Chapter 03

Authors: John Hennessy & David Patterson

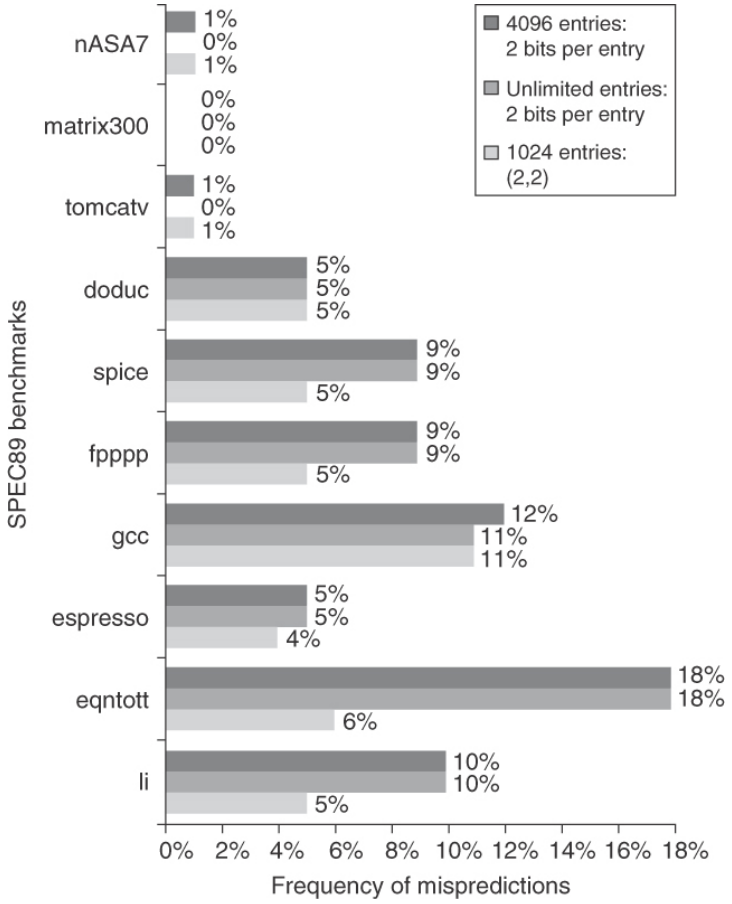

Figure 3.3 Comparison of 2-bit predictors. A noncorrelating predictor for 4096 bits is first, followed by a noncorrelating 2-bit predictor with unlimited entries and a 2-bit predictor with 2 bits of global history and a total of 1024 entries. Although these data are for an older version of SPEC, data for more recent SPEC benchmarks would show similar differences in accuracy.

