
The ARM (and Cortex) processor

ARM - Acorn Risc Machine

ARM is a RISC processor with the following features

- Load-store architecture
- 16 registers, each 32-bits wide
- Three-operand instructions, 32-bits wide
- At each clock cycle a new instruction is issued (well, almost...)
- Low energy consumption (W/MIPS)

It is designed by ARM Ltd. and licensed to manufacturers

- ARM7: nommu, very widespread as a microcontroller
- DEC, then Intel: StrongARM (now dead)
- Intel, then Marvell: Xscale: 80200, PXA255, PXA270, IXP425, ...
- Cirrus Logic: EP93xx: 9302, 9315, ...
- Freescale (now NXP): iMX1, iMX21, iMX27, iMX31, ..., iMX7, iMX8, iMX9, ...
- Atmel: AT91SAM: many flavours
- ... many more ...

http://en.wikipedia.org/wiki/ARM_architecture

ARM Machine Code (32bit: not the Cortex-M code)

Major features of the ARM machine code

- **Unaligned access is not allowed (like all RISC)**
- **No assignment of 32-bit constants is possible (like all RISC)**
- **There is no hardware-managed stack pointer (like all RISC)**
- **Load-multiple and store-multiple instructions**
- **Every instruction is conditionally executed**
- **The status bits are modified only optionally**
- **One operand can be shifted at no cost**
- **All addressing is register-relative**

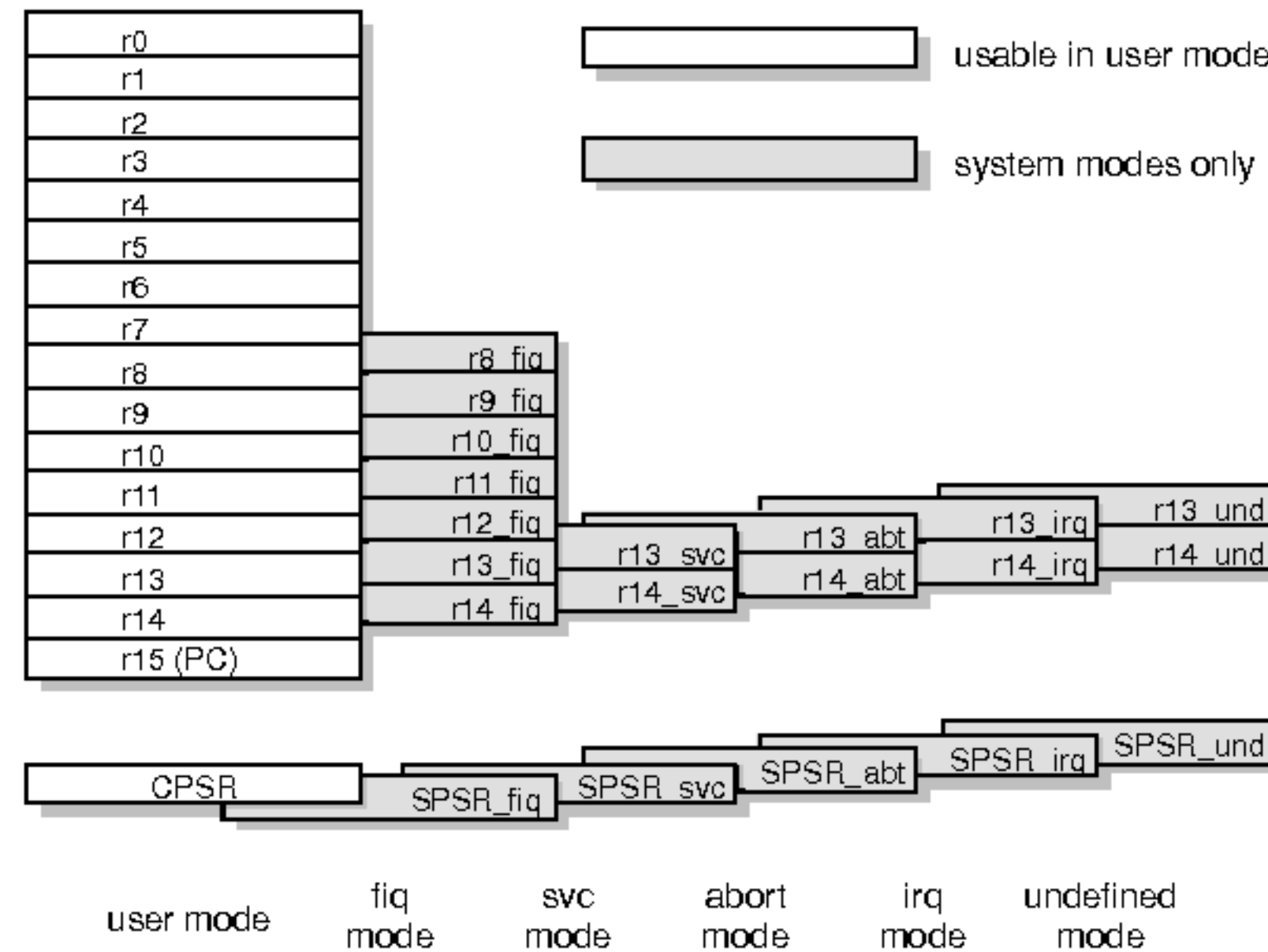
Other features typical of RISC processors, but missing in ARM:

- **Register windows**
- **«Delay slot» for jumps**
- **Zero register**

ARM Registers

The 16 registers are orthogonal, and some "shadow" instance exists

- This does not apply to cortex-M, but it's historically important



©1996 Addison Wesley Longman

The point of shadow registers is that the CPU would never access RAM

- ... unless the instruction is a load or store one.
- This is the same idea behind the "link register" concept.

