**IoT & Automation Lab.**

**Assignment#1**

1. **Prototype? Open-Source and Closed-Source Prototype Platforms:**

A prototype is an early sample, model, or release of a product built to test a concept or process.

* Open-source: Open source is source code that is made freely available for possible modification and redistribution.

E.g. **Arduino, Raspberry Pi.**

* Closed**-source: Closed-source model source code is not released to the public, i.e. it is not available on the public domains.**

E**.g. Google Earth, Skype, WinRAR, Microsoft Windows, Mac OS**

1. **Arduino?**

Arduino is a small popular electronic machine that makes it very easy for people to make electronic things.

It has two parts:

* a Circuit Board
* a program that lets people tell the circuit board what to do.

1. **Arduino Uno R3 Key Specifications:**

* Main Processor
  + ATmega328P, a modified Harvard architecture 8-bit RISC\* processor core. \**Reduced Instruction Set Computer*
* Memory (SRAM, FLASH MEMORY, EEPROM)
  + **SRAM:** Static Random Access Memory a type of RAM which uses a flip-flop to store 1-bit of data.
    - * The system's temporary data or run-time data is stored in the SRAM; with a size of 2KB.
  + **FLASH MEMORY:** In Arduino, the Flash stores the application code to be run.
    - * The Size of Flash Memory is 32KB.
  + **EEPROM:** An Electrically Erasable Programmable Read-Only Memory. It is a form of non-volatile memory that can remember things with the power being turned off, or after resetting the Arduino.
    - * The Size of EEPROM is 1KB.
* I/O Pins
  + **An Arduino has 14 digital input/output pins (of which 6 can be used as PWM\* outputs), 6 analog inputs.**

**Assignment#2**

* **What is an Encoding format? List down encoding formats for various types of data (Text, Number, Photo, Audio, Video).**

**~ Encoding Formats: A Brief Overview ~**

**Encoding format is essentially a standardized method to translate data into a format that computers can understand and process efficiently. It's like converting human language into a language computer can comprehend.**

* **Different Encoding Formats:**
* **Text Encoding**

**Text data is represented by characters, which are assigned specific numerical values. These values are then stored and transmitted in a specific format.**

**- ASCII (American Standard Code for Information Interchange): Represents 128 characters, including uppercase and lowercase letters, numbers, punctuation, and control characters.**

**- Unicode: A more comprehensive character encoding standard, capable of representing text in almost all languages. It includes ASCII as a subset.**

**- UTF-8: A variable-length encoding scheme compatible with ASCII, widely used for web pages and email.**

* **Number Encoding**

**Numbers are represented digitally using different numerical systems.**

**- Binary: Uses only 0s and 1s.**

**- Decimal: The base-10 system we commonly use.**

**- Hexadecimal: Uses 16 digits (0-9, A-F).**

**- Floating-point: Represents real numbers with a decimal point.**

* **Image Encoding**

**Images are represented by pixels, each with color information.**

**- JPEG (Joint Photographic Experts Group): Lossy compression, suitable for photographs.**

**- PNG (Portable Network Graphics): Lossless compression, suitable for images with sharp edges and text.**

**- GIF (Graphics Interchange Format): Supports animation and transparency, often used for simple images and logos.**

**- BMP (Bitmap): Uncompressed format, large file size.**

**- TIFF (Tagged Image File Format): Lossless compression, supports various image depths.**

* **Audio Encoding**

**Audio data is represented by digital samples of sound waves.**

**- MP3: Lossy compression, widely used for music.**

**- AAC (Advanced Audio Coding): Lossy compression, often used in iTunes**

**and digital broadcasting.**

**- WAV (Waveform Audio File): Lossless compression, high-quality audio format.**

**- FLAC (Free Lossless Audio Codec): Lossless compression, maintains audio quality.**

* **Video Encoding**

**Video data combines image and audio data.**

**- MP4 (MPEG-4 Part 14): Commonly used for video storage and distribution.**

**- AVI (Audio Video Interleave): Container format supporting various codecs.**

**- MOV (QuickTime Movie): Apple's video format.**

**- WMV (Windows Media Video): Microsoft's video format.**

**Assignment#3**

* Explain Basic Structure of an Arduino Program.
* There are two required parts or functions that enclose blocks of statements.
* **setup()** is the preparation, **loop()** is the execution.
* Both functions are required for the program to work.

void setup()

{

statements;

}

void loop()

{

statements;

}

* **The setup() function is called once**, when the Arduino board is first turned on or reset. It is used to initialize the board and set up the hardware.

The setup function should follow the declaration of any variables at the very beginning of the program. It is **the first function to run in the program**, is run only once, and is used to set **pinMode** or initialize serial communication.

Setting pinMode: This tells the Arduino **whether a specific pin is going to be used for input** (reading data) or **output** (sending data).

