4 ARM Organization and Implementation

Summary of chapter contents

The organization of the ARM integer processor core changed very little from the first 3 micron devices developed at Acorn Computers between 1983 and 1985 to the ARM6 and ARM7 developed by ARM Limited between 1990 and 1995. The 3-stage pipeline used by these processors was steadily tightened up, and CMOS process technology reduced in feature size by almost an order of magnitude over this period, so the performance of the cores improved dramatically, but the basic principles of operation remained largely the same.

Since 1995 several new ARM cores have been introduced which deliver significantly higher performance through the use of 5-stage pipelines and separate instruction and data memories (usually in the form of separate caches which are connected to a shared instruction and data main memory system).

This chapter includes descriptions of the internal structures of these two basic styles of processor core and covers the general principles of operation of the 3-stage and 5-stage pipelines and a number of implementation details. Details on particular cores are presented in Chapter 9.

4.1 3-stage pipeline ARM organization

The organization of an ARM with a 3-stage pipeline is illustrated in Figure 4.1 on page 76. The principal components are:

- The register bank, which stores the processor state. It has two read ports and one write port which can each be used to access any register, plus an additional read port and an additional write port that give special access to r15, the program counter. (The additional write port on r15 allows it to be updated as the instruction fetch address is incremented and the read port allows instruction fetch to resume after a data address has been issued.)
- The barrel shifter, which can shift or rotate one operand by any number of bits.
- The ALU, which performs the arithmetic and logic functions required by the instruction set.
- The address register and incrementer, which select and hold all memory addresses and generate sequential addresses when required.
- The data registers, which hold data passing to and from memory.
- The instruction decoder and associated control logic.

In a single-cycle data processing instruction, two register operands are accessed, the value on the B bus is shifted and combined with the value on the A bus in the ALU, then the result is written back into the register bank. The program counter value is in the address register, from where it is fed into the incrementer, then the incremented value is copied back into r15 in the register bank and also into the address register to be used as the address for the next instruction fetch.

The 3-stage pipeline

ARM processors up to the ARM7 employ a simple 3-stage pipeline with the following pipeline stages:

- Fetch;
  the instruction is fetched from memory and placed in the instruction pipeline.
- Decode;
  the instruction is decoded and the datapath control signals prepared for the next cycle. In this stage the instruction 'owns' the decode logic but not the datapath.
- Execute;
  the instruction 'owns' the datapath; the register bank is read, an operand shifted, the ALU result generated and written back into a destination register.

At any one time, three different instructions may occupy each of these stages, so the hardware in each stage has to be capable of independent operation.

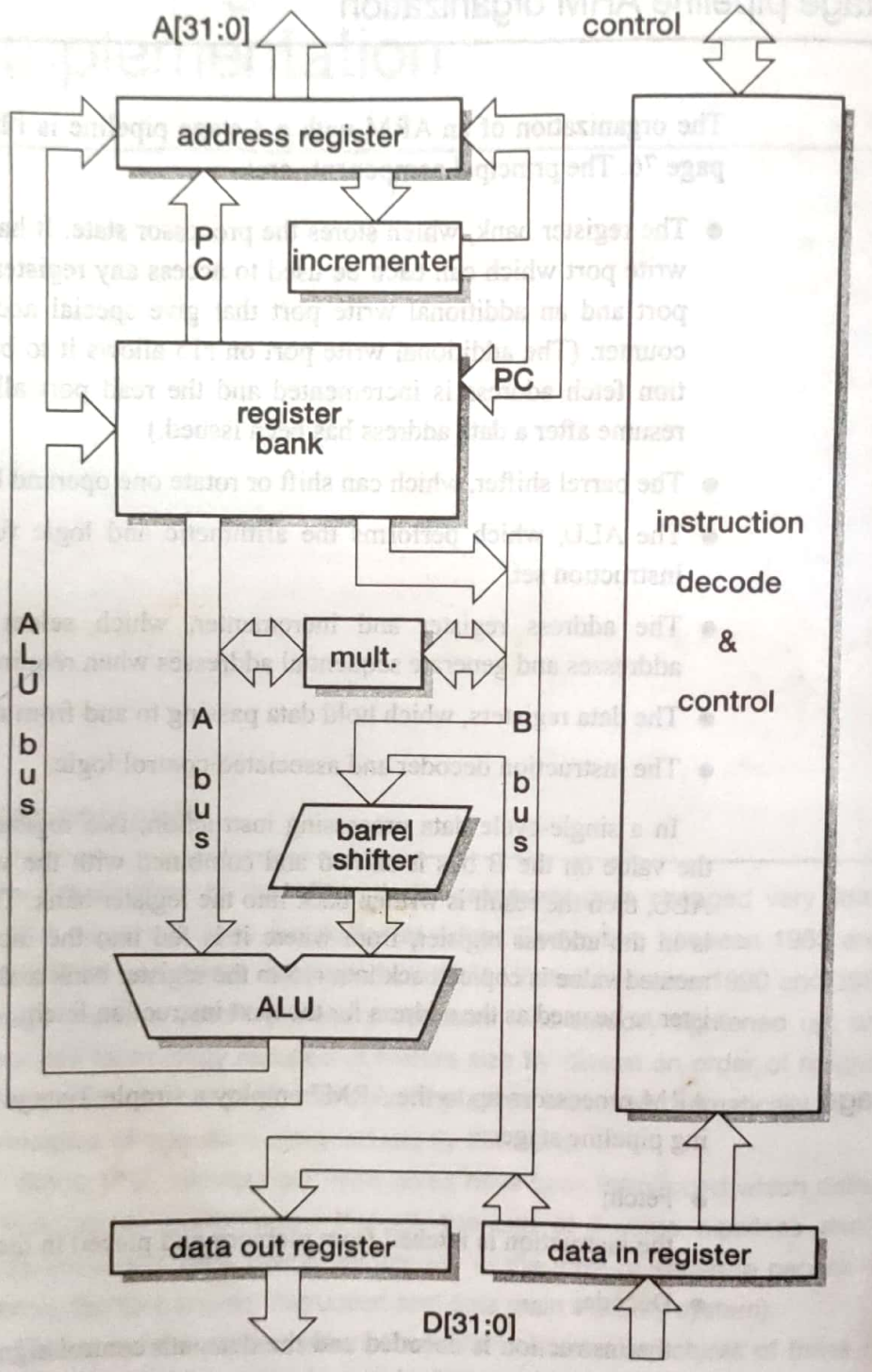

Figure 4.1 3-stage pipeline ARM organization.

