Chapter 2

ARM Processor Fundamentals

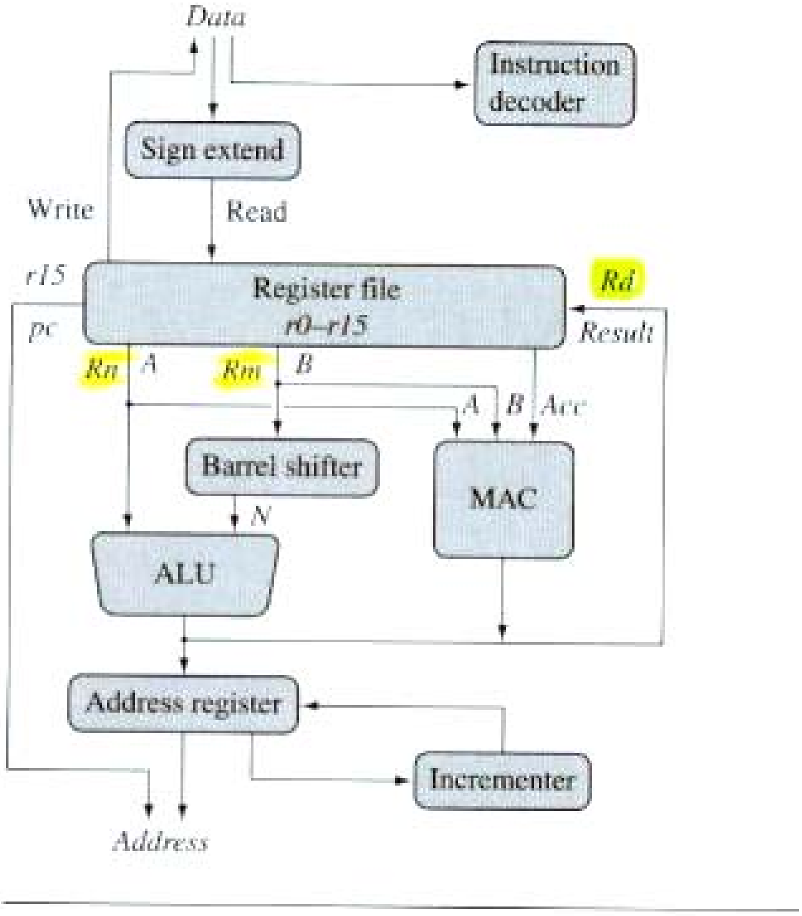

Figure 2.1 ARM core dataflow model.

(Note: MAC = multiply-accumulate unit)

- A 32-bit processor implemented either by
  Von Neumann architecture (shown above)
  or
  Harvard architecture with 2 types of instruction sets—
  load and store instructions

- Instructions have 2 source registers (***Rn*** and ***Rm***) and one destination register (***Rd***)

- ***Rm*** can be preprocessed in the barrel shifter before it enters the ALU

2.1 Register

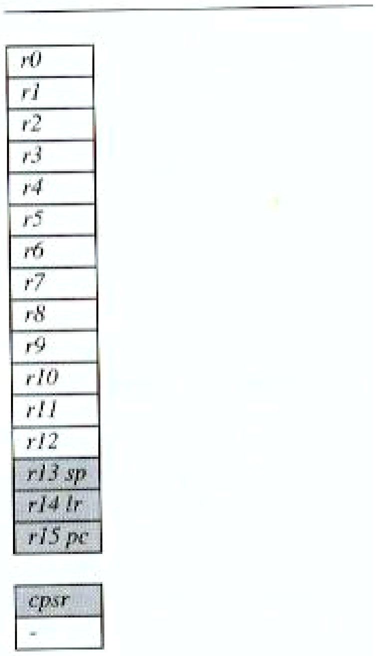

Figure 2.2 Registers available in *user* mode.

