ARM

Introduction

- The ARM processor is a Reduced Instruction Set Computer (RISC).
- Originated in processor research programmes at Stanford and Berkeley universities around 1980.
- The ARM was originally developed at Acorn Computers Limited of Cambridge, England, between 1983 and 1985.
- It was the first RISC microprocessor developed for commercial use.
- It has become established as a market-leader for low-power and cost-sensitive embedded applications.
- The ARM is supported by a toolkit which includes an instruction set emulator for hardware modelling and software testing and benchmarking, an assembler, C and C++ compilers, a linker and a symbolic debugger.

- Acorn RISC Machine
- Advanced RISC Machines Limited
- the only examples of RISC architectures were the Berkeley RISC I and II and the Stanford MIPS (which stands for Microprocessor without Interlocking Pipeline Stages)
- The ARM architecture incorporated a number of features from the Berkeley RISC design
    - a load-store architecture
    - fixed-length 32-bit instructions
    - 3-address instruction formats

The features that were employed on the Berkeley RISC designs which were rejected by the ARM designers were:

- Register windows.
- Delayed branches.
- Single-cycle execution of all instructions.

The ARM programmer's model

Current Program Status Register

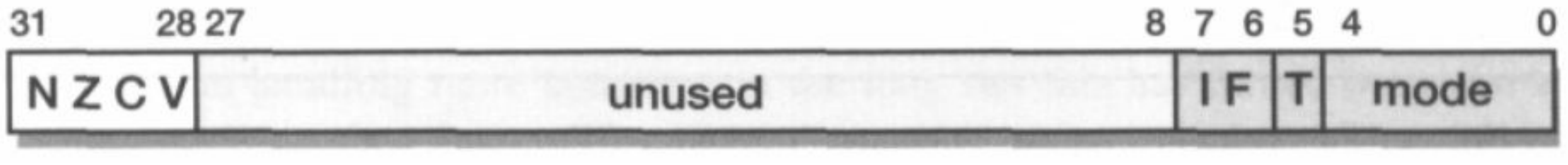

Figure 2.2 ARM CPSR format.