When the processor is executing simple data processing instructions the pipeline enables one instruction to be completed every clock cycle. An individual instruction takes three clock cycles to complete, so it has a three-cycle **latency**, but the **throughput** is one instruction per cycle. The 3-stage pipeline operation for single-cycle instructions is shown in Figure 4.2 on page 77.

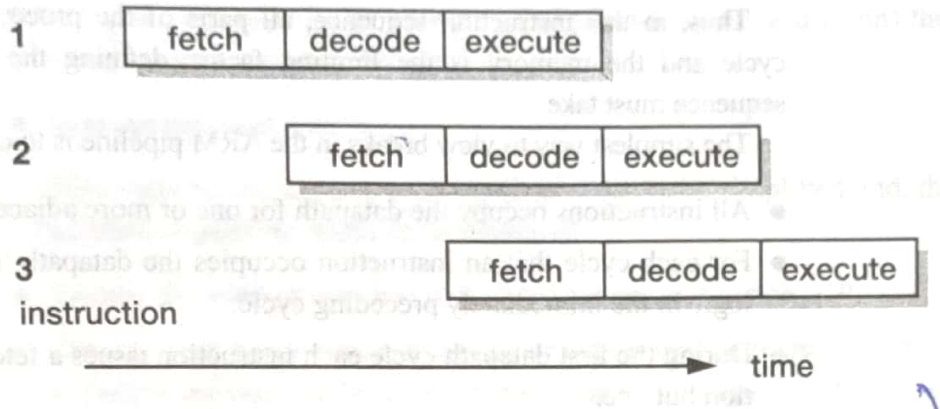

Figure 4.2 ARM single-cycle instruction 3-stage pipeline operation.

When a multi-cycle instruction is executed the flow is less regular, as illustrated in Figure 4.3. This shows a sequence of single-cycle ADD instructions with a data store instruction, STR, occurring after the first ADD. The cycles that access main memory are shown with light shading so it can be seen that memory is used in every cycle. The datapath is likewise used in every cycle, being involved in all the execute cycles, the address calculation and the data transfer. The decode logic is always generating the control signals for the datapath to use in the next cycle, so in addition to the explicit decode cycles it is also generating the control for the data transfer during the address calculation cycle of the STR.

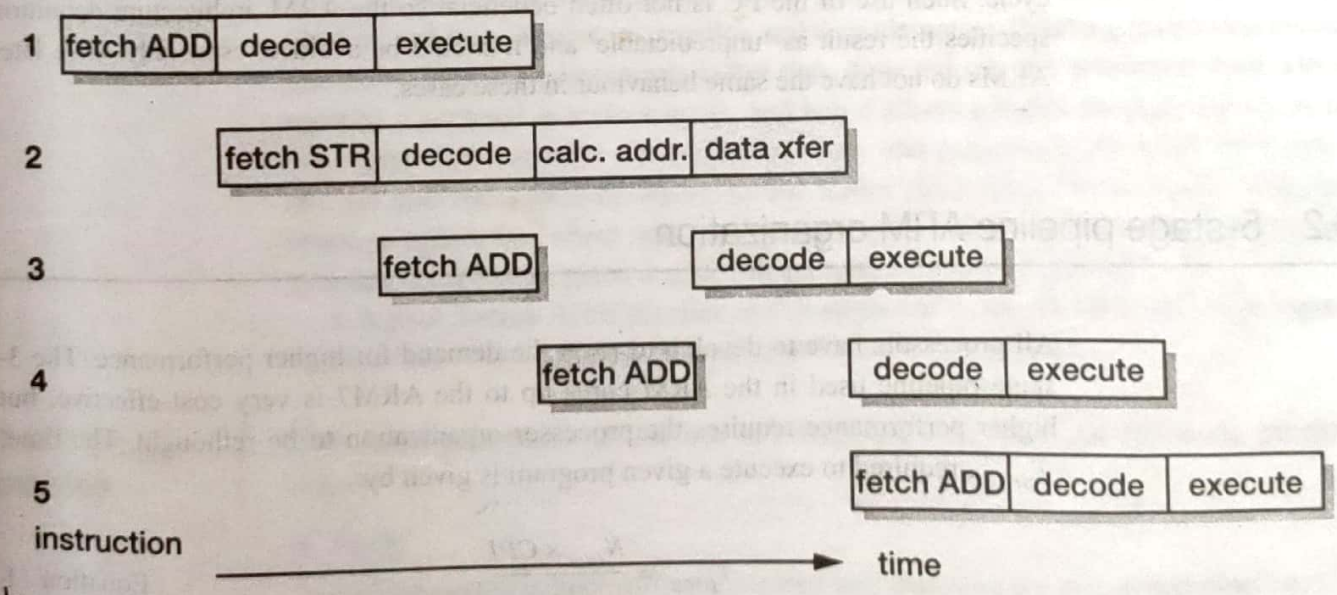

Figure 4.3 ARM multi-cycle instruction 3-stage pipeline operation.