The ARM 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
- Low energy consumption (W/MIPS)

It is designed by ARM Ltd. and licensed to manufacturers

- ARM7: nommu, very widespread as a microcontroller
- StrongARM: DEC, then Intel (now dead)
- Xscale: Intel, then Marvell (80200, PXA255, PXA270, IXP425, ...)
- EP93xx: Cirrus Logic (es: 9302, 9315)
- iMX: Freescale (iMX1, iMX21, iMX27, iMX31, iMX51)
- AT91: Atmel
- ... many more ...

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

ARM Machine Code

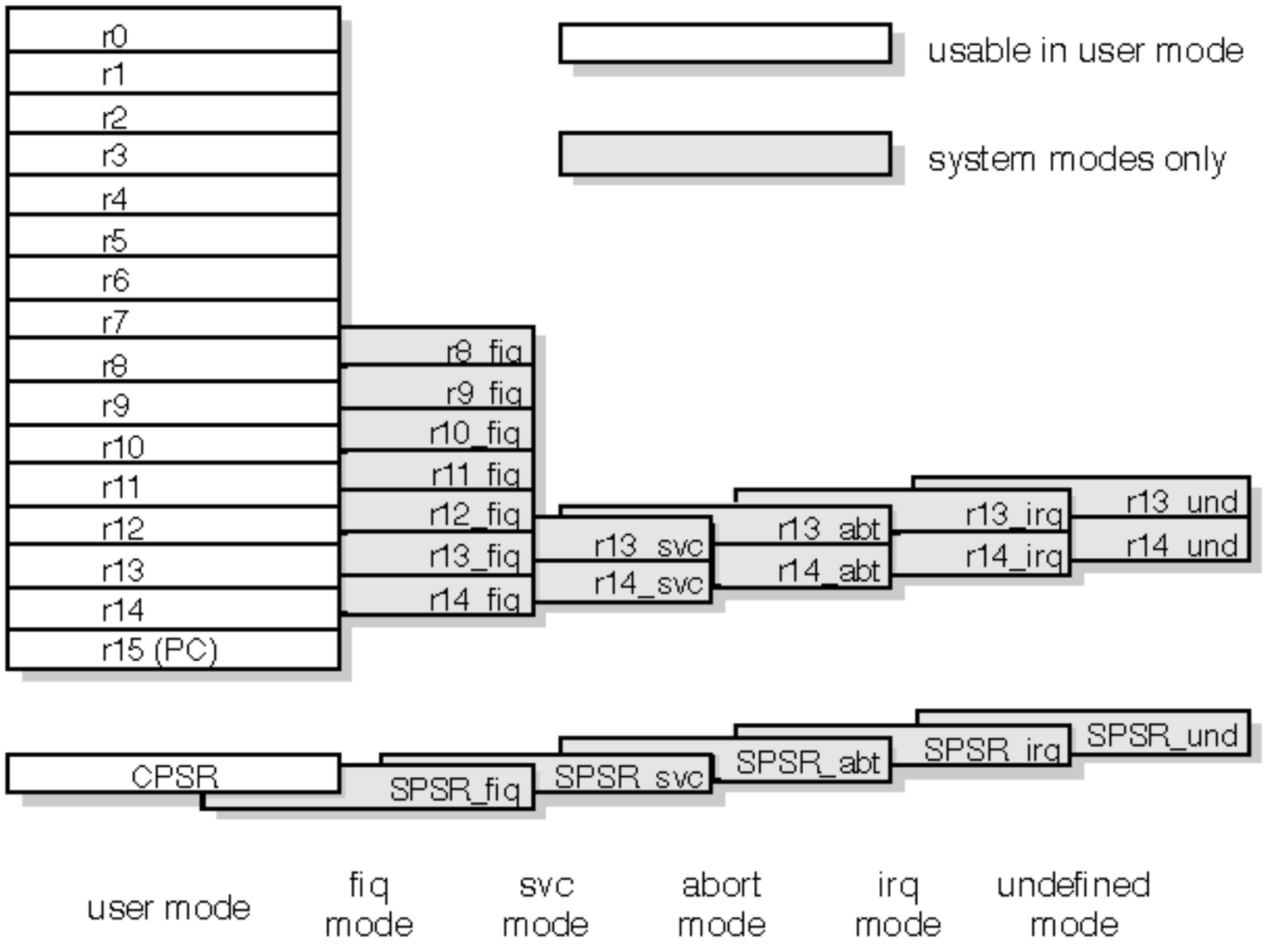
Major features of the ARM machine code

- 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 (32-bit)



How can we work without a hardware-defined stack?

