# Unit 2- ARM Processors and Peripherals

## STACKS AND SUBROUTINES

Stacks and subroutines are fundamental concepts in ARM processors, playing a critical role in function calls and memory management. Here's an overview of each:

**STACKS IN ARM PROCESSOR:**

- A stack is a data structure used for storing temporary data such as function parameters, return addresses, local variables, and register values.
- In ARM processors, the stack is typically implemented in memory and grows either upwards (low to high addresses) or downwards (high to low addresses), depending on the convention.

**Key Concepts of the Stack in ARM:**

**1.Stack Pointer (SP):**

•ARM uses the SP register (R13 in the ARM register set) to point to the top of the stack.

**2.Push and Pop Operations:**

•**Push:** To add (store) data to the stack, the stack pointer is decremented (for a descending stack) and the value is stored at the new address.

•**Pop:** To remove (retrieve) data from the stack, the value at the address pointed to by the stack pointer is read, and the stack pointer is incremented.

**3.Stack Conventions:**

i. **Full Descending:** Commonly used; the stack grows downward, and the SP points to the last valid item.

ii. **Empty Ascending:** The stack grows upward, and the SP points to the next free location.

**4.Usage:**

a) **Subroutine calls:** Save the return address and other temporary values.

b) **Interrupt handling:** Save the processor state.

**5.Instructions:**

ARM processors use instructions like PUSH and POP (in Thumb mode) or memory access instructions like STMFD (Store Multiple Full Descending) and LDMFD (Load Multiple Full Descending) for stack operations.

## Subroutines in ARM Processors

- A subroutine is a reusable block of code that can be called from multiple points in a program. Subroutines allow modular programming and help reduce code duplication.

**Key Concepts:**

**1.Branch with Link (BL):**

•The BL instruction is used to call a subroutine.

•It saves the address of the next instruction (the return address) in the LR (Link Register, R14).

•The processor then jumps to the subroutine address.

**2.Returning from a Subroutine:**

•Use the BX LR instruction to return to the address stored in the LR.

**3.Saving Registers:**

•Before using registers in a subroutine, their current values are often saved on the stack.

**EXAMPLE:**

```
PUSH {R0, R1, LR}   ; Save registers R0, R1, and the link register
...
POP {R0, R1, PC}    ; Restore registers and return (pop PC = return to LR)
```

**Parameter Passing:**

- ARM typically uses registers R0–R3 to pass the first four arguments to a subroutine.
- Additional arguments are passed on the stack. Example:

```
my_function:
PUSH {R4, LR}        ; Save R4 and return address
ADD R4, R0, R1       ; Perform some operation
MOV R0, R4           ; Set the return value in R0
POP {R4, PC}         ; Restore R4 and return
```

## Stacks and Subroutines Example:

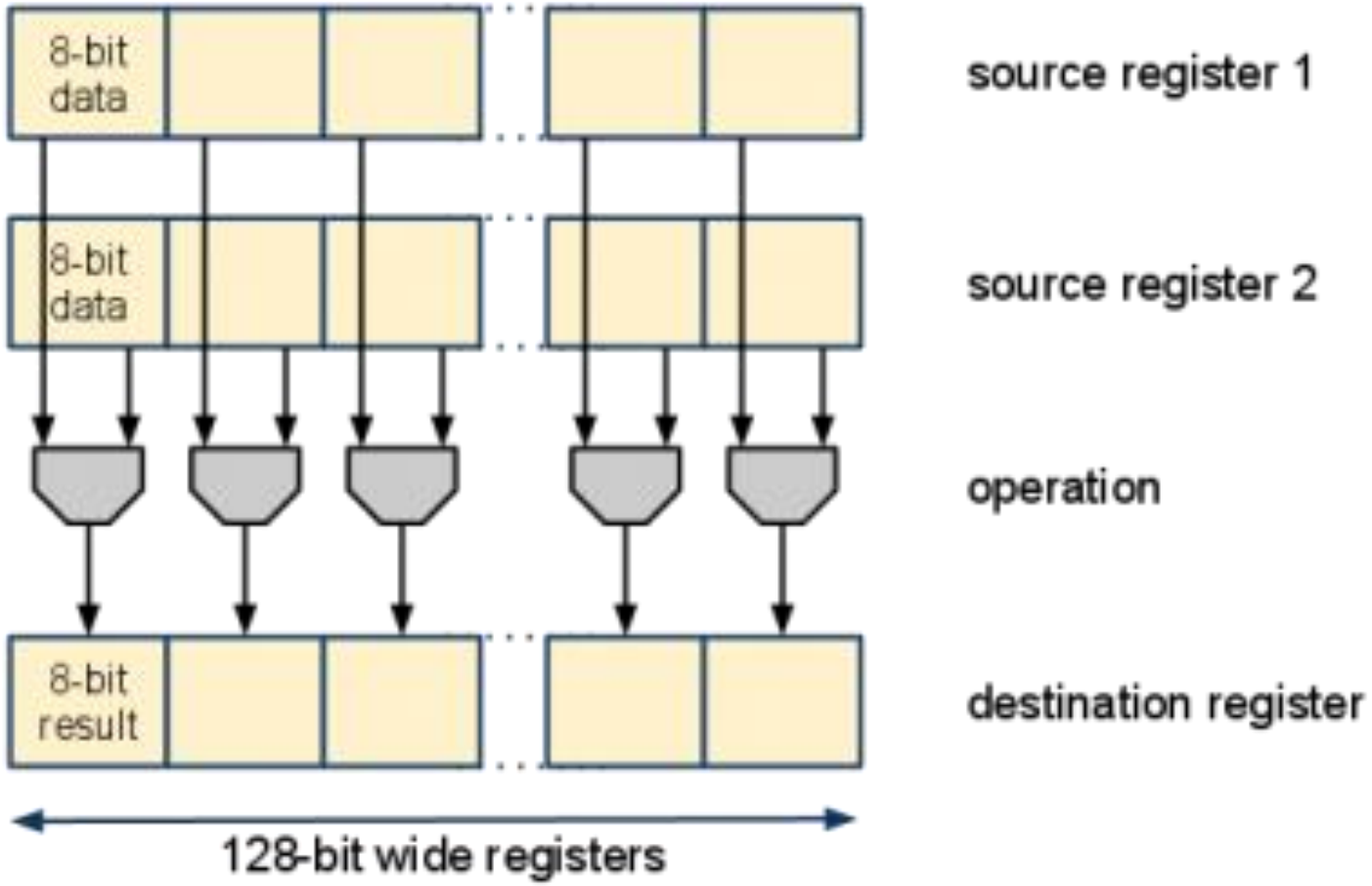
```
main:
    MOV R0, #5              ; Argument 1
    MOV R1, #10             ; Argument 2
    BL add_numbers          ; Call subroutine
    ; Result in R0
    B end                   ; End program

add_numbers:
    PUSH {LR}               ; Save the return address
    ADD R0, R0, R1          ; Add the numbers
    POP {PC}                ; Return to caller

end:
    B end                   ; Infinite loop
```

**Advantages of Using Stacks and Subroutines:**

- **Stacks:** Provide a flexible way to save and restore the state of registers.
- **Subroutines:** Facilitate code reuse and organization, reducing program size and complexity.


8-bit data
8-bit data
8-bit result
source register 1
source register 2
operation
destination register
128-bit wide registers

# Thank you