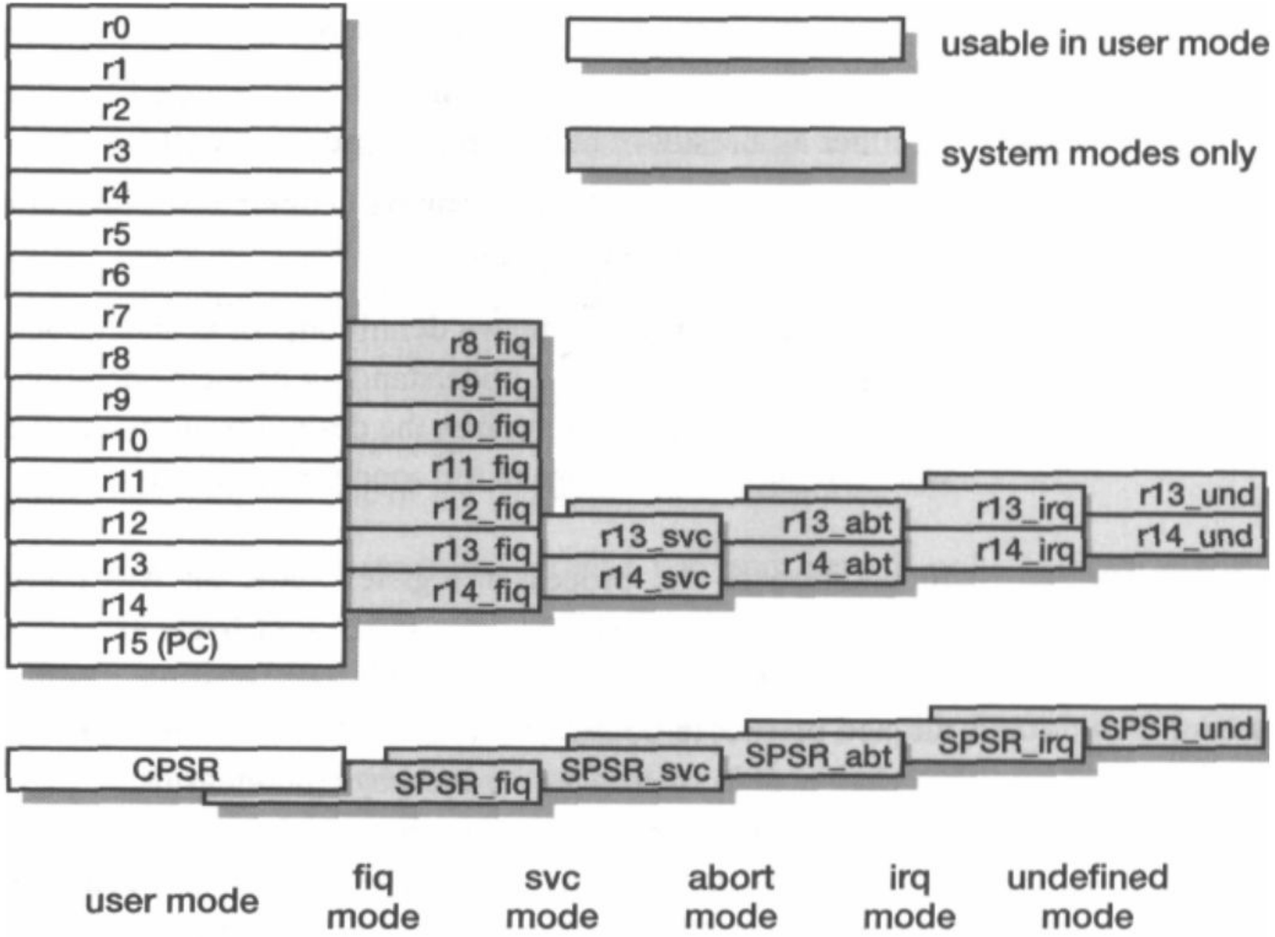

Figure 2.1 ARM's visible registers.

User level programs - 15 general purpose 32 bit registers (r0 to r14), CPSR

System level programs and exception handling - remaining registers are used

CPSR - used in user level programs, to store condition code bits

Least significant bits - control the processor modes, instruction set, interrupt enable, protected from change by the user level program

Most significant bits - condition code flags

N - Negative, Z- Zero, C - Carry, V - Overflow

Processor register state

Memory state - linear array of bytes from 0 to $2^{32} - 1$

Data items may be 8 bit bytes, 16 bit half words, 32 bit words

Words are always aligned on 4 byte boundaries

Half-words are aligned on even byte boundaries

Each byte location has a unique number

A byte may occupy any of these locations

A word-sized data item must occupy a group of four byte locations starting at a byte address which is a multiple of four

