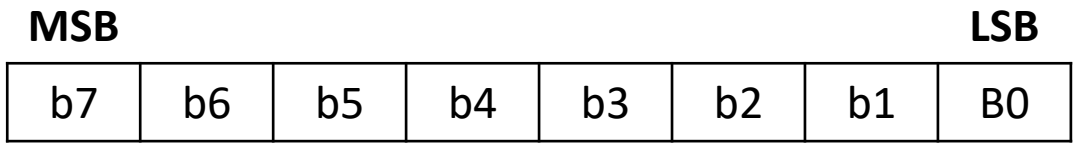
- Binary is a natural language of computers but difficult for humans
  - A 4-bit binary number is called a **Nibble**, e.g. 0000, 1001, 1111 etc.
  - An 8-bit binary number is called a **Byte**, e.g. 10011001, 11110000, etc.
  - A 16-bit binary number is called a **Halfword**, e.g. 1010110010011001, 1111000011110000, etc.
- Hexadecimal (often abbreviated as "hex") representation is a convenient mechanism for us humans to define binary information, because it is extremely simple for humans to convert back and forth between binary and hexadecimal.
- A **nibble** is defined as 4 binary bits, or one hexadecimal digit. Each value of the 4-bit nibble is mapped into a unique hex digit
  - Convert the binary number 110110101011_2 to hexadecimal.
  - Convert the hex number 0xBEEF to binary.
  - How many binary bits does it take to represent 0x12345?

Hex Digit	Decimal Value	Binary Value
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A or a	10	1010
B or b	11	1011
C or c	12	1100
D or d	13	1101
E or e	14	1110
F or f	15	1111

Signed Numbers

DIGITAL REPRESENTATION OF NUMBERS

- A **byte** contains 8 bits, where each bit b7,...,b0 is binary and has the value 1 or 0.
- If a byte is used to represent an unsigned number, then the value of the number is
  - $N = 128 \times b7 + 64 \times b6 + 32 \times b5 + 16 \times b4 + 8 \times b3 + 4 \times b2 + 2 \times b1 + b0$
  - Notice that the significance of bit n is 2^n
- Therefor, we specify b7 as the **most significant bit** or MSB, and b0 as the **least significant bit** or LSB.
- In C, the specifier *char* means 8 bits or 1 byte. In C99, the specifiers *uint8_t* and *int8_t* mean 8 bits or 1 byte.
- In this class, we will use **char** to store ASCII characters and we will use **uint8_t** and **int8_t** to store 8-bit numbers.



Signed Numbers

DIGITAL REPRESENTATION OF NUMBERS

- One of the first schemes to represent signed numbers was called **one's complement**.
- It was called one's complement because to negate a number, we complement (logical not) each bit.
- For example, if 25 equals 00011001_2 in binary, then –25 is 11100110_2 .
- An 8-bit one's complement number can vary from -127 to +127.
- The most significant bit is a sign bit, which is 1 if and only if the number is negative.
- The difficulty with this format is that there are two zeros +0 is 00000000_2 , and –0 is 11111111_2
- What is the one's complement representation of -65?

Signed Numbers

DIGITAL REPRESENTATION OF NUMBERS

- The **two's complement** number system is the most common approach used to define signed numbers.
- It is called two's complement because to negate a number, we complement each bit (like one's complement), and then add 1.
- For example, if 25 equals 00011001_2 in binary, then -25 is 11100111_2 .
- If a byte is used to represent a signed two's complement number, then the value of the number is $N = -128 \times b7 + 64 \times b6 + 32 \times b5 + 16 \times b4 + 8 \times b3 + 4 \times b2 + 2 \times b1 + b0$
- There are 256 different signed 8-bit numbers. The smallest signed 8-bit number is -128 and the largest is 127
- One usually means two's complement when one refers to signed integers
- Convert the signed binary number 11101010_2 to signed decimal.
- Convert -73 decimal number to 8-bit signed binary number.

IEC Prefixes

DIGITAL REPRESENTATION OF NUMBERS

- Use of a binary prefixes, defined by IEC, are appropriate for representing large unit of information or computer storage
- The correct terminology is to use the SI-decimal abbreviations to represent powers of 10, and the IEC-binary abbreviations to represent powers of 2
- The term kibibyte is a contraction of kilo binary byte and is a unit of information abbreviated KiB
- 1 KiB = 2^{10} bytes = 1024 bytes ≠ 1000 bytes

Value	SI Decimal	SI Decimal		Value	IEC Binary	IEC Binary
1000^1	k	kilo-		1024^1	Ki	kibi-
1000^2	M	mega-		1024^2	Mi	mebi-
1000^3	G	giga-		1024^3	Gi	gibi-
1000^4	T	tera-		1024^4	Ti	tebi-
1000^5	P	peta-		1024^5	Pi	pebi-
1000^6	E	exa-		1024^6	Ei	exbi-
1000^7	Z	zetta-		1024^7	Zi	zebi-
1000^8	Y	yotta-		1024^8	Yi	yobi-

