Assembly Code for Cortex-M

Experiment: Running Assembly on Cortex-M (QEMU + GDB)

1. Files Used

- `foo.s` → Assembly source file.
- `map.ld` → Linker script (defines memory layout).
- `Makefile` → Automates build, linking, and debugging.

---

2. Flow of Execution

1. `foo.s` → fed into **assembler** → generates **foo.o** (object file).
2. `foo.o` + `map.ld` → fed into **linker** → generates **foo.elf** (executable + metadata).
3. `foo.elf` → run on **QEMU** (simulating Cortex-M3 STM32 board).
4. **GDB** is used to **debug/step** through code (`ni` for next instruction).

Automation:

- `make qemu` → builds and runs QEMU.
- `make gdb` → launches GDB and attaches to QEMU.
- `make clean` → removes build artifacts.

---

3. Linker Script: `map.ld`

- Defines memory region:

```
MEMORY {
   MEM : ORIGIN = 0x0, LENGTH = 0x4000
}
```

- Places sections into memory:

```
SECTIONS {
    .text : {
        *(.vectors*)   // vector table
        *(.text*)      // program code
    } > MEM
}
```

👉 This tells the **linker where each section should go in memory**.

---

4. Assembly File: `foo.s`

```
.section .vectors
vector_table:
    .word 0xABC0          // Initial value for Stack Pointer (SP)
    .word reset_handler   // Reset Vector (address of Reset Handler)
    .zero 400             // Reserve 400 bytes

.section .text
.align 1
.type reset_handler, %function
reset_handler:
    mov r1, #0x1
    mov r2, #0x2
    add r3, r1, r2
    bl .                  // Branch to itself (infinite loop)
```

🔎 **Explanation**:

- `.section .vectors` → defines **vector table**.
  - `0xABC0` → loaded into **R13 (SP)**.
  - `reset_handler` → loaded into **PC** at boot.
  - `.zero 400` → reserves extra space.

- `.section .text` → holds actual program instructions.
- `reset_handler` → simple function: load values, add, loop.

---

5. ELF & Debugging

- **foo.elf** → main executable (binary + symbol info).
- **foo.elf.lst** → disassembly with memory addresses (shows actual PC values assigned).
- **foo.elf.debug** → full ELF metadata (readelf output).

👉 When QEMU runs:

- **SP = 0xABC0** (from vector table).
- **PC = reset_handler address** (from vector table).
- CPU begins executing instructions at that location.

---

6. Key Notes

- **Sections** (`.vectors`, `.text`, etc.) tell the **linker where to place code/data**.
- **Makefile** automates compiling, linking, disassembly, and debugging.
- Using `ni` in **GDB**, we step instruction-by-instruction through program flow.