Half-words occupy two byte locations starting at an even byte address

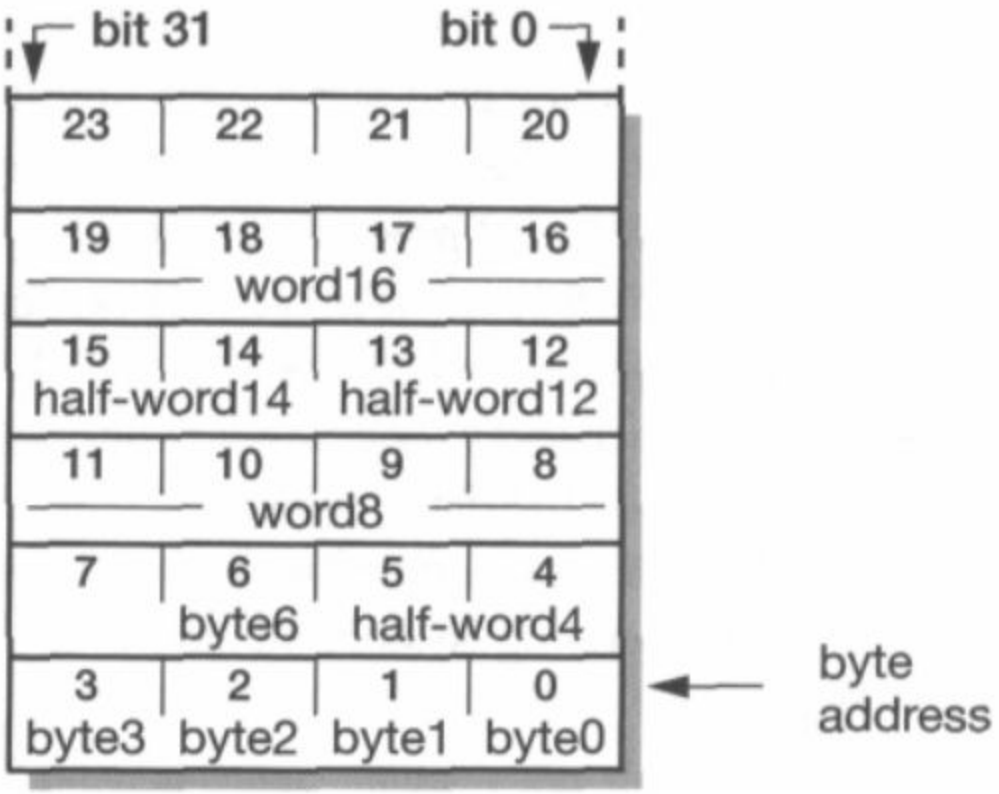

Figure 2.3 ARM memory organization.

Load-Store Architecture

- In common with most RISC processors, ARM employs a load-store architecture.
- the instruction set will only process (add, subtract, and so on) values which are in registers (or specified directly within the instruction itself), and will always place the results of such processing into a register
- Operations copy memory values into registers (load instructions)
- Operations copy register values into memory (store instructions)
- CISC processors typically allow a value from memory to be added to a value in a register, and sometimes allow a value in a register to be added to a value in memory.
- ARM does not support such 'memory-to-memory' operations.

ARM instructions fall into one of the following three categories:

- Data processing instructions. - use and change only values in registers
- Data transfer instructions.- copy memory values to register (load) and register values to memory (store)
- Control flow instructions. - causes execution to switch to different address, either permanently (branch) or saving a return address to resume original sequence (branch and link) or trapping into system code (supervisor calls).

Supervisor mode

The ARM processor supports a protected supervisor mode.

The protection mechanism ensures that user code cannot gain supervisor privileges without appropriate checks.

System-level functions can only be accessed through specified supervisor calls.

These functions generally include any accesses to hardware peripheral registers.

The ARM instruction set

- All ARM instructions are 32 bits wide.
- ARM instructions are aligned on 4-byte boundaries in memory.
- ARM instruction set has more formats.
- Complex instruction decoding
- High code density
- Features:
  - The load-store architecture
  - 3-address data processing instructions
  - the two source operand registers and the result register are all independently
  - conditional execution of every instruction
  - the inclusion of very powerful load and store multiple register instructions
  - the ability to perform a general shift operation and a general ALU operation in a single instruction that executes in a single clock cycle
  - open instruction set extension through the coprocessor instruction set, including adding new registers and data types to the programmer's model
  - a very dense 16-bit compressed representation of the instruction set in the Thumb architecture

I/O System

- The ARM handles I/O (input/output) peripherals (such as disk controllers, network interfaces, and so on) as memory-mapped devices with interrupt support.
- The internal registers in these devices appear as addressable locations within the ARM's memory map and may be read and written using the same (load-store) instructions as any other memory locations.
- Peripherals may attract the processor's attention by making an interrupt request using either the normal interrupt (IRQ) or the fast interrupt (FIQ) input.
- Both interrupt inputs are level-sensitive and maskable.
- Normally most interrupt sources share the IRQ input, with just one or two time-critical sources connected to the higher-priority FIQ input.
- Some systems may include direct memory access (DMA) hardware external to the processor to handle high-bandwidth I/O traffic.

ARM exceptions

The ARM architecture supports a range of interrupts, traps and supervisor calls, all grouped under the general heading of exceptions.

1. The current state is saved by copying the PC into rl4_exc and the CPSR into SPSR_exc (where exc stands for the exception type).
2. The processor operating mode is changed to the appropriate exception mode.
3. The PC is forced to a value between 0016 and 1C16, the particular value depending on the type of exception.

The instruction at the location the PC is forced to (the vector address) will usually contain a branch to the exception handler.

The exception handler will use rl3_exc, which will normally have been initialized to point to a dedicated stack in memory, to save some user registers for use as work registers.

The return to the user program is achieved by restoring the user registers and then using an instruction to restore the PC and the CPSR atomically.

The main features of the ARM architecture are:

- a large set of registers, all of which can be used for most purposes;
- a load-store architecture;
- 3-address instructions (that is, the two source operand registers and the result register are all independently specified);
- conditional execution of every instruction;
- the inclusion of very powerful load and store multiple register instructions;
- the ability to perform a general shift operation and a general ALU operation in a single instruction that executes in a single clock cycle;
- open instruction set extension through the coprocessor instruction set, including adding new registers and data types to the programmer's model.
- a very dense 16-bit compressed representation of the instruction set in the Thumb architecture.