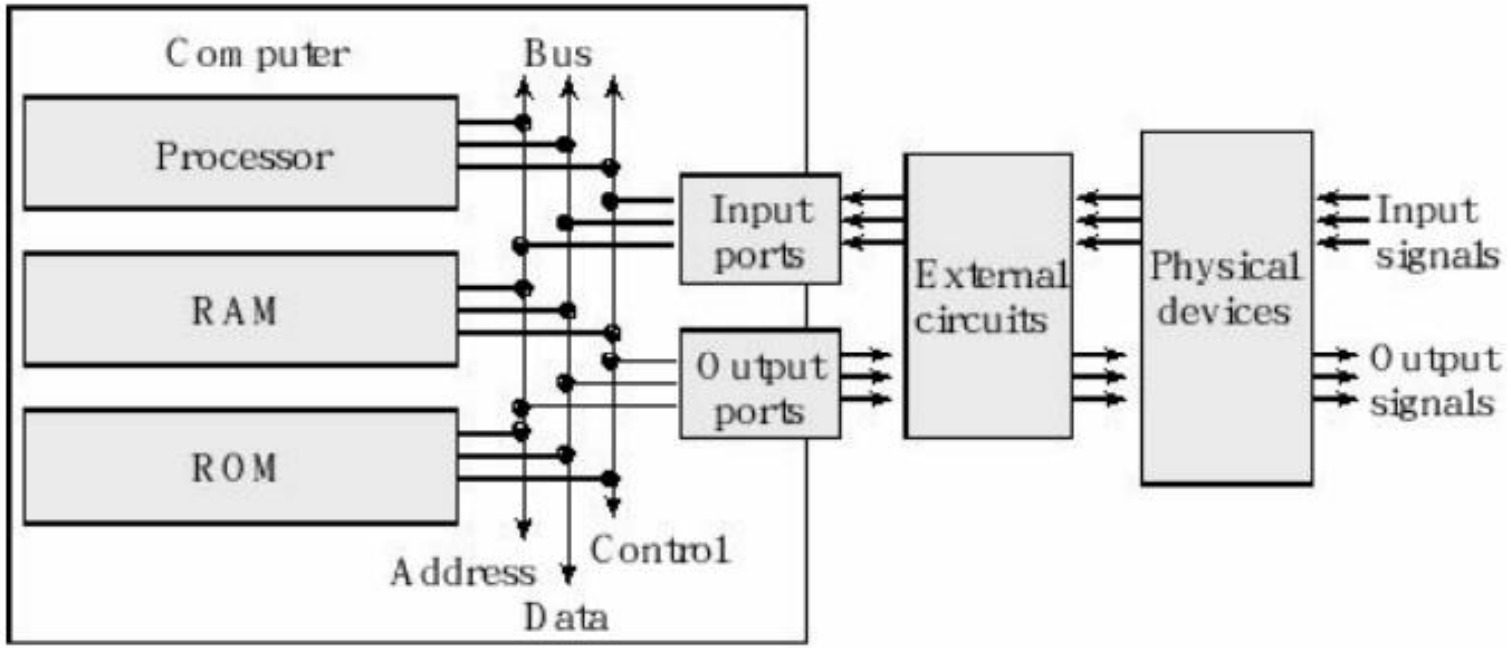
COMPUTER ARCHITECTURE

Elements of a Computer, Processor, RAM, ROM, I/O Ports, Bus, Read/Write Cycle, DMA

