Lecture1 & lab1
Questions & solutions

Lecture Part

1. What is Microprocessor:

A **Microprocessor** is an integrated circuit includes all or most of the functions of the Central Processing Unit (CPU).

2. What is CPU:

A CPU is a logic machine that can execute computer programs.

3. What is Computer Program:

A Computer Program is a **sequence of instructions** stored in the memory. Usually the execution of the instruction follows 4 steps: reading the instructions(**fetch**), decoding the instructions(**decode**), executing the instructions(**execute**), and writing the result(**write back**).

4. What is a Microcontroller:

The multiple components of the microprocessor-based systems form the Microcontroller when they are included in the same integrated circuit (Flash memories, Timers, counters, serial/parallel communication interfaces, etc).

5. What is the term that defines as to what instructions the microprocessor / microcontroller supports:

ISA – Instructions Set Architecture.

6. What family of microcontrollers does the Arduino MEGA 2560 development board use:

ATMega2560 microcontroller.

7. In terms of the SUPPORTED INSTRUCTIONS, what is the architecture of the microcontroller used by the Arduino MEGA? (RISC or CISC):

RISC (Reduced Instruction Set Compute).

8. What is the difference between CISC and RISC architecture

RISC: Simple, fast instructions; efficient for performance and power.
CISC: Complex, multi-step instructions; easier programming but potentially slower performance.

9. In terms of the STORAGE AND PATHWAYS FOR INSTRUCTION AND PROGRAM MEMORY, what is the architecture of the microcontroller used by the Arduino MEGA? (Harvard or Von Neumann):

Harvard Architectrue.

10. What is the difference between H & VN:

In a **Harvard architecture**, the microcontroller has separate memory spaces for program instructions and data. This means:

- **Program memory (Flash memory)** and **data memory (SRAM)** are physically separate.
- Instructions and data can be accessed simultaneously, which allows for faster execution in many cases.

This separation of program and data memory is a hallmark of the Harvard architecture, making it distinct from the **Von Neumann architecture**, where program instructions and data share the same memory space.

In a **Von Neumann architecture**, the microcontroller has a **single memory space** that stores both **program instructions** and **data**. This means:

- **Program memory** and **data memory** are **shared**, existing in the same physical memory space.
- Instructions and data are fetched through the **same bus** or pathway, meaning that the CPU accesses either instructions **or** data, but not both at the same time.

This shared memory space can lead to potential bottlenecks, as the CPU must alternate between fetching an instruction and reading/writing data, which can slow down execution.

11. What are the individual parts of the two-stage pipeline of the AVR:

Fetch & Execute

12. How many General Purpose Registers (GPRs) are there in the AVR:

32 - General Porpose Register (GPR).

13. Within which registers can you load immediate values:

In the last 16 registers : R1 -> R31

14. What are the DATA COPY register operations supported by the AVR:

MOV dist, src : Copy content of src into dist.

15. What are the IMMEDIATE register operations supported by the AVR:

LDI r16, 5 --------------- Load Immediate
ORI r16, 0xF0 --------- OR Immediate
ANDI r16, 0x80 ------- AND Immediate
SUBI r20, 1 ------------- Subtraction Immediate

16. What are the ALU based register operations supported by the AVR:

ADD r1, r2 --------------- Add

OR r3, r4 ----------------- OR
LSL r5 -------------------- Logic Shift Left
MUL **r5, r18** ------------ Multiplication
ROL r7 ------------------- Rotate Left
ROR r9 ------------------- Rotate Right
INC r19 ------------------ Increment
DEC r17 ------------------ Decrement

17. What type of memory is used for the Program Memory:

Flash Memory.

18. What 8-bit register contains information about the state of the microcontroller and about the result of operations:

I : global interrupt activation flag.

T : transfer bit, can be copied to and from register bits using the BLD and BST instructions.
H : half carry (carry between half bytes, used for BCD operations)
S : Sign bit, **N XOR V..**
V : overflow flag, indicates if the sign bit is changed due to overflow.
N : indicates a negative result
Z : indicates a null result.
C : carry.

Lab Part

1. What kind of MCU does the Arduino MEGA 2560 board use and on how many bits does it operate on:

Atmel ATMega 2560 MCU.

2. What are some of the communication protocols / interfaces (facilities) that the development board supports:

UART, SPI, I2C.

3. What is the speed (operating frequency) of the MCU:

16 MHz.

4. Explain the difference between pull-up and pull-down resistors:

Pull-up : resistors keep a pin high when it is not actively driven low.
Pull-down : resistors keep a pin low when it is not actively driven high.

