Stacks and Subroutines

Stacks are highly flexible in the ARM architecture. In the ARM processor, any one of the general purpose registers could be used as a stack pointer.

Stack instructions : The ARM instruction set does not contain any stack specific instructions like push and pop. The instruction set also does not enforce in anyway the use of a stack. Push and pop operations are performed by memory access instructions, with auto-increment addressing modes.

Stack pointer : The stack pointer is a register that points to the top of the stack. In the ARM processor, there are no dedicated stack pointer registers, and any one of the general purpose registers can be used as the stack pointer.

Stack types : Since it is left to the software to implement a stack, different implementation choices result different types of stacks. There are two types of stack depending on how the stack grows.

Ascending stack : In a push the stack pointer is incremented, i.e the stack grows towards higher address.

Descending stack : In a push the stack pointer is decremented, i.e the stack grows towards lower address.

There are two types of stack depending on what the stack pointer points to.

1. **Empty stack** : Stack pointer points to the location in which the next item will be stored. A push will store the value, and increment the stack pointer.
2. **Full stack** : Stack pointer points to the location in which the last item was stored. A push will increment the stack pointer and store the value.

Four different stacks are possible : Full-ascending, full-descending, empty-ascending, empty-descending. All four can be implemented using the register load store instructions.

A subroutine is a reusable program module. A main program can call or jump to the subroutine one or more times. The stack is used in several ways when subroutines are called.