For example, if you have an LED connected to pin 13, you would specify in the setup() function that pin 13 is an output pin.

**Initializing Serial Communication:** This is like **opening a communication channel** between your Arduino and your computer or another device. This is useful for **sending data back and forth**.

* **The loop() function is called repeatedly,** until the Arduino board is turned off or reset. It is where the Arduino program does most of its work.

The loop function follows next and includes the code to be executed continuously – reading inputs, triggering outputs, etc.

**Assignment#4**

**1. The Architecture of Modern Computers**

**Parallelism in Modern CPUs**

Modern CPUs employ various techniques to achieve parallelism, including:

* **Pipeline:** Instructions are broken down into stages, and multiple instructions are processed simultaneously in different stages.
* **Superscalar:** Multiple instructions are executed simultaneously in a single cycle.
* **Multi-core:** A single CPU contains multiple cores, each capable of executing instructions independently.
* **SIMD (Single Instruction, Multiple Data):** A single instruction operates on multiple data elements simultaneously.

These techniques significantly improve performance but require careful software design to maximize their benefits. For example, programmers must ensure that instructions are not dependent on each other to avoid pipeline stalls.

**RISC vs. CISC Architectures**

* **RISC (Reduced Instruction Set Computing):** Uses a small set of simple instructions that can be executed in a single cycle. This leads to simpler and faster hardware but requires more complex software.
* **CISC (Complex Instruction Set Computing):** Uses a large set of complex instructions that can perform multiple operations in a single cycle. This simplifies software but requires more complex and slower hardware.

RISC architectures are generally preferred in modern systems due to their simplicity and efficiency. However, CISC architectures are still used in some legacy systems. The choice of architecture influences the design of operating systems, as the OS must be tailored to the specific instruction set.

**CISC vs. RISC and Von Neumann vs. Harvard Architectures**

* **CISC vs. RISC:** As discussed above, CISC has complex instructions while RISC has simpler ones.
* **Von Neumann Architecture:** Uses a single memory for both instructions and data. This is the most common architecture in modern computers.
* **Harvard Architecture:** Uses separate memories for instructions and data. This can improve performance but is more complex to implement.

The Von Neumann architecture is more common due to its simplicity and cost-effectiveness. However, the Harvard architecture can be advantageous in certain applications, such as digital signal processing, where there is a high demand for both instruction and data fetches.

**2. MicroControllers (e.g., Arduino Uno R3)**

**AVR vs. ARM Architecture**

* **AVR (Atmel RISC Architecture):** Used in the Arduino Uno R3. It is a 8-bit RISC architecture with a simple instruction set and low power consumption.
* **ARM Cortex-M:** A family of 32-bit RISC architectures used in more advanced microcontrollers. They offer higher performance, larger address space, and more features than AVR.

**8-bit vs. 16/32-bit Microcontrollers**

* **Memory Addressing:** 8-bit microcontrollers have a smaller address space, limiting the amount of memory they can access.
* **Processing Power:** 16-bit and 32-bit microcontrollers have higher processing power, making them suitable for more demanding applications.
* **Application Suitability:** 8-bit microcontrollers are well-suited for simple applications with limited memory and processing requirements, while 16-bit and 32-bit microcontrollers are better for more complex applications.

**3. Memory Segmentation**

**Memory Segmentation and Backward Compatibility**

Memory segmentation in x86 architecture divides the address space into segments, which can be used to isolate different processes or parts of a program. This allows for backward compatibility with older 16-bit software, which used segmented memory. However, segmentation can be inefficient and complex to manage.

**User Space vs. Kernel Space**

* **User Space:** The area of memory accessible to user applications.
* **Kernel Space:** The area of memory reserved for the operating system kernel.

This separation is important for security and isolation. It prevents user applications from accessing or modifying kernel code, which could lead to system instability or security breaches.

**Memory Segmentation in Older vs. Modern Systems**

* **Older Systems:** Memory segmentation was used extensively to manage memory and protect processes.
* **Modern Systems:** Segmentation is still used in some cases, but flat memory models are more common. Flat memory models provide a simpler and more efficient way to manage memory, especially in 64-bit systems.

**4. Memory Location and Byte Addressability**

**1-Byte Memory Locations**

Each memory location is generally 1 byte in modern computer systems because it is a convenient unit of data. Bytes can represent a wide range of values, from characters to small integers. This design choice simplifies memory addressing and data manipulation.

**Endianness**

Endianness refers to the order in which bytes are stored in memory.

* **Big-Endian:** The most significant byte is stored first.
* **Little-Endian:** The least significant byte is stored first.

Developers must be aware of the endianness of the system they are working with to ensure correct data interpretation. Misunderstanding endianness can lead to data corruption and errors.