Elements of a Computer

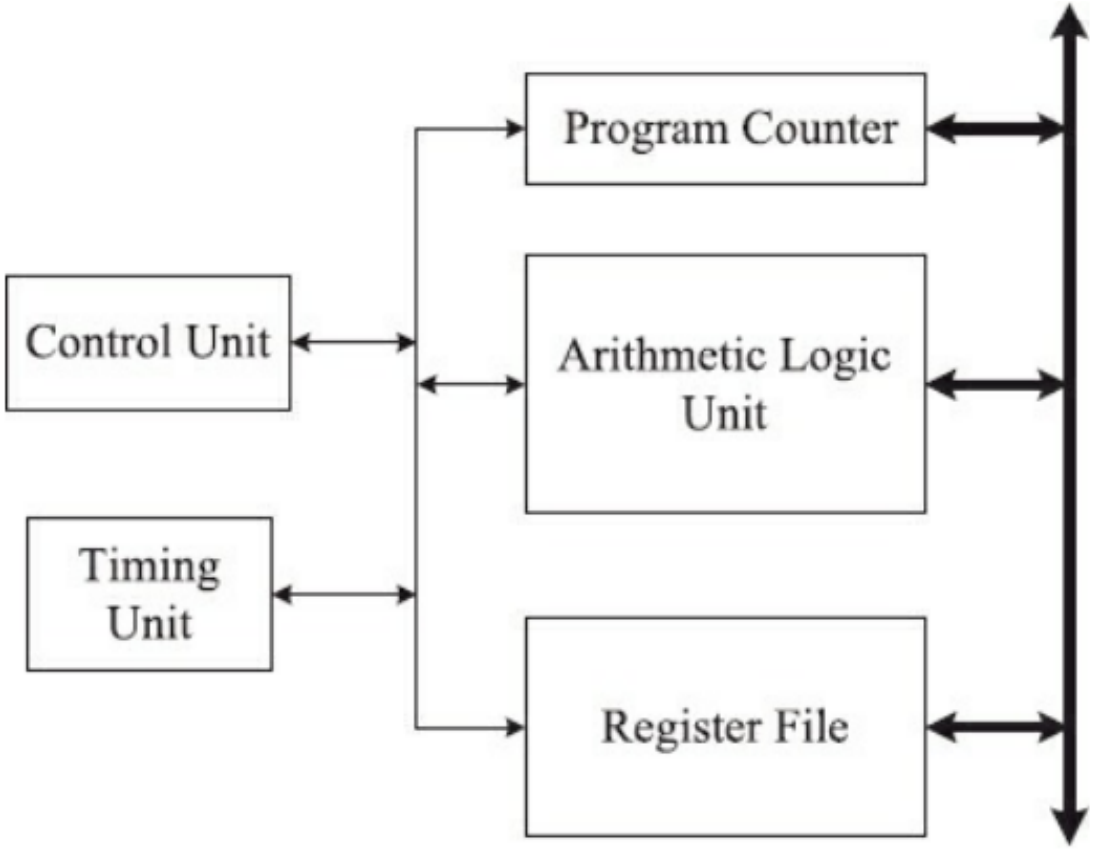
COMPUTER ARCHITECTURE

- A computer combines a **processor**, random access memory (**RAM**), read only memory (**ROM**), and input/output (**I/O**) ports, all connected through **bus**.
- ***von Neumann Architecture*** of a computer



Processor

COMPUTER ARCHITECTURE



- A processor, also known as a central processing unit, has a capability of executing instructions at an extremely high speed.
- Following are the five basic components of a processor:
  - **Program counter**: carries the address of next (opcode) instruction to be performed by the ALU
  - **Arithmetic logic unit (ALU)**: perform arithmetic and logical operations (e.g. add, subtract, AND, etc.)
  - **Register file**: to temporarily store the operands of operation to be performed by the ALU
  - **Control unit**: decodes opcode and generates binary signals to control execution of an operation
  - **Timing Unit**: clock to synchronize processor tasks according to a timing sequence
  - **Bus**: data transfer among processor register and ALU

Processor

COMPUTER ARCHITECTURE

- The job of a microprocessor is to execute a user program.
- The executable program is stored in memory initially and is executed instruction by instruction.
- During program execution, the microprocessor performs different operations including
  - fetching instructions and data from memory,
  - communicating with input/output ports, etc.
- The instructions from memory and data from memory or I/O ports is stored temporarily inside the microprocessor using processor registers.
- Now a days, processors can execute billions of such instructions in one second

Processor

COMPUTER ARCHITECTURE

Steps to execute the R = X + Y addition instruction

Step 1: The microprocessor fetches the instruction by applying PC contents on the address bus, a read signal on the control bus

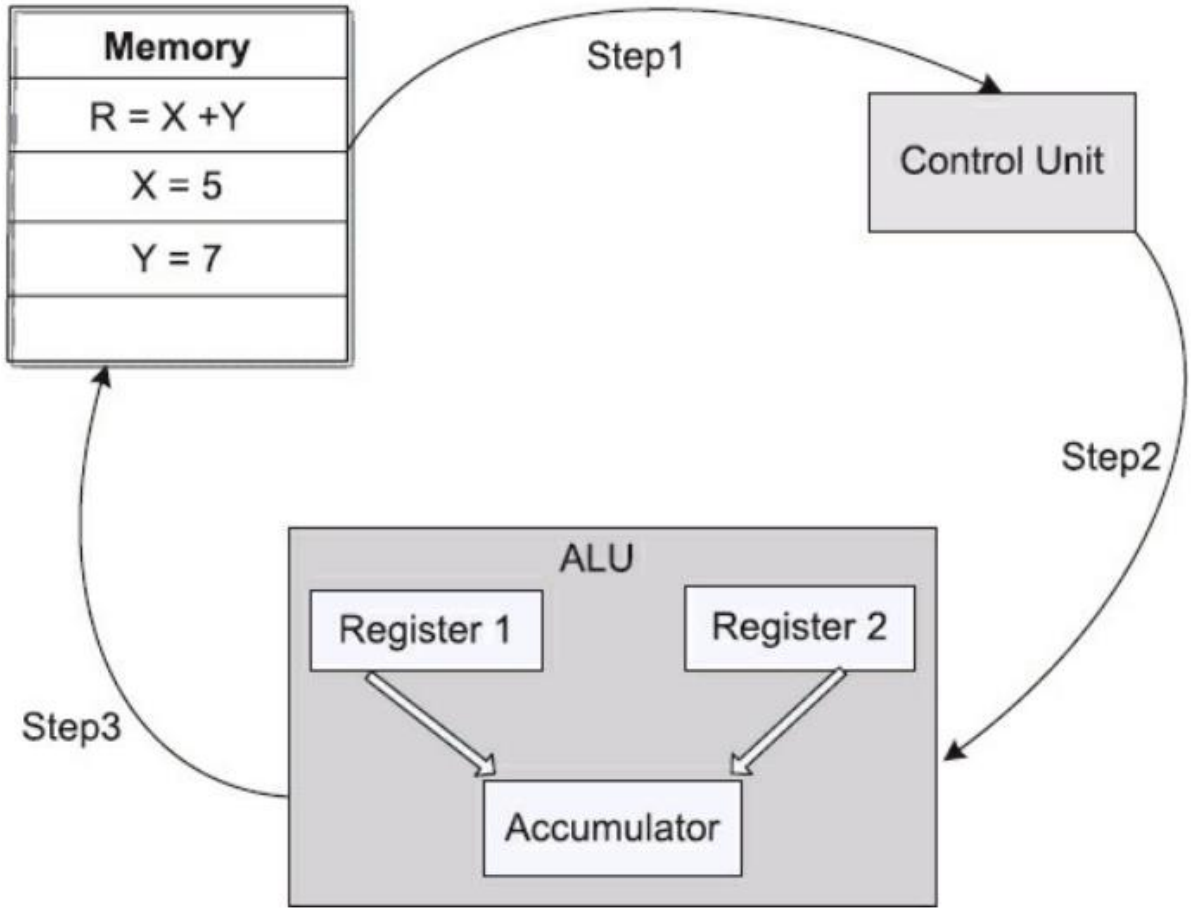
Step 2: The control unit decodes its opcode