- Up to 18 active register:

  16 data registers (r0~r15) : for holding either data or address
  2 processor status registers

- r13 : stack pointer (sp)
- r14 : link register (lr)
- r15 : program counter (pc)

- r0~r13 are orthogonal (equally well application)

- cpsr : current program status register
- spsr : saved program status register

2.2 Current Program Status Register

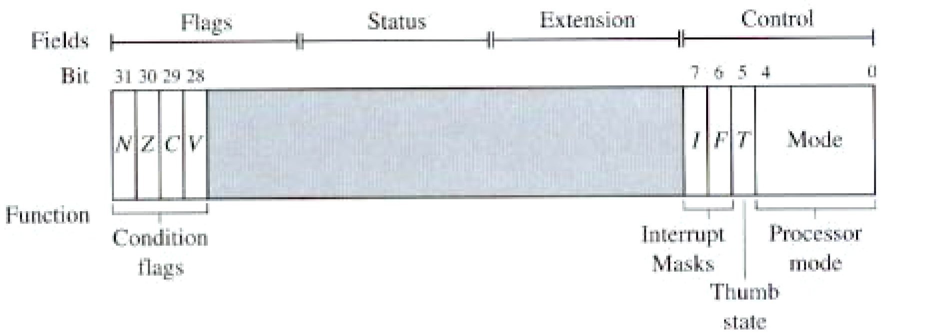

Figure 2.3 A generic program status register (*psr*).

Divided into 4 fields (each with 8-bit wide):

- Flags : holding instruction conditions
- Status : reserved
- Extension : reserved
- Control : indicate the processor mode

2.2.1 Processor Modes

- The processor mode determines which registers are active and the access rights to the ***cpsr*** register itself.
- 7 processor modes:

Processor mode	***Abbrev.***	***Description***	***Notes***
User	usr	Noraml program exection mode	
System	sys	Run privileged OS tasks	***P***
Supervisor	svc	A protected mode for the OS	***p, e***
Abort	abt	Implements virtual memory and/or memory protection	***p, e***
Undefined	und	Supports software emulation of hardware coprocessors	***p, e***
Interrupt	irq	Used for general-purpose interrupt handling	***p, e***
Fast interrupt	fiq	Supports a high-speed data transfer or channel process	***p, e***

[note] Privileged modes : ***p*** Execption modes : ***e***

- Each process mode is either privileged or nonprivileged

  A privileged mode allows full read-write access to the ***cpsr***

  A nonprivileged mode only allows to read access to the control filed in the ***cpsr*** but still allows read-write access to the condition flags.

2.2.2 Banked Registers

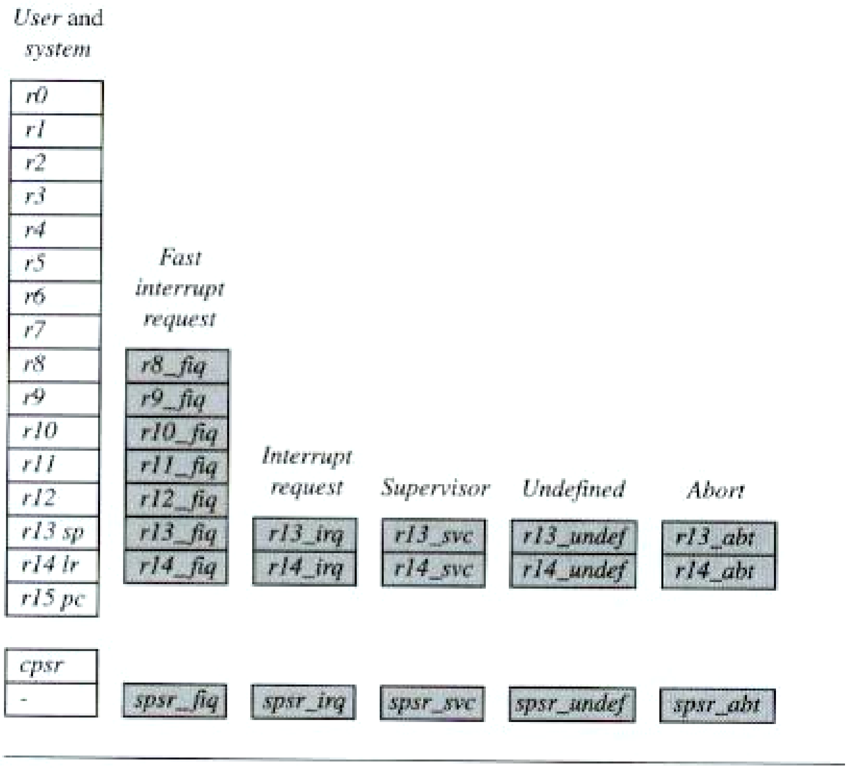

Figure 2.4 Complete ARM register set.