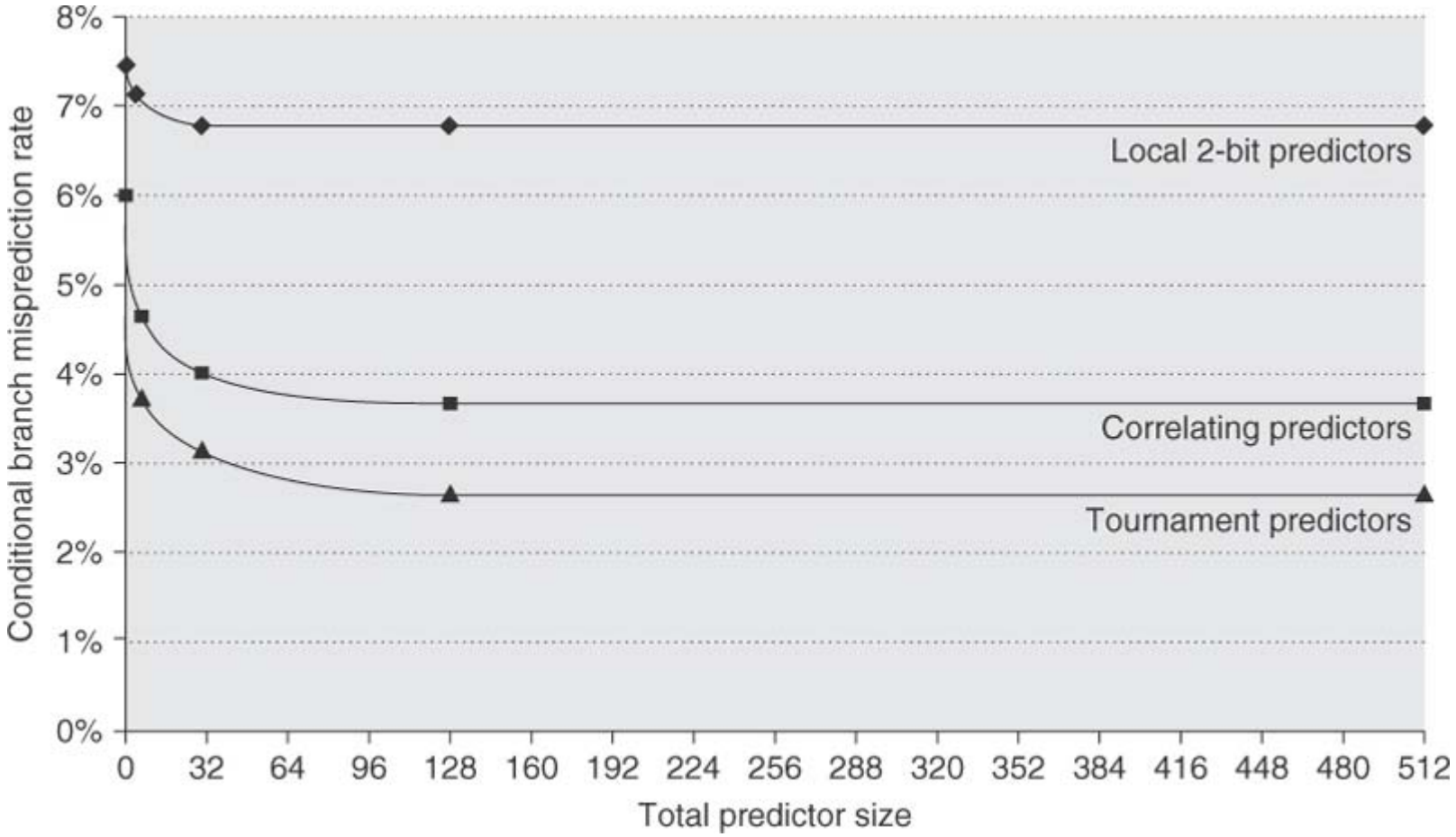

Figure 3.4 The misprediction rate for three different predictors on SPEC89 as the total number of bits is increased. The predictors are a local 2-bit predictor, a correlating predictor that is optimally structured in its use of global and local information at each point in the graph, and a tournament predictor. Although these data are for an older version of SPEC, data for more recent SPEC benchmarks would show similar behavior, perhaps converging to the asymptotic limit at slightly larger predictor sizes.