Step 3: Processor executes R = X + Y by

- fetching the current value of X from the memory
- fetching the current value of Y from the memory
- instructing the ALU to add these two numbers
- writing the sum back to the memory address of R

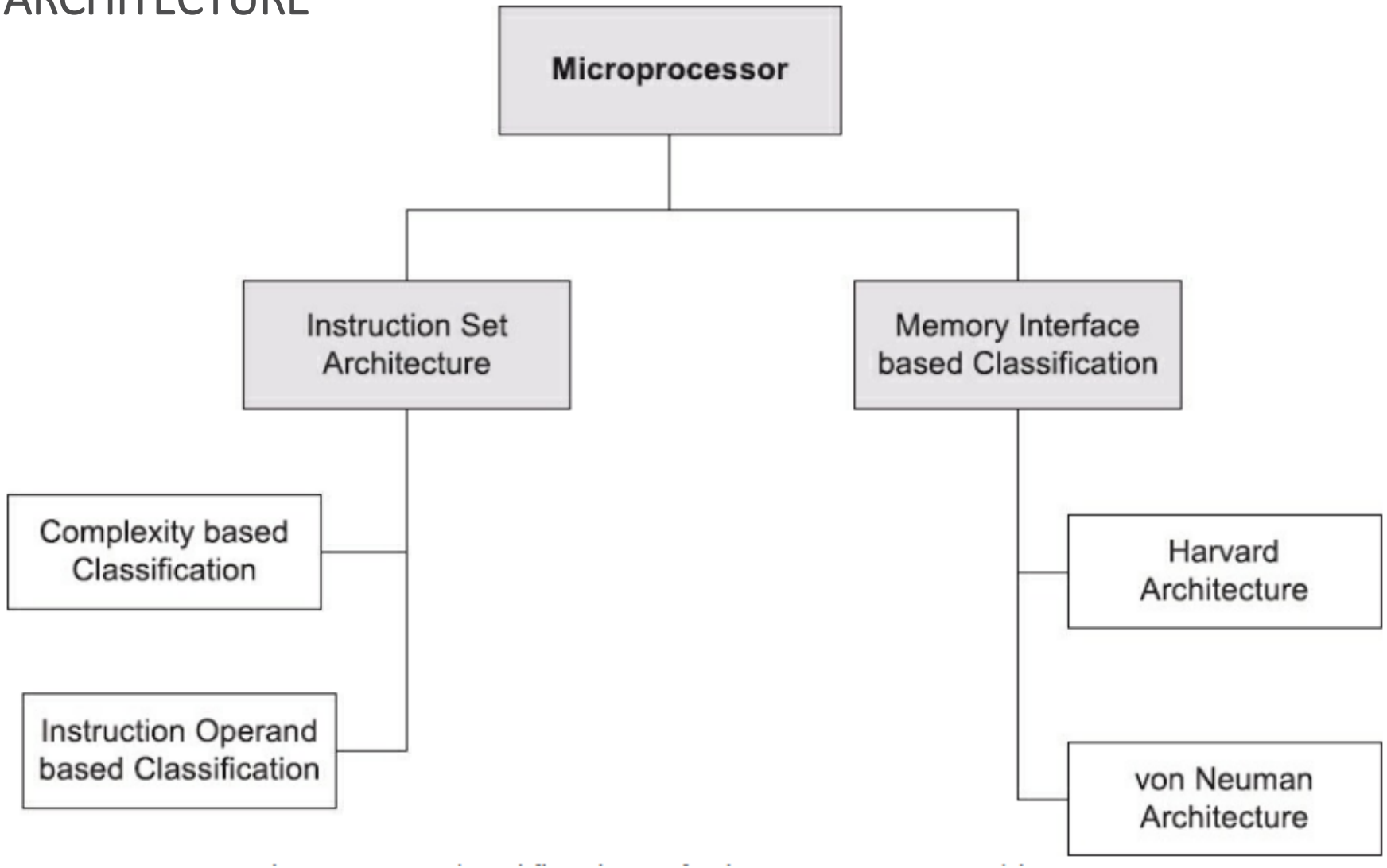
R = X + Y instruction will consist of 3 fields:

1) operational code representing addition
2) memory address of variable X
3) memory address of variable Y



Processor Architecture Classification

COMPUTER ARCHITECTURE



Instruction Set Architecture (ISA)

There are two important aspects that can be used to define ISA classification.

