GARRAIO
EMPOWERED BY INNOVATION

Lab 4

C Programming

1. Write a C program that defines macros for basic arithmetic operations (addition, subtraction, multiplication, division) and uses these macros to perform calculations.

2. Create a header file that contains function declarations and includes guards. Write a C program that includes this header file multiple times in different source files to demonstrate how include guards prevent duplicate inclusion. Show “*.i” file when header guards exist and not exist.

3. Define macros that convert numeric constants to string literals using the stringizing operator (#) and concatenate strings using the token pasting operator (##).

4. Write a C program that uses preprocessor macros to conditionally compile code for different platforms or architectures. For example, define different behavior based on the target platform (Windows, Linux, etc.).
5. Create a C program that implements a customizable logging system with different log levels (DEBUG, INFO, WARNING, ERROR). The user should be able to enable or disable specific log levels using preprocessor directives.
    a. Define preprocessor macros for different log levels (DEBUG, INFO, WARNING, ERROR).

b. Define a macro for the current log level (e.g., LOG_LEVEL) that the user can customize to control the verbosity of logging messages.
c. Implement logging functions for each log level (e.g., log_debug, log_info, log_warning, log_error). These functions should take a message as input and print the log message to the console if the log level is enabled.
d. The logging functions should also include the file name and line number in the log messages to help identify the source of the logs.

6. Bit Manipulation:
    a. Write a C macro that reads a particular bit position in a register.
    b. Generalize your macro in (a) so that extracts a group of bits at a given offset in a register.
    c. Write a C macro that inserts a value in a particular bit position in a register,
    d. Generalize your macro in (c) so that it inserts a value in a group of bits at a given offset in a register.
    e. Write a C variadic function that sets any number of bits in a register knowing its memory mapped address.
    f. Is it better to implement bit manipulation operations as C functions or macros? Justify your choice.

7.

a. You were asked by your boss to implement an interrupt controller driver for a 32-bit microcontroller. Your first task is very simple; you need to write a function that enables a particular interrupt given its interrupt number. The function prototype is as follows:

void enableInterrupt(uint8_t interruptNumber);

There are a total of 100 interrupts assigned numbers from 0 to 99. To enable an interrupt, there are four registers with the names of NVIC_ISER0, NVIC_ISER1, NVIC_ISER2 and NVIC_ISER3 where each register can be used to enable 32 interrupts out of the 100. This means that if you want to enable interrupt number 3, you need to set bit 3 in register NVIC_ISER0, and if you want to enable interrupt number 45, you need to set bit 13 in NVIC_ISER1, and so on.

Implement enableInterrupt that takes the interrupt number as an argument and enables it by setting the corresponding bit in the corresponding register.

b. You completed your first task, and your boss is happy with your work. So, he decides to give you another one. You now need to write a function that sets the priority of a particular interrupt given its interrupt number and priority. The prototype is as follows:

void setInterruptPriority(uint8_t interruptNumber, uint8_t priority);

The interrupt controller has 25 priority registers that are named NVIC_PRI0, NVIC_PRI1, NVIC_PRI2, ..., NVIC_PRI23, NVIC_PRI24. Each register is 32 bits wide and is used to set the 8-bit priority value of 4 interrupts. This means that:

- if the interrupt number mod 4 = 0, then the priority of this interrupt is set in bits 7:0 of the corresponding register.
- if the interrupt number mod 4 = 1, then the priority of this interrupt is set in bits 15:8 of the corresponding register.
- if the interrupt number mod 4 = 2, then the priority of this interrupt is set in bits 23:16 of the corresponding register.
- if the interrupt number mod 4 = 3, then the priority of this interrupt is set in bits 31:24 of the corresponding register.

Implement the function setInterruptPriority that takes the interrupt number and the priority as arguments and sets the priority of the interrupt in the corresponding register.

c. Does your implementation of (b) make the function reentrant? Why or why not?

8. You are developing an application on an 8-bit microcontroller, and you need to access a register with a datasheet name of UART_STATUS_R and has the following structure:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
frame_error	ovf_error	Reserved			State		busy

a. Knowing that it is memory mapped to an address of 0x4800'0000, write an object-like macro that can be used to access this register, and then use it to read the 'state' bits.

b. Modify your solution in (a) so that there's no need to use bit manipulation techniques to read the 'state' bits. (Hint: use bit fields)

c. Can you combine both solutions in (a) and (b) so that you have the options to either access the register as a whole or access a particular bit field?

d. You've discovered that the UART peripheral has more registers and that their memory mapping is as follows:

Register Name	Address
Status Register	0x4800’0000
Data Register	0x4800’0008
Control Register	0x4800’0009

Try to find a way to combine the registers of the UART peripheral into a single C structure so that the following C statement works:

```
uint8_t state_value = UartRegisters.StatusRegister.state;
```