# ARM Architectural Reference

## 1. Data Types

ARM Architecture Version 4 processors support the following data types:

| | |
|---|---|
| Byte | 8 bits |
| Halfword | 16 bits; halfwords must be aligned to two-byte boundaries |
| Word | 32 bits; words must be aligned to four-byte boundaries |

ARM instructions are exactly one word ( and therefore aligned on a four-byte boundary). THUMB instructions are exactly one halfword ( and therefore aligned on a two-byte boundary).

All data operations (e.g. ADD) are performed on word quantlities.
Load and store operations can transfer bytes, halfwords, ad words to and from memory, automatically zero-extending or sign-extending bytes of halfwords as they are loaded.

Signed operands are in two's complement format.

## 2. Processor Modes

ARM Version 4 supports seven processor modes:

| Processor mode | | | Description |
|---|---|---|---|
| 1 | User | (usr) | Normal program execution mode |
| 2 | FIQ | (fiq) | Supports a high-speed data transfer or channel process |
| 3 | IRQ | (irq) | Used for general purpose interrupt handling |
| 4 | Supervisor | (svc) | A protected mode for the operating system |
| 5 | Abort | (abt) | Implements virtual memory and/or memory protection |
| 6 | Undefined | (und) | Supports software emulation of hardware coprocessors |
| 7 | System | (sys) | Runs privileged operating system tasks (Architecture Version 4 only) |

Mode changes may be made under software control or may be caused by external interrutps or exception processing. Most application programs will execute in User mode. The other modes, known as privileged modes, will be entered to service interrupts or exceptions or to access protected resources.

## 3. Registers

The processor has a total of 37 registers:

- 30 general-purpose registers
- 6 status registers
- 1 program counter

The registers are arranged in partially overlapping banks: a different register bank for each processor mode. At any one time, 15 general-purpose registers (R0 to R14),one or two status registers and program counter are visible. The general-purpose registers and status registers currently visible depend on the current processor mode. The register bank organisation is shown in the table. The banked registers are in red-bold font.

The general-purpose registers are 32 bits wide.

Register 13 (the Stack Pointer or SP) is banked across all modes to provide a private stack pointer for each mode (except system mode which share the user mode R13).

Register 14 (the Link Register or LR) is used as the subroutine return address link register. R14 is also banked across all modes (expect system mode which shares the user mode R14). When a Subroutine call (Branch and Link instruction) is executed, R14 is set to the subroutine return address; R14_svc, R14_irq, R14_fiq, R14_abort and R14_undef are used similarly to hold the return address when exceptions occur (or a subroutine return address if subroutine call are executed within interrupt or exception routines). R14 may be treated as a general-purpose register at all other time.

FIQ mode also has banked registers R8 to R12 (as well as R13 and R14). R8_fiq,R9_fiq, R10_fiq,R11_fiq, R12_fiq are provided to allow very fast interrupt processing (without the need to preserve register contents by storing them to memory), and to preserve values across interrupt calls (so that register contents do not need be restored from memory).

Register R15 holds the Program Counter (PC). When R15 is read, bits[1:0] are zero and bits[31:2] contain the PC. When R15 is written, n bits[1:0] are ignored and bits[31:2] are written to the PC. Depending on how it is used, the value of the PC is either the address of the instruction plus 8 or is UNPREDICATABLE.

The Current Program Status Register (CPSR) is also accessible in all processor modes. It contains condition code flags, interrupt enable flags and the current mode. Each privileged mode (except system mode) has a Saved Program Status Register (SPSR) which is used to preserve the value of the CPSR when an exception occurs.

| ARM MODE | | | | | |
|---|---|---|---|---|---|
| user/system | supervisor | abort | undefined | interrupt | FIQ |
| **R0** | R0 | R0 | R0 | R0 | R0 |
| **R1** | R1 | R1 | R1 | R1 | R1 |
| **R2** | R2 | R2 | R2 | R2 | R2 |
| **R3** | R3 | R3 | R3 | R3 | R3 |
| **R4** | R4 | R4 | R4 | R4 | R4 |
| **R5** | R5 | R5 | R5 | R5 | R5 |
| **R6** | R6 | R6 | R6 | R6 | R6 |
| **R7** | R7 | R7 | R7 | R7 | R7 |
| **R8** | R8 | R8 | R8 | R8 | **R8_FIQ** |
| **R9** | R9 | R9 | R9 | R9 | **R9_FIQ** |
| **R10** | R10 | R10 | R10 | R10 | **R10_FIQ** |
| **R11** | R11 | R11 | R11 | R11 | **R11_FIQ** |
| **R12** | R12 | R12 | R12 | R12 | **R12_FIQ** |
| **R13** | **R13_SVC** | **R13_ABORT** | **R13_UNDEF** | **R13_IRQ** | **R13_FIQ** |
| **R14** | **R14_SVC** | **R14_ABORT** | **R14_UNDEF** | **R14_IRQ** | **R14_FIQ** |
| **PC** | PC | PC | PC | PC | PC |
| **CPSR** | CPSR | CPSR | CPSR | CPSR | CPSR |
| | **SPSR_SVC** | **SPSR_ABORT** | **SPSR_UNDEF** | **SPSR_IRQ** | **SPSR_FIQ** |

**4. Program Status Registers**

The format of the Current Program Status Register (CPSR) and the Saved Program Status Registers (SPSR) are shown in the following table. The N, Z, C and V (Negative, Zero, Carry and

oVerflow) bits collectively known as the condition code flags. The condition code flags in the CPSR can be changed as a result of arithmetic and logical operations in the processor, and can be tested by all instructions to determine if the instruction is to be executed.