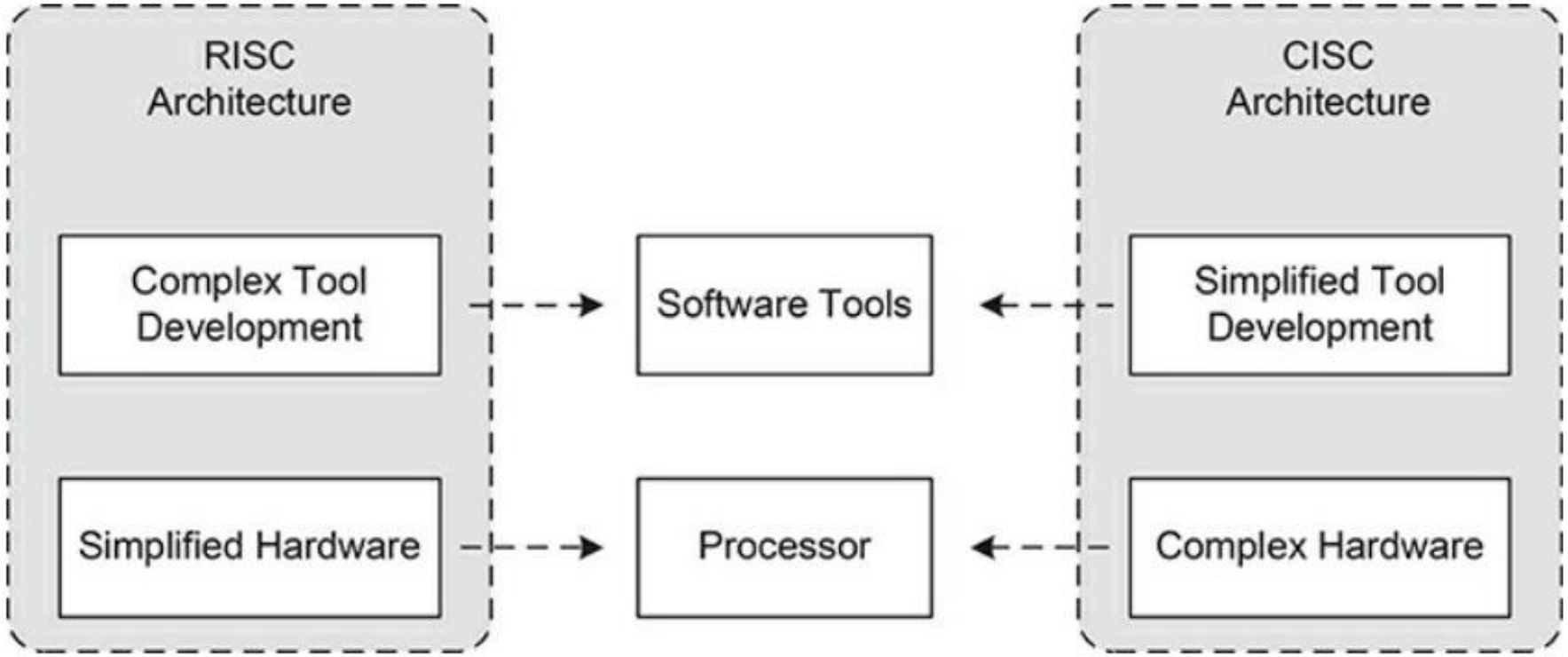
1. based on the complexity of instructions
2. based on the instruction operands

Processor

COMPUTER ARCHITECTURE

Instruction Set Architecture (ISA): based on the complexity of instructions

Microprocessor can be categorized into complex instruction set computer (CISC) and reduced instruction set computer (RISC).



Processor

COMPUTER ARCHITECTURE

Instruction Set Architecture (ISA): based on the complexity of instructions

Complex instruction set computers (**CISC**)

- Suited for computers where the processors are much faster than available memories
- A single complex instruction can perform many operations
- High code density due to small number of complex instructions
- The complexity is embedded in the processor hardware, making the compilation tools design simpler
- CISC architecture examples are Intel (x86) and Freescale 9S12

Reduced Instruction Set Computers (**RISC**)

- suited for those scenarios where the processor speeds match that of memories
- Each instruction can perform one simplified operation at a time
- Low code density due to relatively larger number instructions required for a task
- Hardware architecture simplicity in RISC is complemented by the complexity in the generation of assembly code by the tools (compiler) or by the programmer.
- Examples of RISC are MIPS, ARM, SPARC, and PowerPC