5. What is the name of the defined flag that activates the internal resistors of the development board:

INPUT_PULLUP.

6. Why is a breadboard useful:
   a- Easy prototyping > Quick setup.
   b- Reuseable > Non-permenanant connections.
   c- No soldering reqired > User friendly.
   d- Multiple connection points.
   e- Debugging and Testing > Easy modification.

7. What is the parameter of the Serial.begin(VALUE) method:

Parameter: VALUE

- Type: long
- Description: This parameter specifies the baud rate for the serial communication. The baud rate is the rate at which information is transferred in a communication channel and is typically measured in bits per second (bps).

8. What is the method to set a certain pin on the dev board as input / output / input_pullup?

pinMode(pin, mode).
Modes :
INPUT: Sets the pin as an input.

OUTPUT: Sets the pin as an output.
INPUT_PULLUP: Sets the pin as an input with an internal pull-up resistor enabled

9. What is one method that you can use in order to get a value from a certain pin:

digitalRead(pin)

10. Explain the '<<' operator from C / C++:

The “<<” operator shifts the bits of the left operand to the left by the number of positions specified by the right operand. Each left shift effectively multiplies the number by 2.
Example :

```
stat = (b4<<5) // b4 is going to be shifted 5 bits to the left
```

11. Give an example of how to find out as to how long your Arduino based application has been running in SECONDS

use the built-in millis() function. This function returns the number of milliseconds since the Arduino board began running the current program. By dividing the value returned by millis() by 1000, you can convert milliseconds to seconds.

Important to know

Atmega 2560 microcontroller family – Technical features

- 135 instructions, most are executed in 1 clock cycle
- 32 general purpose 8 bit registers
- 256 K Bytes re-programmable flash memory
- 4K Bytes EEPROM
- Internal SRAM 8K Bytes
- Read/write cycles: 10,000 Flash/100,000 EEPROM
- Up to 64 KB RAM addressable locations (if external RAM is used)

Integrated peripherals

- Two 8-bit timer/counters
- Four 16 bit timer/counters
- 4 PWM channels (8 bit), 12 PWM channels (16 bit)
- 16 Analog/Digital conversion channels (10 bit)
- 4 programmable USART interfaces
- 1 SPI interface
- Two Wire Interface (TWI), similar to I2C
- Interrupt generation by pin state change detection

Lecture2 & lab2
Questions & solutions

Lecture sesion

1. What are the Input / Output ports for the Arudino Mega: (

A B C D E F G H J K L

2. What register would you use to configure a certain port as either input or output:

DDRx – **Data Direction Register**, Determines whether the corresponding pins of port x are set as inputs or outputs (1 for output, 0 for input).

3. What register would you use to write to a certain port:

PORTx **-** Used to **write data** to the output pins of the corresponding port (i.e., setting pin values to HIGH or LOW).

4. What register would you get a certain port's state:

PINxn : x is the port name, n is the number of the pin in that port, Used to **read the input** values from the corresponding port's pins.

Lab sesion

1. How would you configure all the pins of port A as input:

DDRA = 0x00.

2. How would you configure all the pins of port A as output:

DDRA = 0xFF.

3. How would you configure the lower half of port B as output and the upper half of it as input:

DDRB = 0x0F

4. How would you read data from port A:

set all pins as input
assign value : DDRA = 0x00;
assign the pin you need to read from to a variable :
char a = digitalRead(PINAn);

5. Give an example of transmitting data from port A of the MCU to a peripheral

8 LEDs are hooked up to the port and to make half of them light up:
DDRA = 0xFF;

PORTA = 0b10101010;)

6. What is the operator for the "BITWISE or" operation in the C / C++ programming language:

The **bitwise "OR"** operator compares each bit of its operands. If **either** corresponding bit of the operands is 1, the resulting bit is set to 1. Otherwise, the resulting bit is set to 0.

7. What is the operator for the "BITWISE XOR" operation in the C / C++ programming language:

The **bitwise "XOR"** operator compares each bit of its operands. If the corresponding bits of the operands are **different**, then the resulting bit is set to 1. If both corrisponding bits are the same, then the resulting bit is set to 0.

8. Explain the following line of code: "cdigit ^= 1;" (Same as "cdigit = cdigit ^ 1;")

performs a bitwise XOR between cdigit and 1. XOR compares each corresponding bit of the operands, and if the bits are different, the result is 1. If the bits are the same, the result is 0.

9. Explain the terms LSB and MSB, what does the acronym stand for and what is the meaning behind it:

LSB : Least Segnificant Bit – leftmost bit in a binary number
MSB : Most Segnificant Bit – rightmost bit in a binary number