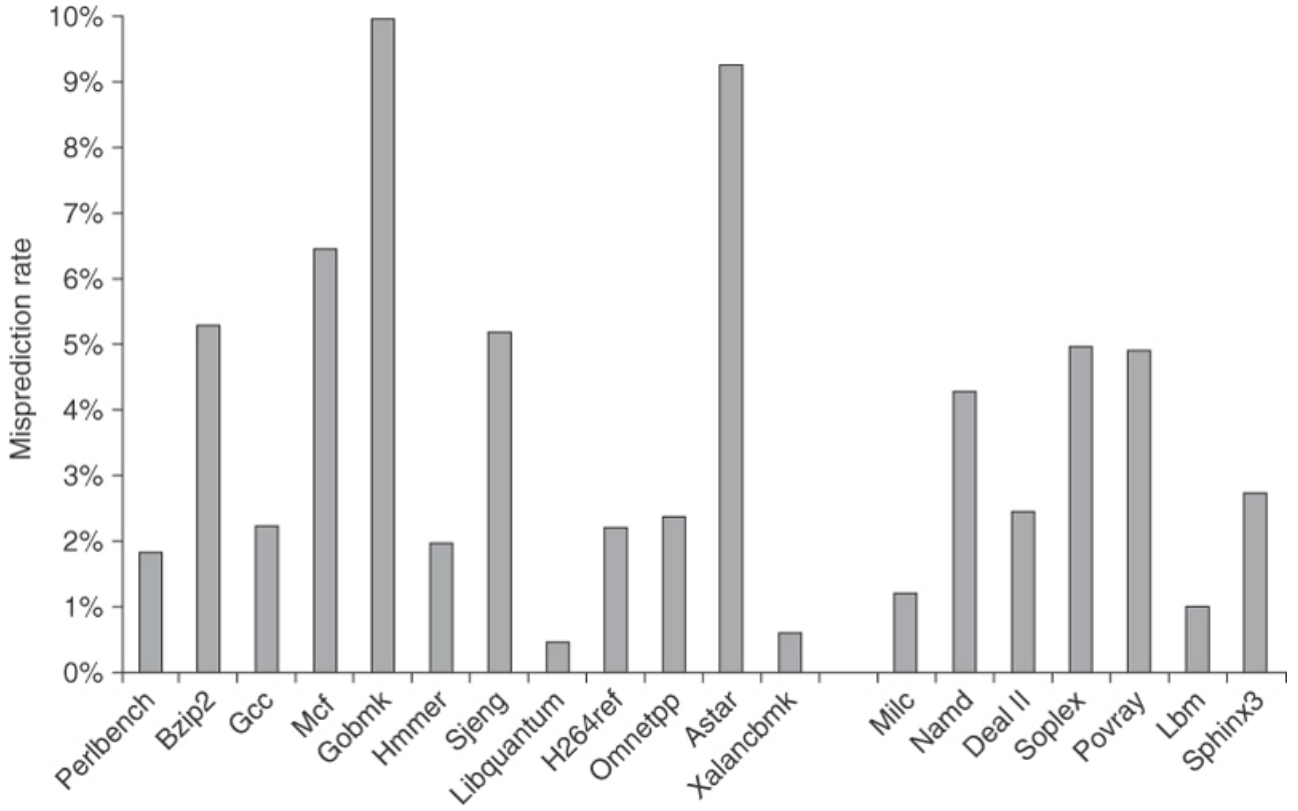

Figure 3.5 The misprediction rate for 19 of the SPECCPU2006 benchmarks versus the number of successfully retired branches is slightly higher on average for the integer benchmarks than for the FP (4% versus 3%). More importantly, it is much higher for a few benchmarks.

