Execution of a complete instruction

Complete Instruction Execution: Consider the instruction Add (R3),Rl , which adds the contents of a memory location pointed to by R3 to register R1. Executing this instruction requires the following actions:

1. Fetch the instruction.
2. Fetch the first operand (the contents of the memory location pointed to by R3).
3. Perform the addition.
4. Load the result into R1.

Figure 2.5 gives the sequence of control steps required to perform these operations for the singlebus architecture of Figure 2.1. Instruction execution proceeds as follows. In step 1, the instruction fetch operation is initiated by loading the contents of the PC into the MAR and sending a Read request to the memory. The Select signal is set to Select4, which causes the multiplexer MUX to select the constant 4. This value is added to the operand at input B, which is the contents of the PC, and the result is stored in register Z. The updated value is moved from register Z back into the PC during step 2, while waiting for the memory to respond. In step 3, the word fetched from the memory is loaded into the IR.

Steps 1 through 3 constitute the instruction fetch phase, which is the same for all instructions. The instruction decoding circuit interprets the contents of the IR at the beginning of step 4. This enables the control circuitry to activate the control signals for steps 4 through 7, which constitute the execution phase. The contents of register R3 are transferred to the MAR in step 4, and a memory read operation is initiated. Then

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Figure 2.5 Control sequence for execution of instruction Add (R3), Rl. the contents of R 1 are transferred to register Y in step 5, to prepare for the addition operation. When the Read operation is completed, the memory operand is available in register MDR, and the addition operation is performed in step 6. The contents of MDR are gated to the bus, and thus also to the B input of the ALU, and register Y is selected as the second input to the ALU by choosing Select Y The sum is stored in register Z, then transferred to R 1 in step 7. The End signal causes a new instruction fetch cycle to begin by returning to step 1.

This discussion accounts for all control signals in Figure 2.5 except Y in in step 2. There is no need to copy the updated contents of PC into register Y when executing the Add instruction. But, in Branch instructions the updated value of the PC is needed to compute the Branch target address. To speed up the execution of Branch instructions, this value is copied into register Y in step 2. Since step 2 is part of the fetch phase, the same action will be performed for all instructions. This does not cause any harm because register Y is not used for any other purpose at that time.

BRANCH INSTRUCTION: A branch instruction replaces the contents of the PC with the branch target address. This address is usually obtained by adding an offset X, which is given in the branch instruction, to the updated value of the PC. Figure 2.6 gives a control sequence that implements an unconditional branch instruction. Processing starts, as usual, with the fetch phase. This phase ends when the instruction is loaded into the IR in step 3. The offset value is extracted from the IR by the instruction decoding circuit, which will also perform sign extension if required. Since the value of the updated PC is already available in register Y, the offset X is gated onto the bus in step 4, and an addition operation is performed. The result, which is the branch target address, is loaded into the PC in

step 5.

The offset X used in a branch instruction is usually the difference between the branch target address and the address immediately following the branch instruction. For example, if the branch instruction is at location 2000 and if the branch target address is 2050, the value of X must be 46. The reason for this can be readily appreciated from the control sequence in Figure 2.6. The PC is incremented during the fetch phase, before knowing the type of instruction being executed. Thus, when the branch address is computed in step 4, the PC value used is the updated value, which points to the instruction following the branch instruction in the memory.

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	Offset-field-of-IR_{out}, Add, Z_{in}
5	Z_{out}, PC_{in}, End

Figure 2.6 Control sequence for an unconditional instruction.

Consider now a conditional branch. In this case, we need to check the status of the condition codes before loading a new value into the PC. For example, for a Branch-on-negative (Branch <0) instruction, step 4 in Figure 2.6 is replaced with

Offset-field-of-IR_{out}, Add, Z_{in}, If N = 0 then End

Thus, if N = 0 the processor returns to step 1 immediately after step 4. If N = 1, step 5 is performed to load a new value into the PC, thus performing the branch operation.