ARM THUMB INTERWORKING

Interworking between ARM and Thumb modes allows a program to switch between the 32-bit ARM instruction set and the 16-bit (or mixed 16/32-bit in Thumb-2) Thumb instruction set. This capability is useful for optimizing code density while maintaining performance and flexibility. Interworking is achieved using specific instructions and careful management of function calls and returns.

Interworking Instructions

The ARM architecture provides instructions specifically designed for switching between ARM and Thumb modes:

1. **BX (Branch and Exchange): This instruction is used to branch to an address and switch modes based on the least significant bit (LSB) of the target address:**
    - **If the LSB is `0`, the processor switches to ARM state.**
    - **If the LSB is `1`, the processor switches to Thumb state.**

BX R0 ; Branch to the address in R0, switching to ARM or Thumb based on LSB

2. BLX (Branch with Link and Exchange): This instruction is used to call a subroutine and switch modes based on the LSB of the target address. It also stores the return address in the link register (LR).

Example of ARM and Thumb Interworking

Here's an example that demonstrates how to switch between ARM and Thumb modes using interworking instructions.

<u>ARM Code:</u>

```
    .syntax unified

    .arm

    .global main



main:

    MOV R0, #1          ; Load 1 into R0

    BL switch_to_thumb       ; Call Thumb function

    MOV R0, #2          ; Load 2 into R0 (execution returns here)

    B end               ; Branch to end



switch_to_thumb:

    LDR R1, =thumb_func + 1 ; Load the address of thumb_func with LSB set to
1 (Thumb mode)
```

```
    BX R1                  ; Branch to thumb_func in Thumb mode


end:

    B end                  ; Infinite loop to end the program
```

Thumb Code:

```
    .syntax unified

    .thumb

    .thumb_func


thumb_func:

    MOV R0, #3             ; Load 3 into R0

    BLX return_to_arm          ; Call ARM function and switch to ARM mode

    B .                ; Infinite loop to end the Thumb code


return_to_arm:

    LDR R2, =main              ; Load the address of main (ARM mode)
    BX R2                   ; Branch to main in ARM mode
```

Explanation

1. **<u>In the ARM Code:</u>**
   - `main`: **The main function starts in ARM mode.**
   - `MOV R0, #1`: **Load the value 1 into register R0.**
   - `BL switch_to_thumb`: **Call the** `switch_to_thumb` **function, storing the return address in LR.**
2. **<u>Switching to Thumb Mode:</u>**
   - `switch_to_thumb`: **The function to switch to Thumb mode.**
   - `LDR R1, =thumb_func + 1`: **Load the address of** `thumb_func` **with the LSB set to 1 to indicate Thumb mode.**
   - `BX R1`: **Branch to** `thumb_func`**, switching the processor to Thumb mode.**
3. **<u>In the Thumb Code:</u>**
   - `thumb_func`: **The Thumb function starts in Thumb mode.**
   - `MOV R0, #3`: **Load the value 3 into register R0.**
   - `BLX return_to_arm`: **Call the** `return_to_arm` **function, storing the return address in LR and switching to ARM mode.**
4. **<u>Returning to ARM Mode:</u>**

  - `return_to_arm`: **The function to switch back to ARM mode.**
  - `LDR R2, =main`: **Load the address of the** `main` **function (ARM mode).**
  - `BX R2`: **Branch to** `main`**, switching the processor back to ARM mode.**

<u>Important Considerations</u>

- **LSB of Addresses: When branching to an address, the LSB determines the mode.** `0` **for ARM mode and** `1` **for Thumb mode.**
- **Instruction Alignment: ARM instructions are 32-bit aligned, while Thumb instructions are 16-bit aligned. Ensure proper alignment when switching modes.**
- **Link Register (LR): Manage the LR carefully, especially when interworking between ARM and Thumb modes, to ensure correct return addresses.**

<u>Summary</u>

Interworking between ARM and Thumb modes allows for efficient use of code density and performance. By using instructions like `BX` **and** `BLX`**, you can switch between modes and make function calls across different instruction sets. Proper handling of the LSB of addresses and careful management of the link register are crucial for successful interworking.**