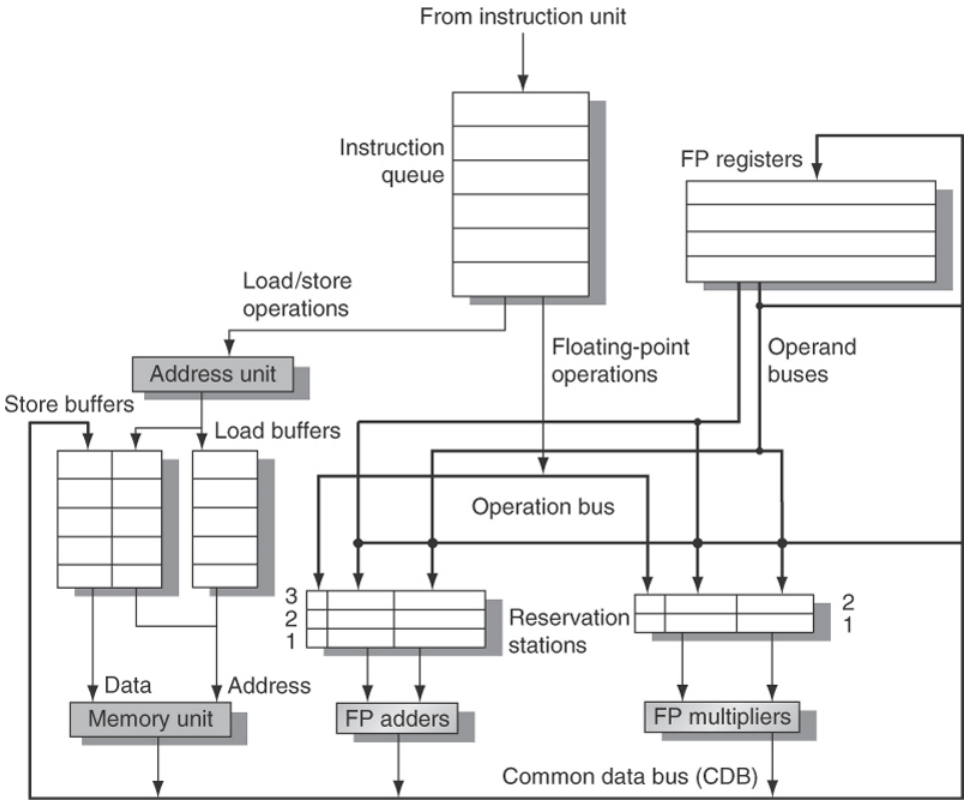

Figure 3.6 The basic structure of a MIPS floating-point unit using Tomasulo's algorithm. Instructions are sent from the instruction unit into the instruction queue from which they are issued in first-in, first-out (FIFO) order. The reservation stations include the operation and the actual operands, as well as information used for detecting and resolving hazards. Load buffers have three functions: (1) hold the components of the effective address until it is computed, (2) track outstanding loads that are waiting on the memory, and (3) hold the results of completed loads that are waiting for the CDB. Similarly, store buffers have three functions: (1) hold the components of the effective address until it is computed, (2) hold the destination memory addresses of out-standing stores that are waiting for the data value to store, and (3) hold the address and value to store until the memory unit is available. All results from either the FP units or the load unit are put on the CDB, which goes to the FP register file as well as to the reservation stations and store buffers. The FP adders implement addition and subtraction, and the FP multipliers do multiplication and division.