- 37 registers in the register file.
- 20 registers, called **banked registers** (identifed by the shading in the diagram), are hidden from a program at different times.
- All processor modes except system mode have a set of associate banked registers that are a subset of the main 16 registers.
- A banked register maps one-to-one onto a user mode register
- The processor mode can be changed by a program that writes directly to the ***cpsr***, if it has the privilege**.**
- The following exceptions and interrupts cause a mode change:
  + Reset
  + Interrupt request
  + Fast interrupt request
  + Software interrupt
  + Data abort,
  + Prefetch abort
  + Undefined instruction.

2.2.2 Banked Registers (cont.)

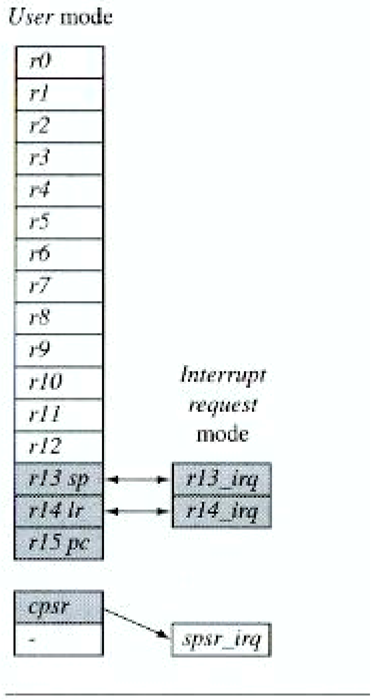

Figure 2.5 Changing mode on an exception.

- The ***cprs*** is not coped into ***spsr*** when a mode change is forced due to a program writing directly to the ***cpsr.***

- The saving of the ***cspr*** only occurs when an exception or interrupt is raised.

Table 2.1 Processor mode.

Mode	Abbreviation	Privileged	Mode[4:0]
Abort	abt	yes	10111
Fast interrupt request	fiq	yes	10001
Interrupt request	irq	yes	10010
Supervisor	svc	yes	10011
System	sys	yes	11111
Undefined	und	yes	11011
User	usr	no	10000

2.2.3 State and Instruction Sets

- There are 3 instruction sets :
  ARM – 32-bit instructions
  Thumb – 16-bit instructions
  Jeazelle – 8-bit instructions

- No intermingle sequence of different instruction set is allowed.

Table 2.2 ARM and Thumb instruction set features.

	ARM (*cpsr* $T = 0$)	Thumb (*cpsr* $T = 1$)
Instruction size	32-bit	16-bit
Core instructions	58	30
Conditional execution[a]	most	only branch instructions
Data processing instructions	access to barrel shifter and ALU	separate barrel shifter and ALU instructions
Program status register	read-write in privileged mode	no direct access
Register usage	15 general-purpose registers +*pc*	8 general-purpose registers +7 high registers +*pc*

[a] See Section 2.2.6.

- The hardware portion of Jazelle only supports a subset of the Java byecodes; the rest are emulated in software.

2.2.4 Interrupt masks

- Two interruput marks—I and F—are used to stop specific interrupt requests from interrupting the processor.
- The mask bits, 7 and 6 (or I and F), are used to control the masking of IRQ and FIQ, respectively.

2.2.5 Condition Flags

Table 2.4 Condition flags.

Flag	Flag name	Set when
Q	Saturation	the result causes an overflow and/or saturation
V	oVerflow	the result causes a signed overflow
C	Carry	the result causes an unsigned carry
Z	Zero	the result is zero, frequently used to indicate equality
N	Negative	bit 31 of the result is a binary 1

- The hardware only sets the flags.
- To clear the flag you need to write to the ***cpsr***.

-

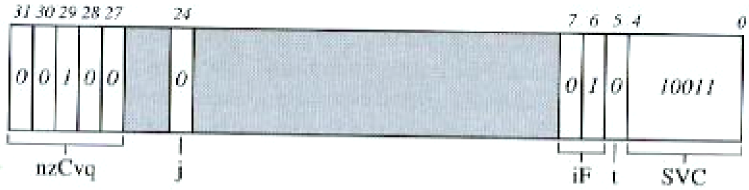

Figure 2.6 Example: *cpsr = nzCvqjiFt_SVC.*

- Bit 5 is used to set for thumb-enable.
- Bit 24 is uese to set for Jazelle-enable.

2.2.6 Condition Execution

Table 2.5 Condition mnemonics.

Mnemonic	Name	Condition flags
EQ	equal	*Z*
NE	not equal	*z*
CS HS	carry set/unsigned higher or same	*C*
CC LO	carry clear/unsigned lower	*c*
MI	minus/negative	*N*
PL	plus/positive or zero	*n*
VS	overflow	*V*
VC	no overflow	*v*
HI	unsigned higher	*zC*
LS	unsigned lower or same	*Z* or *c*
GE	signed greater than or equal	*NV* or *nv*
LT	signed less than	*Nv* or *nV*
GT	signed greater than	*NzV* or *nzv*
LE	signed less than or equal	*Z* or *Nv* or *nV*
AL	always (unconditional)	ignored

- Most instructions have a condition attribute that determines if the core will execute it based on the setting of the condition flag.

- The condition attribute is postfixed in the instruction mnemonic, which is coded into the instruction.

2.3 Pipeline

- to speed up instruction execution

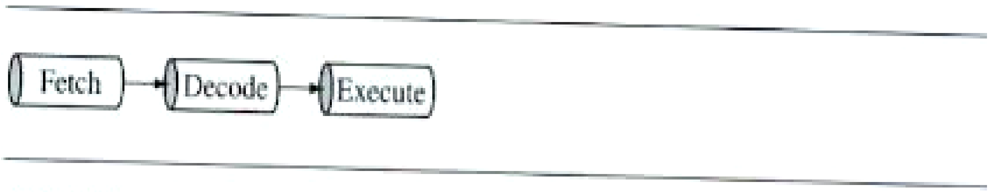

Figure 2.7 ARM7 Three-stage pipeline.

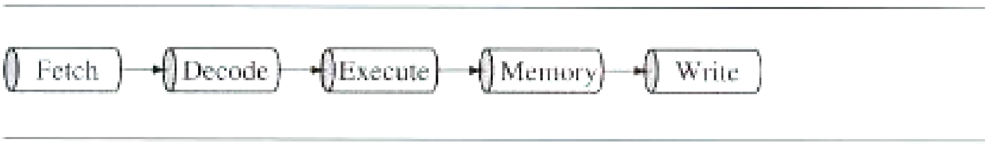

Figure 2.9 ARM9 five-stage pipeline.

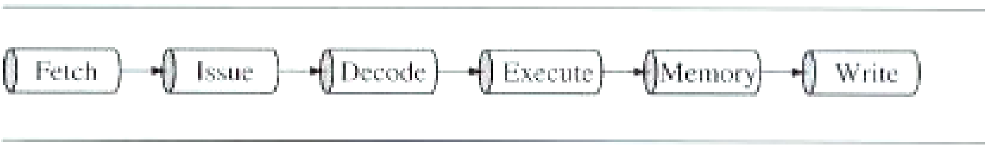

Figure 2.10 ARM10 six-stage pipeline.

- ARM9 : 1.1 Dhrystone MIPS per Mhz—an increasing in instruction throughput by around 13% compared with an ARM7.

- ARM10 : 1.3 Dhrystone MIPS per Mhz—an increasing in instruction throughput by around 34% compared with an ARM7.