Processor

COMPUTER ARCHITECTURE

Instruction Set Architecture (ISA): based on the instruction operands

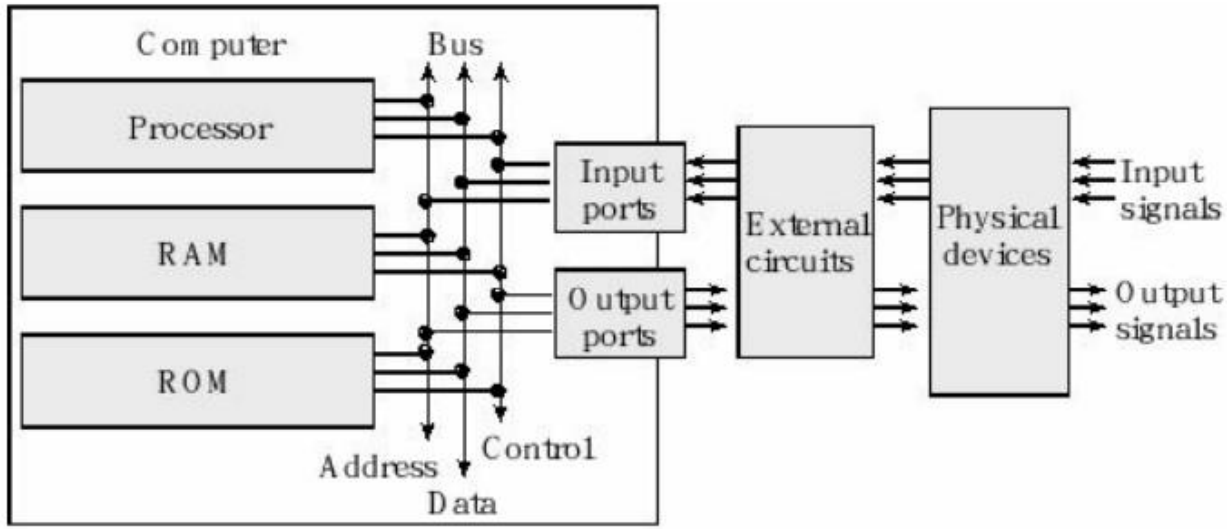
- **Memory-memory**: more than one operand of most instructions can be specified in memory, e.g., VAX and PDP series
- **Register-memory**: one operand is specified in memory, while the other operand is in CPU register, e.g., intel's x86 and Motorola 68k.
  - Due to fewer memory accesses its instructions execute faster, compared to memory-memory based ISA but it requires more of number of instructions to complete the same task.
- **Register-register**: This ISA classification is also called load-store architecture, e.g., ARM, MIPS
  - Direct access to the memory is not allowed to most of the instructions in this ISA
  - load and store instructions, are responsible for any data movement between registers and memory
  - All instructions get their operands from and store their results to registers
  - The execution of most of the instructions is very fast, in many cases single clock cycle
  - requires the greatest number of instructions to complete a given task
  - Saves time by avoiding extra memory load/store operations for immediate results of any operation

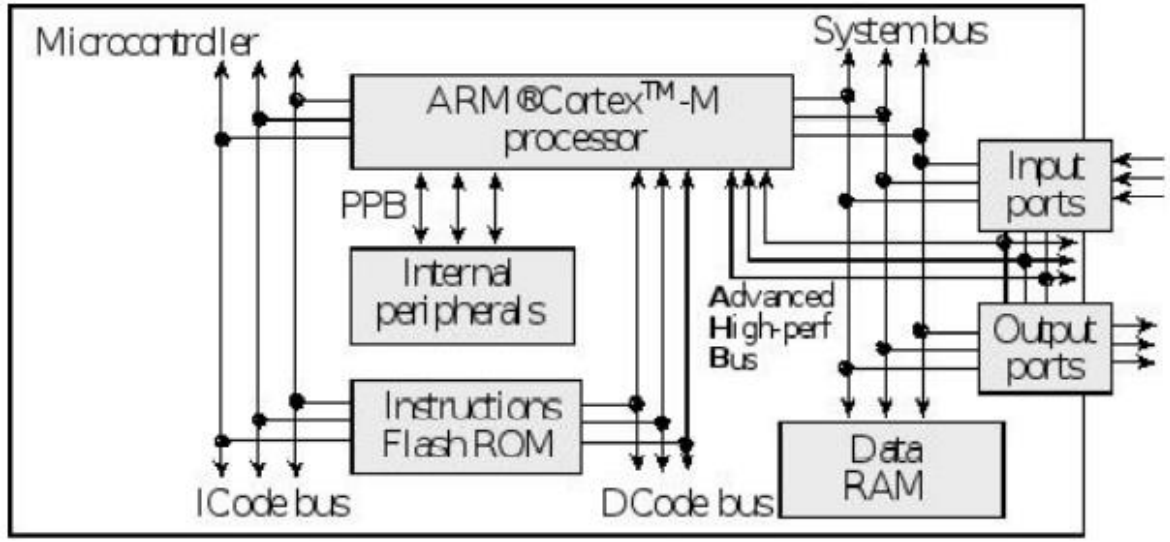
Processor

COMPUTER ARCHITECTURE

Memory Interface-Based Architecture Classification

- **von Neumann architecture**: uses a common bus for both data as well as code memory
- **Harvard architecture**: utilizes separate buses for accessing code and data memories





- What advantage von Neumann has over Harvard architecture?
- What advantage Harvard has over von Neumann architecture?

