University of Dhaka

Department of Computer Science and Engineering

CSE-3103 : Microprocessor Lab

Lab Report 6:

bare-metal programming:Development of lower-level system library for unrestricted programming

Submitted By:

1. Syed Mumtahin Mahmud, Roll: 50

2. Nazira Jesmin Lina, Roll: 55

Submitted On :

July 22, 2023

Submitted To :

Dr. Mosaddek Hossain Kamal

Dr.Upoma Kabir

Contents

1 Definition of Compiler Flags:

- **-c:** Used to instruct the compiler to assemble the source files without linking. Since the linking in this project uses a custom linker script, this flag has been used to generate *.o (Object) files, which will then be passed to the next phase (linking) with flags of its own.
- **-mcpu:** Used to state the architecture of the cross-compilation target. The host machine here is different from the target the program is being built for, and for that reason, the -mcpu flag needs to be supplied with the desired target architecture, which in this case is "cortex-m4".
- **-mthumb:** Tells the compiler to compile to Thumb instructions instead of ARM instructions. The device in question, STM32F446, only supports Thumb mode.
- **-Wall:** This instructs the compiler to generate all the warnings. This is particularly useful for writing better code and keeping potential ambiguous behavior in check.
- **-std=gnu11:** Tells the compiler to use the GNU11 standard of C.
- **-O0:** This is used to tell the compiler to set the optimization level to 0 (Do not optimize). While increasing optimization levels improves performance/size, it also increases compile time and makes debugging harder due to stripping away information related to debugging to generate a smaller binary.

2 Definition of Linker Flags:

- **-nostdlib:** Disables linking the ARM standard library to use the C/AEABI standard library instead.
- **-T stm32_ls.ld:** Specifies the linker script to use for linking. Here, "stm32_ls.ld" is the file path of the linker script.
- **-Wl:** The -Wl tells the compiler to parse the following flag (separated by a comma) to be parsed as a flag for the linker instead of parsing it as a flag of the compiler since
- **-Map:** -Map is not recognized by the compiler. -Map generates a map that contains various information about the linking process, with specific section information.

3 Syntax and Semantics of the Linker Script:

1. **ENTRY(_symbol_name_):** The ENTRY command is used to denote the entry point of the entire program. This information is stored in the header of the final ELF file. In this case, the entry point is the symbol "Reset_Handler," which is essentially a function in the higher-level code. This function is executed whenever the processor goes through a reset.
2. **SECTIONS:** `SECTIONS`
   `{`
   `<section_name>:`
   `{`
   `/* Instructions related to merging go here */`
   `} ¿ <VMA>` AT `<LMA>`
   `}`

The SECTIONS command gives instructions on how different sections in the object files will be merged into the final ELF file. VMA and LMA stand for "Virtual Memory Address" and "Load Memory Address," respectively. The AT `<LMA>` portion is optional, and it is assumed that VMA == LMA if it is omitted.

3. `MEMORY:`
   Syntax:

```
MEMORY
  {
  name(attr): ORIGIN = mem_region_origin_addr, LENGTH =
  mem_region_len
  }
```

   The MEMORY command is used to define the memory regions as per the device memory map. It specifies the name of the memory region, its origin address, and its length.

4 Explanation of Sections in .ld File with Respect to GCC:

4.1 .text

This section holds the code. The .isr_vector and the .rodata also get merged into this section, meaning that it also holds the vector table and read-only data (constants).

4.2 .data

This section holds the initialized data. Usually, the data here is copied from FLASH to the RAM for faster accesses.

4.3 .bss

Finally, this section is for the uninitialized data in the RAM. Maintaining the sizing of this section is crucial for better memory management.

5 ALIGN and Location Counter:

5.1 ALIGN

If certain sections defined in the linker script have a total size not divisible by word size, the end boundary for that section might become unaligned, which can result in inefficient accesses. To mitigate the issue, we use the ALIGN(size) instruction before the end location counters so that it's padded accordingly to comply with aligned word accesses.

5.2 Location Counter

The location counter (.) is a special linker symbol that automatically gets updated with address information. It is used to track section boundaries and can supply the said information to external entities that might need it. It should only appear within the SECTIONS command.

6 Additional Explanations

6.1 Memory Mapping from FLASH to SRAM and Absolute Address Generation

Instructions in the Reset_Handler() function in the startup code copy individual bytes from the .data section in FLASH to the SRAM.

```
uint32_t size = (uint32_t)&_end_data - (uint32_t)&_start_data;
uint8_t *dst = (uint8_t *)&_start_data; // SRAM
uint8_t *src = (uint8_t *)&_end_text; // FLASH

for (uint32_t i = 0; i < size; i++)
{
    *dst++ = *src++;
}
```

This is done for faster access to variables that possibly require the most disk accesses while the program is running.

6.2 Absolute Address Generation

By default, without linking, the addresses for various sections don't have their proper addresses and might end up starting from conflicting addresses. This needs to be resolved via relocation, which is why these sections are referred to as relocatable sections. The linker uses the boundary information given to it via the linker script and generates the absolute addresses for each section in the ELF file.

6.3 Address of GPIOx and RCC and How to Determine Them

The addresses for the GPIOx registers and the RCC registers are device-specific and are mentioned in the reference manual of the device. For STM32F446xx, the boundary addresses of the peripherals are mentioned on page 57. Addresses of each individual register are then calculated by adding the offset to the base address of the peripheral.

6.4 'weak' Definition of the Functions and Section Attributes (.isr_vector) for the NVIC Array

The 'weak' definition of a function allows having a default aliased handler for the function without having to implement each and everyone separately. The application code can override the specific handlers it needs in order to define custom interrupt behavior.