Agenda

- ARM7 Exception handling
- Setup ARM Toolchain
- LPC1768 overview
- LPC1768 BlueBoard overview
- GPIO programming
    - LED
    - Buzzer
    - LCD

Bitwise Operators

- To check nth bit
    - regr & BV(n)
- To set nth bit
    - regr |= BV(n);
- To clear nth bit
    - regr &= ~BV(n);
- To toggle nth bit
    - regr ^= BV(n);

ARM7 Architecture

- Barrel Shifter: Combinational cct to do shift operations on 2nd operand before ALU operations.
- 32-bit ALU: doing all operations on registers only.
- Separate Multiplier -- takes 3-5 CPU cycles.
- Single 32-bit address bus and 32-bit data bus.
- Scan control for debug/ICE.
- Thumb decoder to convert from Thumb instruction into ARM instruction.
- Single IRQ line and single FIQ line.

ARM7 Exception Handling

- ARM7 have 7 exceptions i.e. Reset, Software Interrupt, Prefetch Abort, Data Abort, IRQ, FIQ, Undef.
- IRQ and FIQ are interrupts from peripherals. When I=1, IRQs are disabled and when F=1, FIQ is disabled.
- Exception priorities

    1. Reset -- I=1 and F=1
    2. Data Abort -- I=1
    3. FIQ -- I=1 and F=1
    4. IRQ -- I=1
    5. Prefetch Abort -- I=1
    6. Software Interrupt -- I=1
    6. Undef -- I=1

- For each exception, an exception handler should be implemented. Exception handler is set of instructions (routine) that handles the exception.
- The exception handler addresses must be stored in EVT (exception vector table). Default location of EVT is 0x00000000 (of Flash). Each entry is an instruction of 32-bits.
- Exception vector
    - 0x00: ResetHandler
    - 0x04: UndefHandler
    - 0x08: SwiHandler
    - 0x0C: PrefetchAbtHandler
    - 0x10: DataAbtHandler
    - 0x14: Reserved
    - 0x18: IRQHandler
    - 0x1C: FIQHandler

ARM7 Interrupt Handling

- There are two types of interrupts i.e. IRQ and FIQ.
- IRQ and FIQ are special exception types. Hence basic exception handling flow is same for IRQ as well as FIQ.
- Typical ARM7 chip have VIC that interface multiple peripherals to IRQ & FIQ lines of core.
- To enable Interrupt in VIC (e.g. UART -- Peripheral No UART_IRQ=4)

```
// to enable UART interrupt in VIC
VICVectCntl3 = BV(5) | UART_IRQ;
VICVectAddr3 = uart_isr;
VICIntSelect &= ~BV(UART_IRQ);
VICIntEnable |= BV(UART_IRQ);

// to enable interrupt in UART
IER = BV(RDR);
```

- By default nested interrupts are disabled. However IRQHandler can be implemented to enable them back.

ELF format

- Executable file format for UNIX/Linux.

- It contains multiple sections (sectioned binary).
    - Text: machine level instructions (for user code and library)
    - Data: initialized global & static variables.
    - BSS (Block Started by Symbol): uninitialized global & static variables.
    - RoData: String constants "..."
    - Symbol Table: Contains info about symbols (var/fn names).
        - name, address, flags (F/D), section, size
    - Exe header
        - Magic number
            - It is 2/4 bytes number identifying the file format of binary file. It is always at the start of file.
            - It is verified during loading/reading that file by the software.
            - The exe file is loaded in the RAM by the loader (part of OS). While loading it verifies magic number and if not matching abort the process.
        - Address of entry point function i.e. main()
        - Info about other sections

```
char *p = "Sun";
*p = 'F';
puts(p);
```

```
readelf -a hello.out

objdump -h hello.out

objdump -S hello.out

objdump -t hello.out
```

Toolchain

- Compilation is done using compiler software. It contains set of tools (e.g. preprocessor, compiler, assembler, linker) which are executed one after another. These compiler software are also called as "Toolchain".
- Toolchain also contains additional utilities like debugger, objcopy, objdump, ...

Compilation

- On x86, compile program that will run on x86. This is called as "Native" Compilation.
- On x86, compile program that will run on ARM/AVR/... This is called as "Cross" Compilation.
- For cross compilation we need toolchain corresponding to that architecture.

ARM cross-compilation toolchain

- Keil
    - IDE + toolchain
    - Commercial
- CodeSourcery
    - IDE + toolchain
    - Commercial
- GNU toolchain
    - toolchain
    - Free/Open source

ARM GNU Toolchain

- arm-none-eabi -- Baremetal programming.
- arm-linux-gnueabi
- arm-linux-gnueabihf

Installation

- Download toolchain.
- Unzip toolchain tarball.
    - tar xvjf gcc-arm-none-eabi-xyz.tar.bz2

Skeleton program (_basic)

- Makefile
    - Set GCC_BASE = /path/of/toolchain/
    - Set PROJECT_OBJECTS as appropriate
- main.c
    - User program.
- startup_LPC17xx.c
    - Vector table
    - ResetHandler
        - calls main()
- system_LPC17xx.c
    - SystemInit()
- CMSIS library headers
    - Cortex Microcontroller Software Interface Standard
- LPC17xx.h
    - All structures/members representing IO peripherals.

    - Addresses of all IO peripherals (memory mapped IO).
- LPC1768.ld
    - Linker script: Which sections are to be placed on which addresses.