Random Access Memory

COMPUTER ARCHITECTURE

- The computer can store information in **RAM** by writing to it, or it can retrieve previously stored data by reading from it.
- Most RAMs are volatile; meaning if power is interrupted and restored the information in the RAM is lost.
- RAM is not limited by the number of read and write cycles and is more suitable for storing data that is updated frequently.(Scratch pad memory)
- The read and write operations of RAMs are faster than those of ROMs
- For byte addressable memories, with 16/32-bit data bus interface, the memory accesses can be either aligned or unaligned

Random Access Memory

COMPUTER ARCHITECTURE

- There are different types of RAMs: SRAM, DRAM, Synchronous DRAM, double data rate SDRAM, and Rambus DRAM.
- Static RAM (**SRAM**)
  - employs a flip-flop (consisted of 6 MOSFETs) for storing a bit in a memory cell
  - 4 transistors are used to create two cross-coupled inverters that store the binary information, and the other two are used to read and write the bit.
  - It doesn't require a refreshing circuitry
  - SRAM is normally used for building cache memories
- Dynamic RAM or **DRAM** are smaller in size
  - Each memory cell can store one bit of information, is made up of two transistors (as switches) and a capacitor to hold charge
  - As the capacitor charge leaks, the voltage representing '1' needs to be refreshed. This refresh operation is performed several times in one second and results in reducing the memory operating speed
  - DRAM is mostly employed for RAM chips due to low cost

Aligned and Unaligned Memory Accesses

COMPUTER ARCHITECTURE

A 32-bit processor will most probably have a 32-bit data bus to access the memory. For such a scenario, it is possible to perform memory read/write operations of byte (8-bit), halfword (16-bit), and word (32-bit) size.

- For a **byte** addressable memory, each memory access of size byte is inherently aligned.
- If a memory access, of size **halfword**, is performed from an even address, then this memory access is an aligned access. On the other hand, if an odd address is used, then the resulting memory access is an unaligned access.
- Similarly, for a 32-bit word memory access, **word** aligned means the data is stored on a word boundary, i.e., the memory address accessed is divisible by 4. A word memory access from an address not divisible by 4 is termed unaligned word access.

Aligned and Unaligned Memory Accesses

COMPUTER ARCHITECTURE

Memory address	Alignment (8 bit)	Alignment (16 bit)	Alignment (32 bit)
0x0000_0000	Aligned	Aligned	Aligned
0x0000_0001	Aligned	Non Aligned	Non Aligned
0x0000_0002	Aligned	Aligned	Non Aligned
0x0000_0003	Aligned	Non Aligned	Non Aligned
0x0000_0004	Aligned	Aligned	Aligned
0x0000_0005	Aligned	Non Aligned	Non Aligned
0x0000_0006	Aligned	Aligned	Non Aligned
0x0000_0007	Aligned	Non Aligned	Non Aligned
0x0000_0008	Aligned	Aligned	Aligned

Read Only Memory

COMPUTER ARCHITECTURE

- The memory that allows the processor to only read its contents is read only memory (ROM).
- This type of memory is non-volatile, i.e., it can store information permanently even when no power is applied.
- Instructions or a program code are the information that is normally stored in a ROM.
- For example, the boot sequence with which an operating system is loaded is a piece of code, which does not vary over time and is stored in a ROM.
- This type of memory is low power, low cost, and small size
- Based on how a ROM is programmed, ROMs can be of different types as explained next

Read Only Memory

COMPUTER ARCHITECTURE

1. Programmable ROMs (PROMs)
   - can be programmed only once.
   - any change to the contents will require the replacement of the chip.
2. Erasable Programmable ROMs (EPROMs)
   - can be reused by erasing and reprogramming with the help of ultraviolet (UV) light.
   - need to remove it with the help of dedicated equipment and reinstall it whenever we need to change its current contents.
   - Another drawback of EPROM is that the whole chip needs to be completely erased
3. Electrically Erasable PROMs (EEPROMs)
   - contents can be erased and written to by applying electric signals to the storage cells while chip is installed in the circuit
   - Slow speed devices due to single byte read and write operations. However, modern EEPROMs are capable of multi-byte-based page operations.

Read Only Memory

COMPUTER ARCHITECTURE

4. Flash memory
   - Type of EEPROM that allows read and write operations to be carried out in large multi-byte blocks
   - It allows erasing large block sizes, which provides these memories a significant speed advantage compared to the EEPROM when dealing with large amounts of data.
   - Its cost is also quite low compared to byte-programmable EEPROM.
   - One of the limitations of flash memories as well as EEPROMs is their limited number of read and write cycles.

Input/Output Ports

COMPUTER ARCHITECTURE

- An **input port** is hardware on the microcontroller that allows information about the external world to be entered into the computer.
- The microcontroller also has hardware called an **output port** to send information out to the external world.
- In general, we can classify I/O interfaces into four categories
  - **Parallel** - binary data are available simultaneously on a group of lines
  - **Serial** - binary data are available one bit at a time on a single line
  - **Analog** - data are encoded as an electrical voltage, current, or power
  - **Time** - data are encoded as a period, frequency, pulse width, or phase shift

Input/Output Ports

COMPUTER ARCHITECTURE

- An **interface** is defined as the collection of the I/O ports, external electronics, physical devices, and software, which combine to allow the computer to communicate with the external world
- An example of an input interface is a switch, where the operator toggles the switch, and the software can recognize the switch position.
- An example of an output interface is a light-emitting diode (LED), where the software can turn the light on and off, and the operator can see whether the light is shining.
- List three input interfaces available on a personal computer
- List three output interfaces available on a personal computer

Input/Output Ports

COMPUTER ARCHITECTURE

- All the I/O ports in Cortex-M4 microcontroller are memory mapped I/Os
- In a system with **memory mapped I/O**, the I/O ports are connected to the processor in a manner like memory.
- I/O ports are assigned addresses, and the software accesses I/O using reads and writes to the specific I/O addresses.
- The software inputs from an input port using the same instructions as it would if it were reading from memory.
- Similarly, the software outputs from an output port using the same instructions as it would if it were writing to memory.