- **Using the link register (R14) for function calls**
- **Using banked registers for interrupt management**

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
- R14 is the link register
- R15 is the program counter

Coprocessors

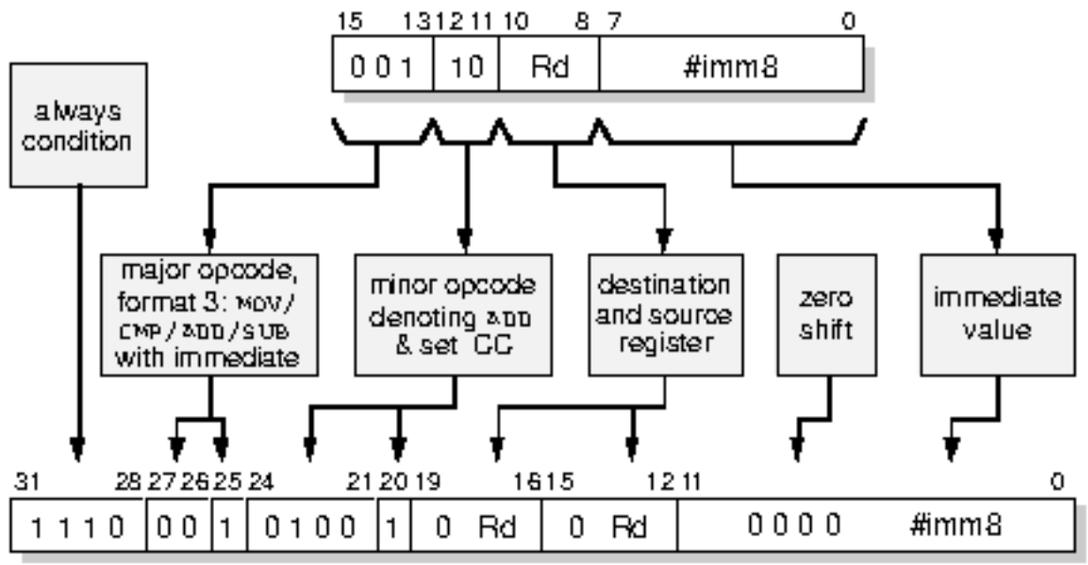
- The architecture defines 16 coprocessors
  - CP15, if present, is used for cache and MMU
  - CP0 and CP1, if present, are used for FPU
- The following instructions are defined by the architecture:
  - Register move CPU/coprocessor: MRC, MCR
  - Coprocessor load and store: LDC, STC
  - Coprocessor data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

The Thumb Extension (ARM7 = ARMv4)

Thumb instructions are 16-bits wide

- The idea is to "expand" instructions on the fly
- Pro: It took very little logic in the core
- Pro: Code is much more compact
- Con: Only a subset of the registers can be accessed
- Con: 2-operand operations (ARM has 3-operand ops)
- Con: No conditional ops nor some other interesting features

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



ARM/Thumb interoperability

Since ARM7T, we have two possible operating modes

- The "T" bit is part of the processor status register
- To set (or clear) it we jump to an odd (even) address.
- Bit 0 of the program counter is thus used as a selector

Not all instructions are available in Thumb mode

- In particular, no special instruction to access banked registers
- Also, interrupt (and (trap) management starts in ARM mode
  - This is needed for compatibility with existing code

The vast majority of UC developers chose Thumb

- There is a little performance penalty, but that's ok
- What is limited (and costly), in microcontrollers, is memory

As a matter of facts, this is what happened in the user base

- The vast majority of code was built in Thumb mode
- Only some OS procedures (and IRQ entry/exit) used ARM mode

The Thumb-2 Instruction Set

The "Cortex" (ARMv7) family introduced Thumb-2

- It is a separate instruction decoder, not a decompressor any more
- It is an extension of the previous Thumb machine code
- It can access all registers
- All core features can be accessed
- New ITE instruction (if then else),

ARM defined a Unified Assembler Language

- You can build the same source as ARM or Thumb instructions
  - The idea saves a lot of conditionals and unmaintained code
  - Unfortunately, it is not trivial

It is now possible to build Thumb-only devices

- Cortex-A (application processor) has both ARM and Thumb2
- Cortex-M (microcontroller) only includes a Thumb2 decoder

This also required a change in the IRQ vectors

- Not a problem when making incompatible changes anyways