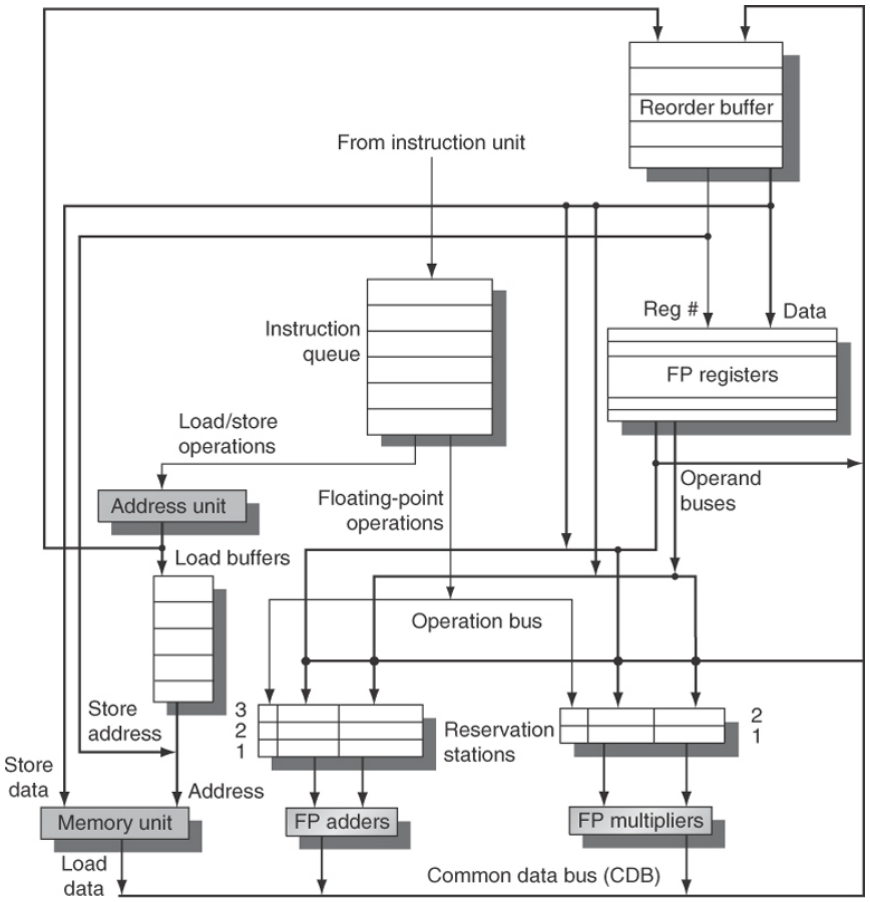

Figure 3.11 The basic structure of a FP unit using Tomasulo's algorithm and extended to handle speculation. Comparing this to Figure 3.6 on page 173, which implemented Tomasulo's algorithm, the major change is the addition of the ROB and the elimination of the store buffer, whose function is integrated into the ROB. This mechanism can be extended to multiple issue by making the CDB wider to allow for multiple completions per clock.

