THUMB REGISTERS IN ARM

In ARM architecture, when the processor is in Thumb mode, the set of registers available for use is a subset of the full ARM register set. Thumb mode still uses the same 16 general-purpose registers as ARM mode, but not all of these registers are accessible for every Thumb instruction.

General-Purpose Registers

ARM architecture defines 16 general-purpose registers (R0-R15). Here's a breakdown of their usage in Thumb mode:

1. **R0-R7: These are the low registers and are fully accessible in Thumb mode.**
2. **R8-R12: These are the high registers and are accessible in certain Thumb-2 instructions.**
3. **R13 (SP): The stack pointer, used for stack operations.**
4. **R14 (LR): The link register, used to store return addresses for subroutine calls.**
5. **R15 (PC): The program counter, which holds the address of the current instruction being executed.**

Usage in Thumb Mode

In the original Thumb (Thumb-1) instruction set, most instructions can only use the low registers (R0-R7). However, with the introduction of Thumb-2, many instructions gained the ability to use the high registers (R8-R12) as well.

Low Registers (R0-R7)

These registers are commonly used for holding data and are accessible by most Thumb instructions.

```
MOV R0, #1         ; Move immediate value 1 into R0

ADD R1, R0, #2   ; Add immediate value 2 to R0 and store the result in R1
```

High Registers (R8-R12)

These registers are accessible in certain instructions in Thumb-2 mode. They are typically used in more complex operations or when more registers are needed for computations.

```
.thumb

.thumb_func



MOV R8, #3         ; Move immediate value 3 into R8 (Thumb-2 instruction)

ADD R9, R8, #4   ; Add immediate value 4 to R8 and store the result in R9
```

Special Registers

R13 (SP): **The stack pointer is used for stack operations such as pushing and popping registers.**

```
PUSH {R0-R3}       ; Push R0-R3 onto the stack

POP {R0-R3}        ; Pop R0-R3 from the stack
```

R14 (LR): The link register holds the return address for subroutine calls. It is typically used with the BL (Branch with Link) and BX (Branch and Exchange) instructions.

```
BL my_function   ; Branch to my_function, storing return address in LR

BX  LR        ; Return from my_function
```

R15 (PC): The program counter holds the address of the current instruction. In Thumb mode, branching instructions can directly modify the PC.

```
B my_label             ; Branch to my_label, updating the PC
```

Example Program Using Thumb Registers

Here's an example program that demonstrates the use of low and high registers in Thumb mode:

```
        .syntax unified

        .thumb

        .global _start
```

```
_start:

        MOV R0, #5          ; Load immediate value 5 into R0

        MOV R1, #10         ; Load immediate value 10 into R1

        ADD R2, R0, R1      ; Add R0 and R1, store result in R2



        MOV R8, #3          ; Load immediate value 3 into R8 (Thumb-2 instruction)

        ADD R9, R8, #4      ; Add immediate value 4 to R8, store result in R9



        PUSH {R0-R3}        ; Push R0-R3 onto the stack

        POP {R4-R7}         ; Pop R0-R3 from the stack into R4-R7



        BL subroutine       ; Call subroutine, storing return address in LR



end_label:

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

```
subroutine:

    ADD R0, R0, #1      ; Increment R0 by 1

    BX LR           ; Return from subroutine
```

Summary

- **Low Registers (R0-R7): Fully accessible in all Thumb instructions.**
- **High Registers (R8-R12): Accessible in Thumb-2 instructions.**
- **Special Registers (SP, LR, PC): Used for specific purposes like stack operations, subroutine calls, and branching.**

Understanding the usage of these registers in Thumb mode is crucial for writing efficient and effective ARM assembly code, especially for embedded systems where memory and performance are critical.