ARM calling convention

- Register usage:

Registers	Function	Value preserved during call
R0-R3	Arguments / Return values	No
R4-R11	Local variables	Yes
R12 (IP)	Intra-procedure-call scratch reg.	No
R13 (SP)	Stack Pointer	Yes
R14 (LR)	Link register	No
R15 (PC)	Program Counter	No

- If a routine has more than 4 arguments R0-R3 are used for the first 4 arguments and the rest are placed on the stack before the call
- The stack must be of the Full-Descending type
- Local variables can also be stored in R0-R3, R12, and even LR, specially in “leaf” subroutines (no other subroutine call)

ARM calling convention

- Typical subroutine prologue:

```
routine: stmfd sp!,{r4-r6,lr}
```

  Saves R4, R5, R6 and LR on the stack. R4-R6 will be used for local variables and LR for calling other subroutines.

- Typical subroutine epilogue:

```
         ldmfd sp!,{r4-r6,pc}
```

  Restores R4, R5 and R6 from the stack. PC is restored instead of LR, therefore also making this instruction a subroutine return

ARM calling convention

- Calling a thumb subroutine from ARM (interworking):

```
                .arm
                …
                ldr     r12,=(tmbrout+1)
                mov     lr,pc                   @return addr.
                bx      r12                     @jump to thumb
                …

                .thumb
                …
      tmbrout:  …
                …
                bx      lr     @return (interworking)
```

Notes:

- PC is 8 bytes (2 instructions) ahead of “mov lr,pc”
- Bit 0 of R12 is 1, meaning a jump to thumb code in “bx r12”
- Bit 0 of LR is 0, meaning a jump to ARM code in “bx lr”