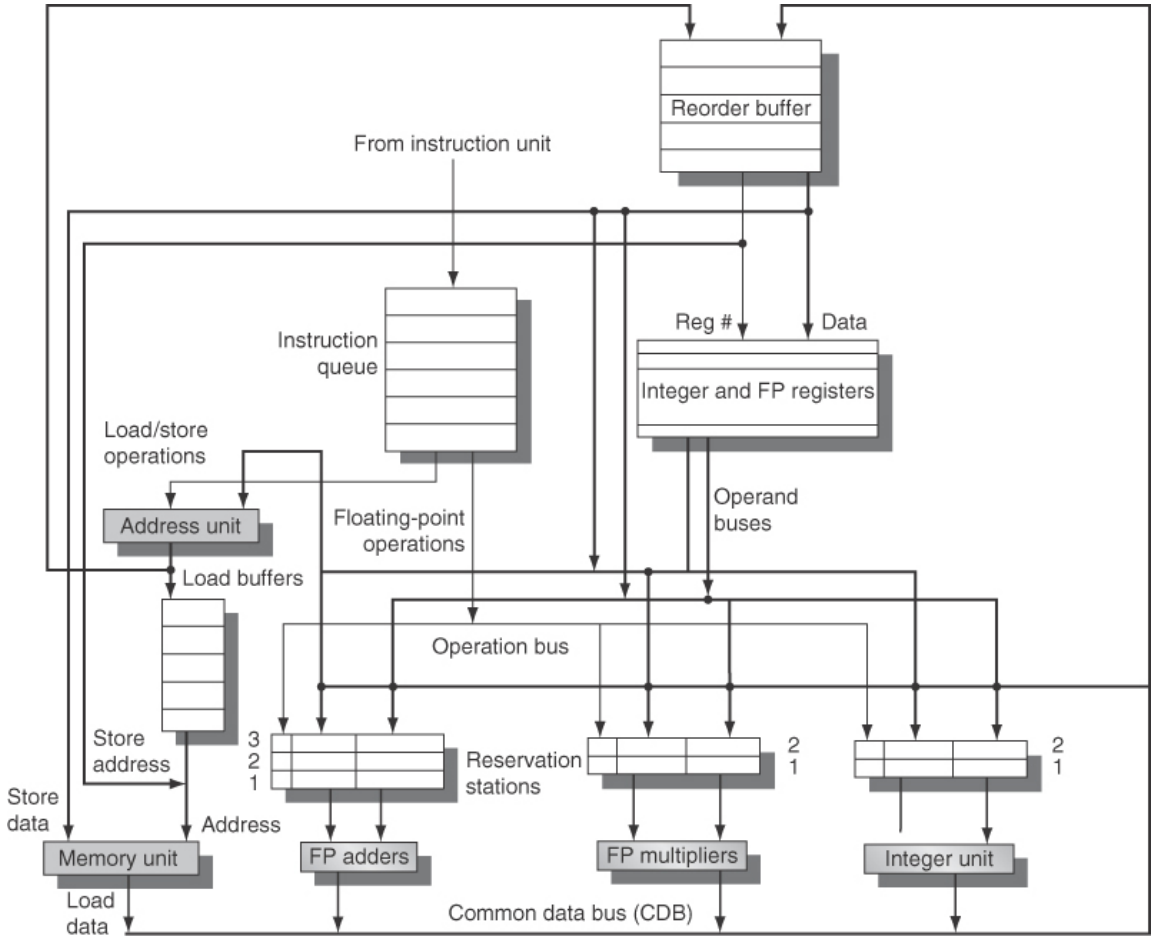

Figure 3.17 The basic organization of a multiple issue processor with speculation. In this case, the organization could allow a FP multiply, FP add, integer, and load/store to all issues simultaneously (assuming one issue per clock per functional unit). Note that several datapaths must be widened to support multiple issues: the CDB, the operand buses, and, critically, the instruction issue logic, which is not shown in this figure. The last is a difficult problem, as we discuss in the text.