| 31 | 30 | 29 | 28 | 27 .. 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | Z | C | V | DNM/RAZ | I | F | T | M4 | M3 | M2 | M1 | M0 |

**4.1 The control bits**

The bottom 8 bits of a PSR (incorporating I, F, T and M[4:0]) are known collectively as the **control bits**. The control bits change when an exception arises and can be altered by software only when the processor is in a privileged mode. The I and F bits are the interrupt disable bits:

I bit disables IRQ interrupt when it is set
F bit disables FIQ interrupt when it is set

The T flag is only implemented on Architecture Version 4T (THUMB):

0 indicates ARM execution
1 indicates THUMB execution

On all other version of the architecture the T flag should be zero (SBZ).

**4.2 The mode bits**

The M[4:0] are the mode bits, and these determine the mode in which the processor operates. The interpretation of the mode bits is shown in the following table. Not all combinations of the mode bits define a valid processor mode. Only those explicitly described can be used; if any other value is programmed into the mode bits[4:0], the result is unpredictable.

| M[4:0] | Mode | Accessible Registers |
|---|---|---|
| 0b10000 | User | PC, R14 to R0, CPSR |
| 0b10001 | FIQ | PC, R14_fiq to R8_fiq, R7 to R0, CPSR, SPSR_fiq |
| 0b10010 | IRQ | PC, R14_irq, R13_irq, R12 to R0, CPSR, SPSR_irq |
| 0b10011 | SVC | PC, R14_svc, R13_svc, R12 to R0, CPSR, SPSR_svc |
| 0b10111 | Abort | PC, R14_abt, R13_abt, R12 to R0, CPSR, SPSR_abt |
| 0b11011 | Undef | PC, R14_und, R13_und, R12 to R0, CPSR, SPSR_und |
| 0b11111 | system | PC, R14 to R0, CPSR (Architecture Version 4 only) |

User mode and system mode do not have a SPSR, as these modes are not entered on any exception, so a register to preserve the CPSR is not required. In User mode or System mode any reads to the SPSR will read an unpredictable value, and any writes to the SPSR will be ignored.

**5. Exceptions**

Exceptions are generated by internal and external source to cause the processor to handle an event; for example, an externally generated interrupt, or an attempt to execute an undefined instruction. The processor state just before handling the exception must be preserved so that the original program can be resumed when the exception routine has completed. More than one exception may arise at the same time.

ARM supports 7 types of exception and has a privileged processor mode for each type of exception. The following table lists the types of exception and the processor mode that is used to

process that exception. When an exception and execution is forced from a fixed memory address corresponding to the type of exception. These fixed address are called the **Hard Vectors**.

The reserved entry at address 0x14 is for an Address Exception vector used when then processor is configured for a 26-bit address space.

| Exception type | Mode | Vector address |
|---|---|---|
| Reset | SVC | 0x0000 0000 |
| Undefined instructions | UNDEF | 0x0000 0004 |
| Software Interrupt (SWI) | SVC | 0x0000 0008 |
| Prefetch Abort (Instruction fetch memory abort) | ABORT | 0x0000 000c |
| Data Abort (Data Access memory abort) | ABORT | 0x0000 0010 |
| IRQ (Interrupt) | IRQ | 0x0000 0018 |
| FIQ (Fast Interrupt) | FIQ | 0x0000 001c |

When taking an exception, the banked registers are used to save state. When an exception occurs, these action are performed:

R14_<exception_mode> = PC
SPSR_<exception_mode> = CPSR
CPSR[5:0] = Exception mode number
CPSR[6] = if <exception_mode> == Reset of FIQ then = 1 else unchanged
CPSR[7] = 1; Interrupt disabled
PC = Exception vector address

To return after handling the exception, the SPSR is moved into the CPSR and R14 is moved to the PC. This can be done atomically in two ways:

1. Using a data-processing instruction with the S bit set, and the PC as the destination.
2. Using the Load Multiple and Restore PSR instruction.

The following sections show the recommended way of returning from each exception.

```
Reset:    No return
Undefined instruction exception:    MOVS   PC, R14
Software interrupt exception:       MOVS   PC, R14
Prefetch Abort:                     SUBS   PC, R14, #4
Data Abort:
        re-execute the aborted instruction:          SUBS   PC, R14, #8
        don't re-execute the aborted instruction:    SUBS   PC, R14, #4
IRQ:                                SUBS   PC, R14, #4
FIQ:                                SUBS   PC, R14, #4
```

## 6. Exception priorities

The Reset exception has the highest priority. FIQ has higher priority than IRQ. IRQ has higher priority than prefetch abort.

Undefined instruction and software interrupt cannot occur at the same time, as they each correspond to particular (non-overlapping) decodings of the current instruction, and both must be lower priority than prefetch abort, as a prefetch abort indicates that not valid instruction was fetched.

The priority of data abort is higher than FIQ and lower priority than Reset, which ensures that the data-abort handler is entered before the FIQ handler is entered (so that the data abort will be resolved after the FIQ handler has completed).

| Exception | Priority |
|---|---|
| Reset | 1 (Highest) |
| Data Abort | 2 |
| FIQ | 3 |
| IRQ | 4 |
| Prefetch Abort | 5 |
| Undefined Instruction, SWI | 6 (Lowest) |

# This doc is for ARM v3 and ARM v4. Gavin Li July 9, 2001