-

- 2.3.1 Pipeline Executing Characteristics

- In ARM state, the ***pc*** always points to the address of the instruction plus 8 bytes.
- In Thumb state, the ***pc*** is the instruction address plus 4 bytes

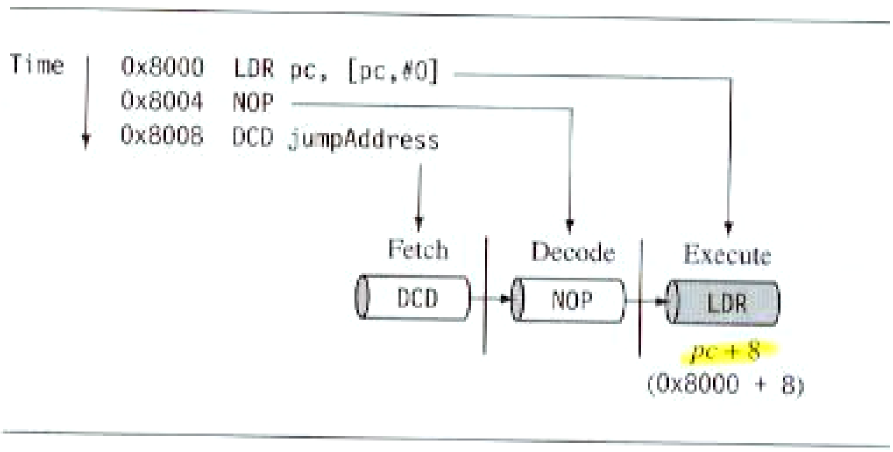

Figure 2.12 Example: pc = address + 8.

- The execution of a branch instruction or branching by the direct modification of the **pc** causes the ARM core to flush its pipeline

.

- ARM10 uses branch prediction which reduces the effect of a pipeline flush.

- An instruction in the execute stage will complete even though an interrupt has been raised. Other instructions in the pipeline will be abounded.

2.4 Exceptions, Interrupts, and the Vector Table

- When an exception or interrupt occurs, the processor sets the ***pc*** to a specific memory address.
- The interrupt address is within a special address called vector table
- The entries in the vector table are instructions that branch to specific routines designed to handles a particular exception or interrupt.
- The memory map address 0x0000000 is reserved for the vector table, a set of 32-bit words.
- On some processes the vector table can be optionally located at a higher address in memory—starting at the offset 0xffff0000.

Table 2.6 The vector table.

Exception/interrupt	Shorthand	Address	High address
Reset	RESET	0x00000000	0xffff0000
Undefined instruction	UNDEF	0x00000004	0xffff0004
Software interrupt	SWI	0x00000008	0xffff0008
Prefetch abort	PABT	0x0000000c	0xffff000c
Data abort	DABT	0x00000010	0xffff0010
Reserved	—	0x00000014	0xffff0014
Interrupt request	IRQ	0x00000018	0xffff0018
Fast interrupt request	FIQ	0x0000001c	0xffff001c

2.5 Core Extensions

- To improve performance, manage resources, and provide extra functionality.
- - Three hardware extension: cache and tightly couple menory, memory management, and coprocessor interface.

2.5.1 Cache and Tightly Couple Memory

- Two types of cache:

  + cache for the combination of data and instruction which is attached to the von Neumann-style core.

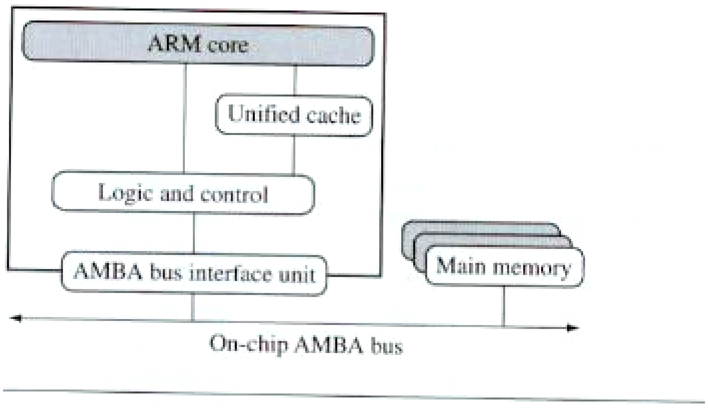

Figure 2.13 A simplified Von Neumann architecture with cache.

  + cache for separated data and instruction which is attached to the Harvard-style core.