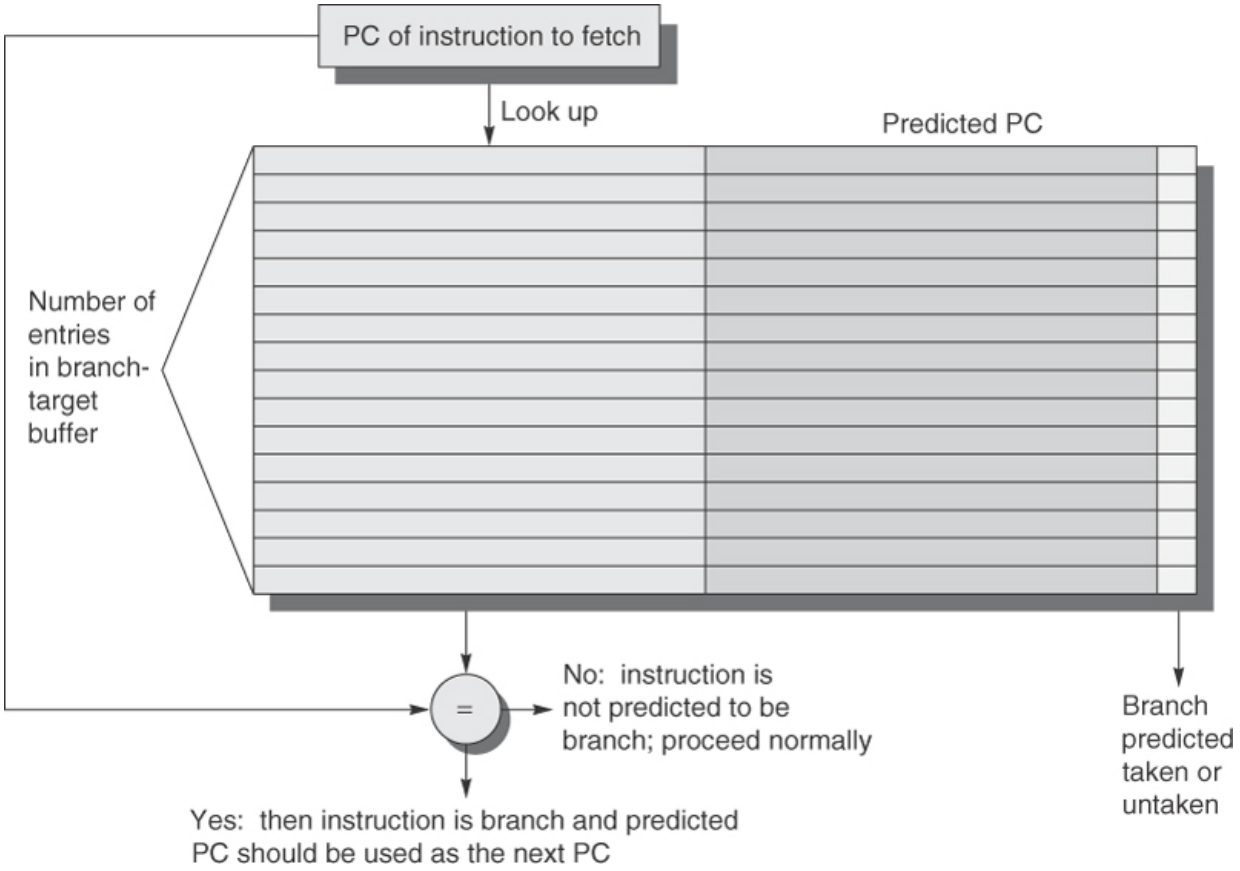

Figure 3.21 A branch-target buffer. The PC of the instruction being fetched is matched against a set of instruction addresses stored in the first column; these represent the addresses of known branches. If the PC matches one of these entries, then the instruction being fetched is a taken branch, and the second field, predicted PC, contains the prediction for the next PC after the branch. Fetching begins immediately at that address. The third field, which is optional, may be used for extra prediction state bits.

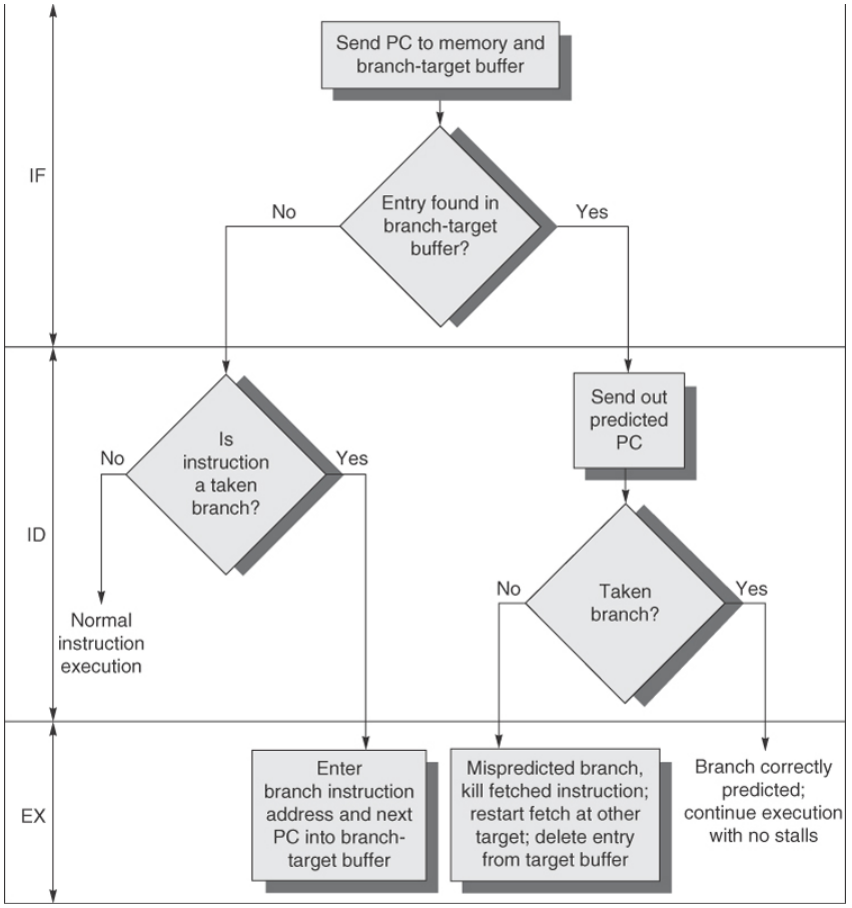

Figure 3.22 The steps involved in handling an instruction with a branch-target buffer.

