5 THE BASE PROCEDURE CALL STANDARD

The base standard defines a machine-level, core-registers-only calling standard common to the ARM and Thumb instruction sets. It should be used for systems where there is no floating-point hardware, or where a high degree of inter-working with Thumb code is required.

5.1 Machine Registers

The ARM architecture defines a core instruction set plus a number of additional instructions implemented by co-processors. The core instruction set can access the core registers and co-processors can provide additional registers which are available for specific operations.

5.1.1 Core registers

There are 16, 32-bit core (integer) registers visible to the ARM and Thumb instruction sets. These are labeled r0-r15 or R0-R15. Register names may appear in assembly language in either upper case or lower case. In this specification upper case is used when the register has a fixed role in the procedure call standard. *Table 2, Core registers and AAPCS usage* summarizes the uses of the core registers in this standard. In addition to the core registers there is one status register (CPSR) that is available for use in conforming code.

Register	Synonym	Special	Role in the procedure call standard
r15		PC	The Program Counter.
r14		LR	The Link Register.
r13		SP	The Stack Pointer.
r12		IP	The Intra-Procedure-call scratch register.
r11	v8		Variable-register 8.
r10	v7		Variable-register 7.
r9		v6 SB TR	Platform register. The meaning of this register is defined by the platform standard.
r8	v5		Variable-register 5.
r7	v4		Variable register 4.
r6	v3		Variable register 3.
r5	v2		Variable register 2.
r4	v1		Variable register 1.
r3	a4		Argument / scratch register 4.
r2	a3		Argument / scratch register 3.
r1	a2		Argument / result / scratch register 2.
r0	a1		Argument / result / scratch register 1.

Table 2, Core registers and AAPCS usage

The first four registers r0-r3 (a1-a4) are used to pass argument values into a subroutine and to return a result value from a function. They may also be used to hold intermediate values within a routine (but, in general, only *between* subroutine calls).

Register r12 (IP) may be used by a linker as a scratch register between a routine and any subroutine it calls (for details, see *§5.3.1.1, Use of IP by the linker*). It can also be used within a routine to hold intermediate values *between* subroutine calls.

The role of register r9 is platform specific. A virtual platform may assign any role to this register and must document this usage. For example, it may designate it as the static base (SB) in a position-independent data model, or it may designate it as the thread register (TR) in an environment with thread-local storage. The usage of this register may require that the value held is persistent across all calls. A virtual platform that has no need for such a special register may designate r9 as an additional callee-saved variable register, v6.

Typically, the registers r4-r8, r10 and r11 (v1-v5, v7 and v8) are used to hold the values of a routine's local variables. Of these, only v1-v4 can be used uniformly by the whole Thumb instruction set, but the AAPCS does not require that Thumb code only use those registers.

A subroutine must preserve the contents of the registers r4-r8, r10, r11 and SP (and r9 in PCS variants that designate r9 as v6).

In all variants of the procedure call standard, registers r12-r15 have special roles. In these roles they are labeled IP, SP, LR and PC.

The CPSR is a global register with the following properties:

- The N, Z, C, V and Q bits (bits 27-31) and the GE[3:0] bits (bits 16-19) are undefined on entry to or return from a public interface. The Q and GE[3:0] bits may only be modified when executing on a processor where these features are present.
- On ARM Architecture 6, the E bit (bit 8) can be used in applications executing in little-endian mode, or in big-endian-8 mode to temporarily change the endianness of data accesses to memory. An application must have a designated endianness and at entry to and return from any public interface the setting of the E bit must match the designated endianness of the application.
- The T bit (bit 5) and the J bit (bit 24) are the execution state bits. Only instructions designated for modifying these bits may change them.
- The A, I, F and M[4:0] bits (bits 0-7) are the privileged bits and may only be modified by applications designed to operate explicitly in a privileged mode.
- All other bits are reserved and must not be modified. It is not defined whether the bits read as zero or one, or whether they are preserved across a public interface.

5.1.1.1 Handling values larger than 32 bits

Fundamental types larger than 32 bits may be passed as parameters to, or returned as the result of, function calls. When these types are in core registers the following rules apply:

- A double-word sized type is passed in two consecutive registers (e.g., r0 and r1, or r2 and r3). The content of the registers is as if the value had been loaded from memory representation with a single `LDM` instruction.
- A 128-bit containerized vector is passed in four consecutive registers. The content of the registers is as if the value had been loaded from memory with a single `LDM` instruction.

5.1.2 Co-processor Registers

A machine's register set may be extended with additional registers that are accessed via instructions in the co-processor instruction space. To the extent that such registers are not used for passing arguments to and from subroutine calls the use of co-processor registers is compatible with the base standard. Each co-processor may provide an additional set of rules that govern the usage of its registers.