The Standard ABI, Coprocessors

r0	a1
r1	a2
r2	a3
r3	a4
r4	v1
r5	v2
r6	v3
r7	v4
r8	v5
r9	v6
r10	v7
r11	v8
r12	ip
r13	sp
r14	lr
r15	pc

Role of registers

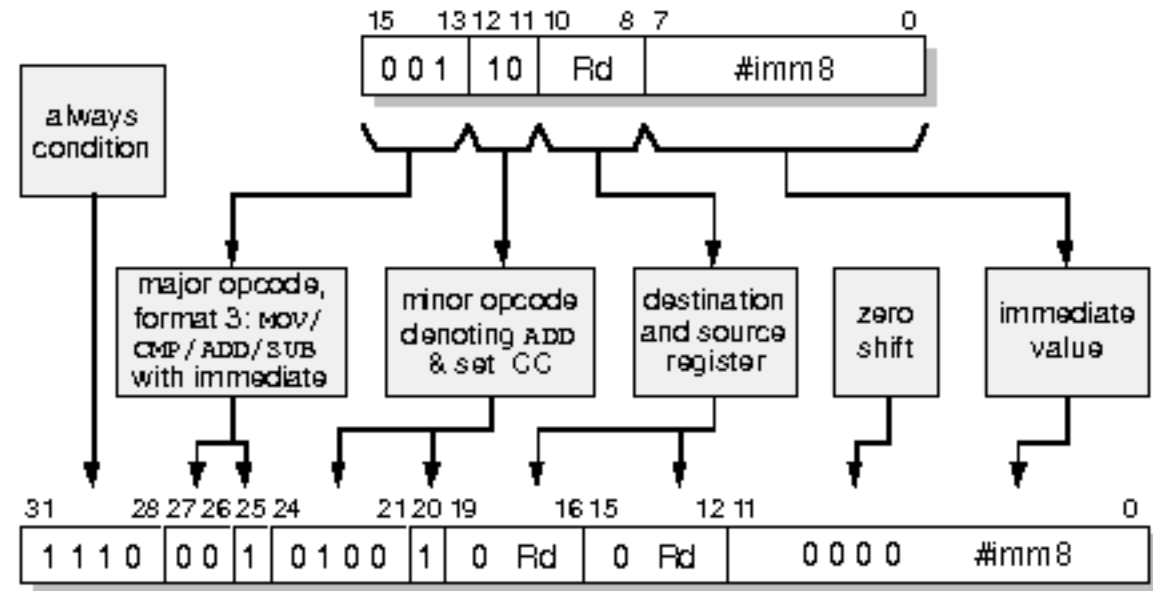
- **A registers are function arguments (caller-saved)**
- **V registers are callee-saved**
- **R12 is the "intra-procedure scratch"**
- **R13 is the stack pointer (software defined)**
- **R14 is the link register (hardware defined)**
- **R15 is the program counter (hardware defined)**

Coprocessors

- **The architecture defines 16 coprocessors**
 - ♦ **CP15, if present, is used for cache and MMU**
 - ♦ **CP0 and CP1, if present, are used for FPU**
- **The following instructions are defined by the architecture:**
 - ♦ **Register move CPU/coprocessor: MRC, MCR**
 - ♦ **Coprocessor load and store: LDC, STC**
 - ♦ **Coprocessor data processing: CDP**

The Thumb Extension (Thumb-1)

Thumb instructions are 16-bits wide



©1996 Addison Wesley Longman

- ◆ **Code is more compact**
- ◆ **2-operand operations**
- ◆ **Only a subset of the registers can be accessed**
- ◆ **Some "special" instructions are not supported.**

BX and BLX (branch (and link) and exchange) switch mode

- **Bit 0 of the program counter is used as a selector**
 - ◆ **ARM instructions run at even addresses**
 - ◆ **Thumb instructions (appear to) run at odd addresses**

The Thumb-2 Instruction Set

The "Cortex" (ARMv6/v7) family introduced Thumb-2

- It extended the thumb instruction set to be complete
- Thumb now accesses all registers (not only 8 of them)
- It includes coprocessor instructions
- New ITE instruction (if then else)

It is a separate instruction decoder, not a decompressor any more

The design also changes how interrupt vectors work

- There used to be 8 _instructions_ at address 00..1f
- Vectors are now pointers rather than instructions (and many of them)
- Moreover, "VIC" is offered as part of the core

So it is now possible to make thumb-only devices

- But the core accesses RAM by itself, which I personally dislike

The Cortex-M family is a microcontroller flavour, thumb-2 only.

- We have Cortex-M0, Cortex-M0+, Cortex-M3, Cortex-M4
- They are all similar and all different. We'll learn to differentiate