Bus

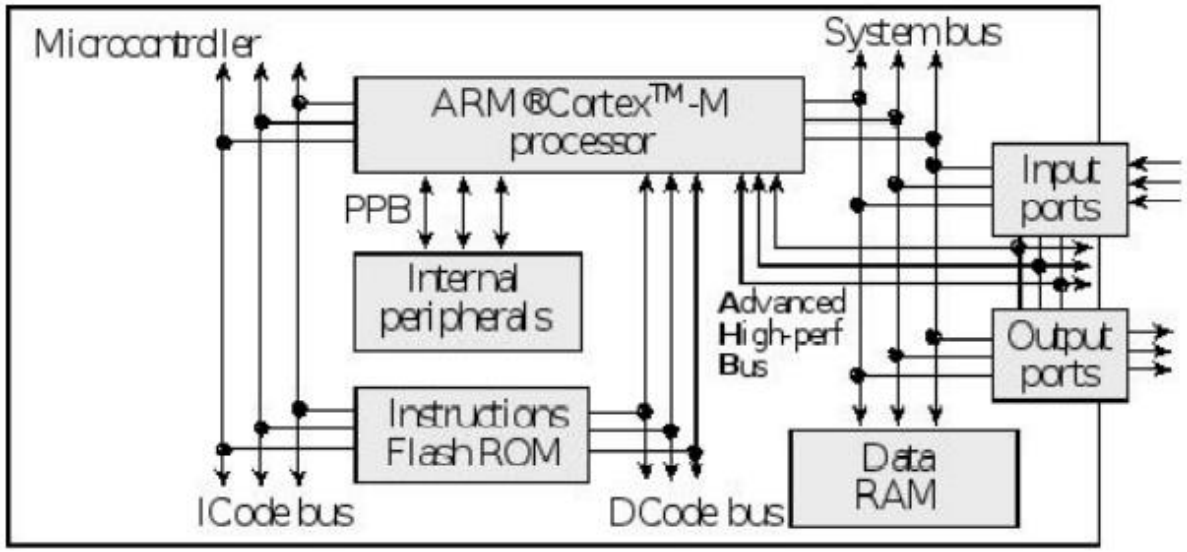
COMPUTER ARCHITECTURE

- A **bus** is defined as a collection of signals, which are grouped for a common purpose.
- The bus has three types of signals:
  - **Address signals**: specifies which module is being accessed
  - **Data signals**: contains the information being transferred
  - **Control signals**: specify the direction of transfer, the size of the data, and timing information
- Together, the bus directs the data transfer between the various modules in the computer
- Based on bus configurations, computers can be divided into two broad categories:
  - **von Neumann Architecture:** Common bus configuration
  - **Harvard Architecture:** Separate buses for accessing instructions and data
- ARM Cortex-M processor is an example of **Harvard Architecture**

Bus

COMPUTER ARCHITECTURE

- There are five buses on ARM Cortex-M processor (***Harvard Architecture***)
  - **ICode bus:** Fetch opcodes from ROM
  - **DCode bus:** Read constant data from ROM
  - **System bus:** Read/write data from RAM or I/O, fetch opcode from RAM
  - **Private peripheral bus:** Read/write data from internal peripherals like the NVIC
  - **Advanced High-performance bus**: Read/write data from high-speed I/O and parallel ports
- The multiple-bus architecture allows simultaneous bus activity, greatly improving performance over single-bus architectures.



Read/Write Cycles

COMPUTER ARCHITECTURE

- A **read cycle** is used to transfer data into the processor. During a read cycle
  - the processor first places the address on the address signals and then issues a read command on the control signals
  - the slave module (RAM, ROM, or I/O) will respond by placing contents at that address on the data signals
  - lastly the processor will accept the data and disable the read command.
- The processor uses a **write cycle** to store data into memory or I/O. During a write cycle
  - the processor also begins by placing the address on the address signals
  - the processor places the information it wishes to store on the data signals
  - the processor issues a write command on the control signals
  - the memory or I/O will respond by storing the information into the proper place
  - after the processor is sure the data has been captured, it will disable the write command.

Type	*Address Driven by*	*Data Driven by*	*Transfer*
Read Cycle	Processor	RAM, ROM or Input	Data copied to processor
Write Cycle	Processor	Processor	Data copied to output or RAM

Direct Memory Access (DMA)

COMPUTER ARCHITECTURE

- The **bandwidth** of an I/O interface is the number of bytes/sec that can be transferred.
- If we wish to transfer data from an input device into RAM, the software must first transfer the data from input to the processor, then from the processor into RAM.
- On the ARM, it will take multiple instructions to perform this transfer. The bandwidth depends both on the speed of the I/O hardware and the software performing the transfer.
- In some microcontrollers like the TM4C123 and TM4C1294, we will be able to transfer data directly from input to RAM or RAM to output using **direct memory access** (DMA).
  - During a **DMA read cycle** data flows directly from the memory to the output device.
  - During a **DMA write cycle** data flows directly from the input device to memory.
- When using DMA the software time is removed, so the bandwidth only depends on the speed of the I/O hardware.
- General purpose computers also support DMA allowing data to be transferred from memory to memory.

THANK YOU

Any Questions???