Building An Instruction Set

CHAPTER OVERVIEW

In this first chapter that looks 'under the bonnet' we discuss how to build instructions from micro-instructions.

This chapter includes:

▶ Micro-instructions revisited - data movement operations, ALU operations, test operations and processor control operations;
▶ How to construct instructions from micro-instructions;
▶ Building new instruction sets.

16.1 The Instruction Set

We have now used our instruction set in anger, and it is time to revisit how instructions actually work, looking at the types of micro-instructions, and then seeing how to create new instructions.

16.2 Micro-Instructions Revisited

We have seen before, in chapter 8 and earlier, how micro-instructions are used to build assembly language instructions.

To remind ourselves, each micro-instruction belongs to one of the following groups:
Data transfer micro-instructions;  
ALU micro-instructions;  
Test micro-instructions;  
Processor control micro-instructions.

Each micro-instruction is actually defined at the hardware level of our processor, and when a micro-instruction is executed it performs certain actions at that level, and we will look into this in chapter 17. To describe the action of a micro-instruction, we have described it in RTL, such as with the data movement

\[ A \leftarrow [MDR] \]

We will look at each group in turn, to make sure we are acquainted with what they do in our processor.

### 16.2.1 Data Transfer Micro-Instructions

Data transfer microcodes copy a bit pattern from one location to another.

For example, \( A \leftarrow [MDR] \) describes the micro-instruction that copies a bit pattern from the MDR into the A register.

### 16.2.2 ALU Micro-Instructions

ALU micro-instructions control the functionality of the ALU, they are used to make the ALU perform a logical or arithmetic operation.

For example, the operation to add the contents of the ALUx and the ALUy registers is described as \( ALUr = [ALUx] + [ALUy] \).

### 16.2.3 Test Micro-Instructions

Test micro-instructions are used to allow assembly language programs to have conditional branches. For example, the assembly language instruction BEQ address would not be possible without the test micro-instruction to check the condition of the Z flag. Let's take a deeper look at that, here is the definition of the BEQ address instruction:

- \( if(PSR(z) == 1) \)
- \( PC \leftarrow [IR(operand)] \)

To perform the branch this is what happens: