Lab 3: Baremetal Software on TM4C123 ARM Cortex-M4

Title: Let us learn everything together with an example!

Objective

This lab aims to explore the process of writing baremetal software for the TM4C123 ARM Cortex-M4 microcontroller. We will toggle GPIO PortF pin 3 (PF3), which controls the green LED, using fundamental embedded programming concepts.

Scope of Work

In this lab, we will:

1. Write software from scratch without using an IDE.
2. Develop and understand key embedded systems components:
    - **Startup.c**: Initializes hardware and jumps to the main function.
    - **Linker.ld**: Defines memory layout and section placements.
    - **main.c**: Contains the application logic to toggle PF3.
    - **Makefile**: Automates the build process.
3. Learn and apply debugging techniques using GDB and related clients to analyze software execution in depth.

Procedure

1. Enabling GPIO PortF Pin 3

To toggle PF3, we need to configure the GPIO registers as follows:

- Enable the GPIO clock:

```
SYSCTL_RCGC2_R |= 0x00000020;
```

- Configure PF3 as an output:

```
GPIO_PORTF_DIR_R |= (1 << 3);
```

- Enable digital functionality for PF3:

```
GPIO_PORTF_DEN_R |= (1 << 3);
```

- Toggle the PF3 pin:

GPIO_PORTF_DATA_R ^= (1 << 3);

2. Writing Baremetal Components

Startup.c

- Initializes the stack pointer and clears the .bss section.
- Copies the .data section from Flash to SRAM.
- Sets up the vector table for exception handling.
- Transfers control to main().

Linker.ld

- Defines memory regions for Flash and SRAM.
- Places sections like .text, .data, and .bss in appropriate regions.

main.c

- Implements the application logic to toggle the green LED.
- Configures GPIO registers for PortF.
- Uses a delay loop to produce a visible toggle effect.

Makefile

- Automates the compilation and linking process.
- Includes targets for building, cleaning, and debugging the project.

Debugging

We explored debugging using GDB to:

- Set breakpoints and observe program flow.
- Inspect memory and registers in real-time.
- Debug startup routines and application logic.

Lessons Learned

1. **Baremetal Programming Concepts**:

- How to configure GPIO registers and understand the memory-mapped I/O system.
- Importance of startup routines and linker scripts in embedded systems.

2. **Memory Layout**:
    - Understanding the placement of sections like .text, .data, and .bss.
3. **Debugging**:
    - Gained proficiency in debugging embedded applications using GDB.
4. **Workflow Automation**:
    - How to use a Makefile to simplify the build process for embedded projects.