

Instruction Issue in the IBM 360/91

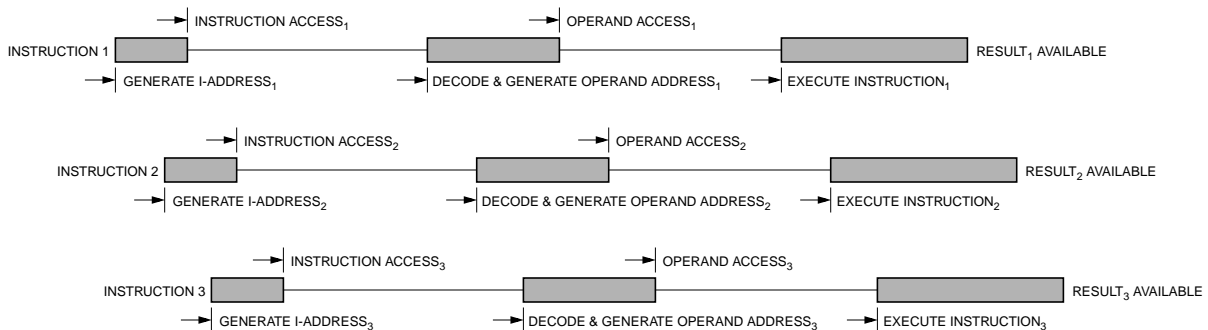
ENEE 759M: Advanced Topics in Microarchitecture, Spring 2000

Prof. Bruce Jacob

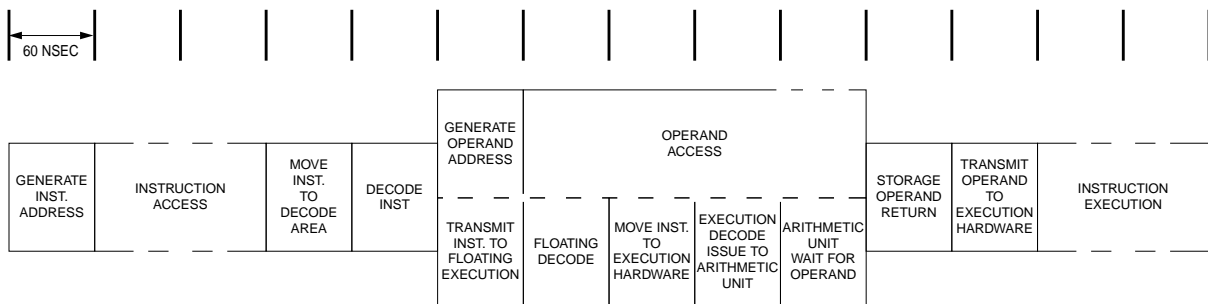
This is a guess (my guess) as to the implementation of the instruction issue mechanism in IBM’s System/360 Model 91, fixed-point pipeline. The guess is based on the text and figures 2, 3 and 6 in the article “The IBM System/360 Model 91: Machine Philosophy and Instruction-Handling” by Anderson, et al.

The fundamental problem is this: how does the system know when a given instruction may write to the general-purpose register file? The pipeline has in-order issue, out-of-order execution and completion, and it synchronizes through the register file: instructions reading their operands from the register file do not obtain them from forwarding paths. The pipeline enforces coherent writing to the register file by scheduling when instructions that would otherwise cause a write-after-write hazard are allowed access to the register file. The article mentions that each instruction that writes to a GPR increments a counter associated with that register during decode and decrements that counter at the time of register file update, and the article says that no instruction may read an operand from a GPR unless its associated counter has returned to zero. What is missing is the mechanism by which an instruction knows that it is its turn to write its result into the GPR.

The paper does not give a detailed diagram of the fixed-point pipeline ... the diagrams are a bit simple, but this is probably enough detail (taken from figure 2):

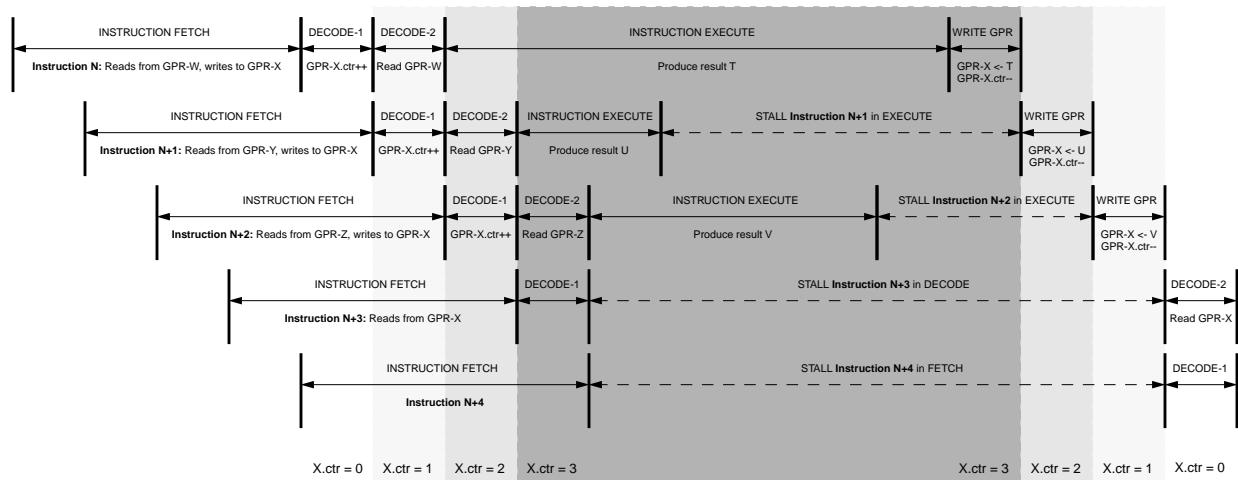


For comparison, here is the floating-point pipeline, taken from figure 3:



We’ll assume for the moment that the fixed-point pipeline is similar to the floating-point pipeline, as is suggested by the general flow shown in figure 2. Here is the behavior of four instructions in

the fixed-point pipeline, three of which write to GPR X, the last of which reads from GPR X. This figure is adapted (and extrapolated) from figure 6:



The question: how does an instruction “know” when it is allowed to write to a GPR? For instance, in the above example, instruction N completes execution after instruction N+1, but instruction N+1 delays writing its result to the register file until N completes. How is this done?

Here is my guess. The pipeline registers that hold the state for an instruction also hold the value of the CTR variable for the GPR that the instruction targets. In the preceding example, instruction N would retain the value “0”; instruction N+1 would retain the value “1”; and instruction N+2 would retain the value “2”. These values would be updated whenever another instruction updates the same register in the general-purpose register file. For instance, when instruction N writes its result to GPR X, it broadcasts that fact—i.e. the value “X”—on a result bus. Note that only one result can be written to the register file at a time, and the register number is a small value requiring only a few bits, so this is relatively efficient. Any instruction that targets the register GPR X would see this broadcast and decrement the value for CTR that it is holding. Once that value reaches zero, the instruction may write its result back to the register file. This is illustrated below.