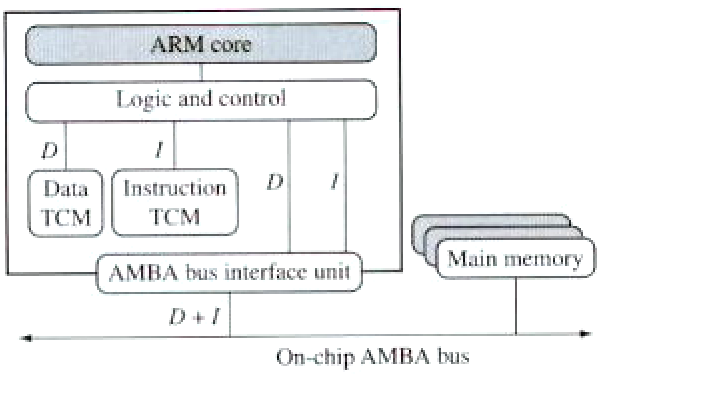

Figure 2.14 A simplified Harvard architecture with TCMs.

2.5.1 Cache and Tightly Couple Memory (cont.)

- The tightly coupled memory (TCM), a type of fast SRAM, is located close to the core for the real-time system.
- TCMs appear as memory in the address map and can be accessed as fast memory.
- A combination of caches and TCMs is shown below.

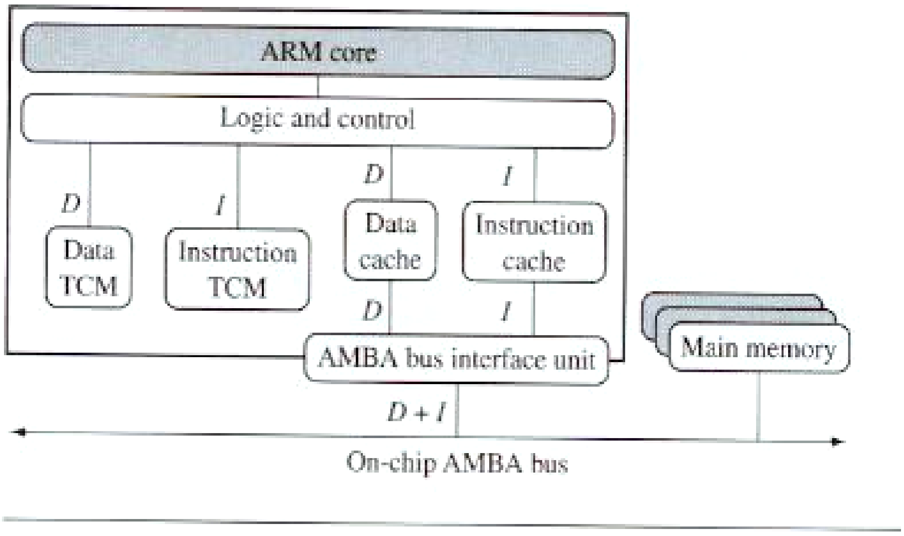

Figure 2.15 A simplified Harvard architecture with caches and TCMs.

2.5.2 Memory Management

- Generally, embedded system use multiple memory devices—ROM, SRAM, FlahROM, DRAM

- It's necessary to have a method to help organize these devices and protect the system from applications trying to make inappropriate accesses to hardware.

- ARM cores have 3 different types of memory management hardware:

  + No extensions – nonprotected memory for small, simple embedded systems.

  + Memory protection unit (MPU) – using a limited number of memory region which are controlled with a set of special coprocessor registers, and each region is defined with specific access permission.

  + Memory management unit (MMU) – using a set of translation table to provide fine-grained control over memory, which are stored in main memory and provide a virtual-to-physical address memory map as well as access permission.

2.6 Architecture Revisions

Every ARM processor implementation executes a specific instruction set architecture—ISA.

2.6.1 Nomenclature

ARM{x}{y}{z}{T}{D}{M}{I}{E}{J}{F}{-S}

x—family
y—memory management/protection unit
z—cache
T—Thumb 16-bit decoder
D—JTAG debug
M—fast multiplier
I—EmbeddedICE macrocell
E—enhanced instructions (assumes TDMI)
J—Jazelle
F—vector floating-point unit
S—synthesizible version

Figure 2.16 ARM nomenclature.

- JTAG : IEEE1149.1 Standard Test Access Port

- EmbeddedICE macrocell : the debug hardware built into the processor that allows breakpoints and watchpoints to be set. (ICE : in-circuit emulator)

- Synthesizible : meaning that the processor core is supplied as source code that can be compiled into a form easily used by EDA (electronic design automation) tools.

2.6.2 Architecture Evolution

Table 2.7 Revision history.

Revision	Example core implementation	ISA enhancement
ARMv1	ARM1	First ARM processor 26-bit addressing
ARMv2	ARM2	32-bit multiplier 32-bit coprocessor support
ARMv2a	ARM3	On-chip cache Atomic swap instruction Coprocessor 15 for cache management
ARMv3	ARM6 and ARM7DI	32-bit addressing Separate *cpsr* and *spsr* New modes—*undefined instruction* and *abort* MMU support—virtual memory
ARMv3M	ARM7M	Signed and unsigned long multiply instructions
ARMv4	StrongARM	Load-store instructions for signed and unsigned halfwords/bytes New mode—*system* Reserve SWI space for architecturally defined operations 26-bit addressing mode no longer supported
ARMv4T	ARM7TDMI and ARM9T	Thumb
ARMv5TE	ARM9E and ARM10E	Superset of the ARMv4T Extra instructions added for changing state between ARM and Thumb Enhanced multiply instructions Extra DSP-type instructions Faster multiply accumulate
ARMv5TEJ	ARM7EJ and ARM926EJ	Java acceleration
ARMv6	ARM11	Improved multiprocessor instructions Unaligned and mixed endian data handling New multimedia instructions

Table 2.8 Description of the *cpsr*.

Parts	Bits	Architectures	Description
Mode	4:0	all	processor mode
T	5	ARMv4T	Thumb state
I & *F*	7:6	all	interrupt masks
J	24	ARMv5TEJ	Jazelle state
Q	27	ARMv5TE	condition flag
V	28	all	condition flag
C	29	all	condition flag
Z	30	all	condition flag
N	31	all	condition flag

2.7 ARM Processor Families

Table 2.9 ARM family attribute comparison.

	ARM7	ARM9	ARM10	ARM11
Pipeline depth	three-stage	five-stage	six-stage	eight-stage
Typical MHz	80	150	260	335
mW/MHz[a]	0.06 mW/MHz	0.19 mW/MHz (+ cache)	0.5 mW/MHz (+ cache)	0.4 mW/MHz (+ cache)
MIPS[b]/MHz	0.97	1.1	1.3	1.2
Architecture	Von Neumann	Harvard	Harvard	Harvard
Multiplier	8 × 32	8 × 32	16 × 32	16 × 32

[a] Watts/MHz on the same 0.13 micron process.
[b] MIPS are Dhrystone VAX MIPS.

Table 2.10 ARM processor variants.

CPU core	MMU/MPU	Cache	Jazelle	Thumb	ISA	E[a]
ARM7TDMI	none	none	no	yes	v4T	no
ARM7EJ-S	none	none	yes	yes	v5TEJ	yes
ARM720T	MMU	unified—8K cache	no	yes	v4T	no
ARM920T	MMU	separate—16K /16K $D + I$ cache	no	yes	v4T	no
ARM922T	MMU	separate—8K/8K $D + I$ cache	no	yes	v4T	no
ARM926EJ-S	MMU	separate—cache and TCMs configurable	yes	yes	v5TEJ	yes
ARM940T	MPU	separate—4K/4K $D + I$ cache	no	yes	v4T	no
ARM946E-S	MPU	separate—cache and TCMs configurable	no	yes	v5TE	yes
ARM966E-S	none	separate—TCMs configurable	no	yes	v5TE	yes
ARM1020E	MMU	separate—32K/32K $D + I$ cache	no	yes	v5TE	yes
ARM1022E	MMU	separate—16K/16K $D + I$ cache	no	yes	v5TE	yes
ARM1026EJ-S	MMU and MPU	separate—cache and TCMs configurable	yes	yes	v5TE	yes
ARM1136J-S	MMU	separate—cache and TCMs configurable	yes	yes	v6	yes
ARM1136JF-S	MMU	separate—cache and TCMs configurable	yes	yes	v6	yes

[a] E extension provides enhanced multiply instructions and saturation.

-end-