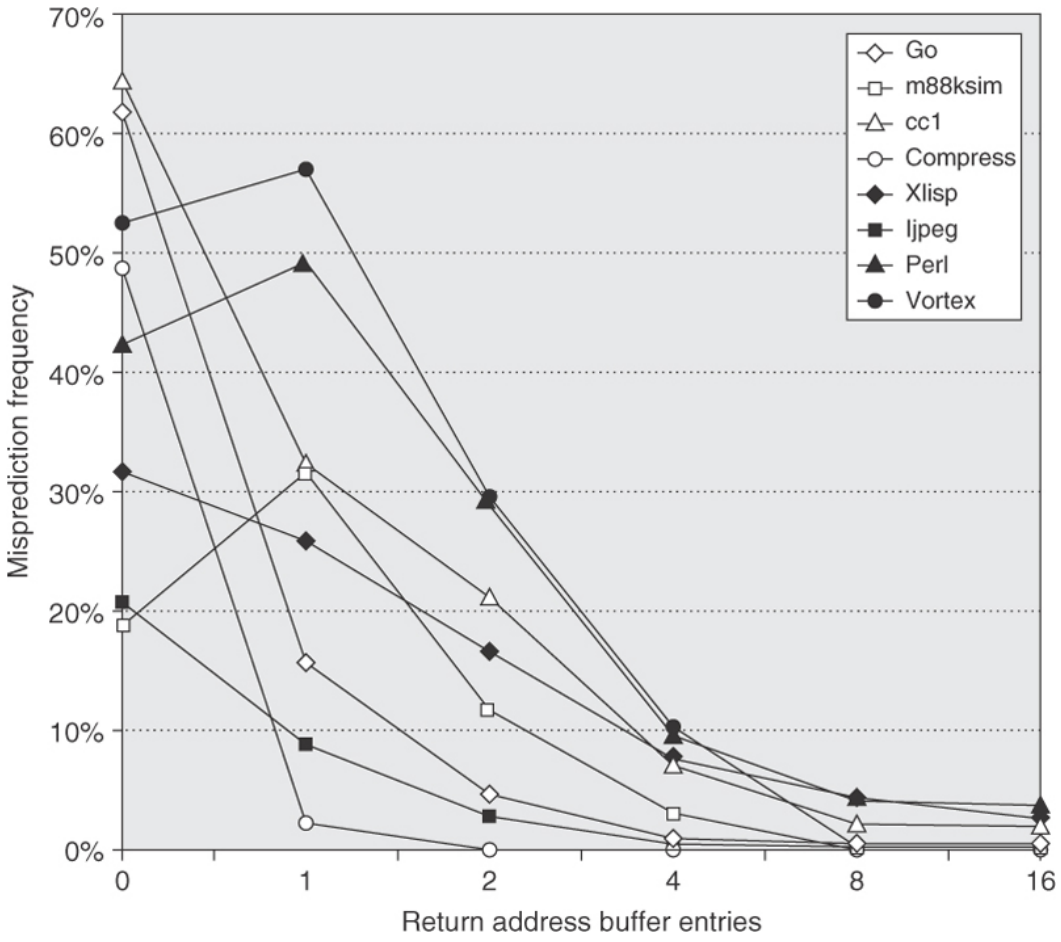

Figure 3.24 Prediction accuracy for a return address buffer operated as a stack on a number of SPEC CPU95 benchmarks. The accuracy is the fraction of return addresses predicted correctly. A buffer of 0 entries implies that the standard branch prediction is used. Since call depths are typically not large, with some exceptions, a modest buffer works well. These data come from Skadron et al. [1999] and use a fix-up mechanism to prevent corruption of the cached return addresses.

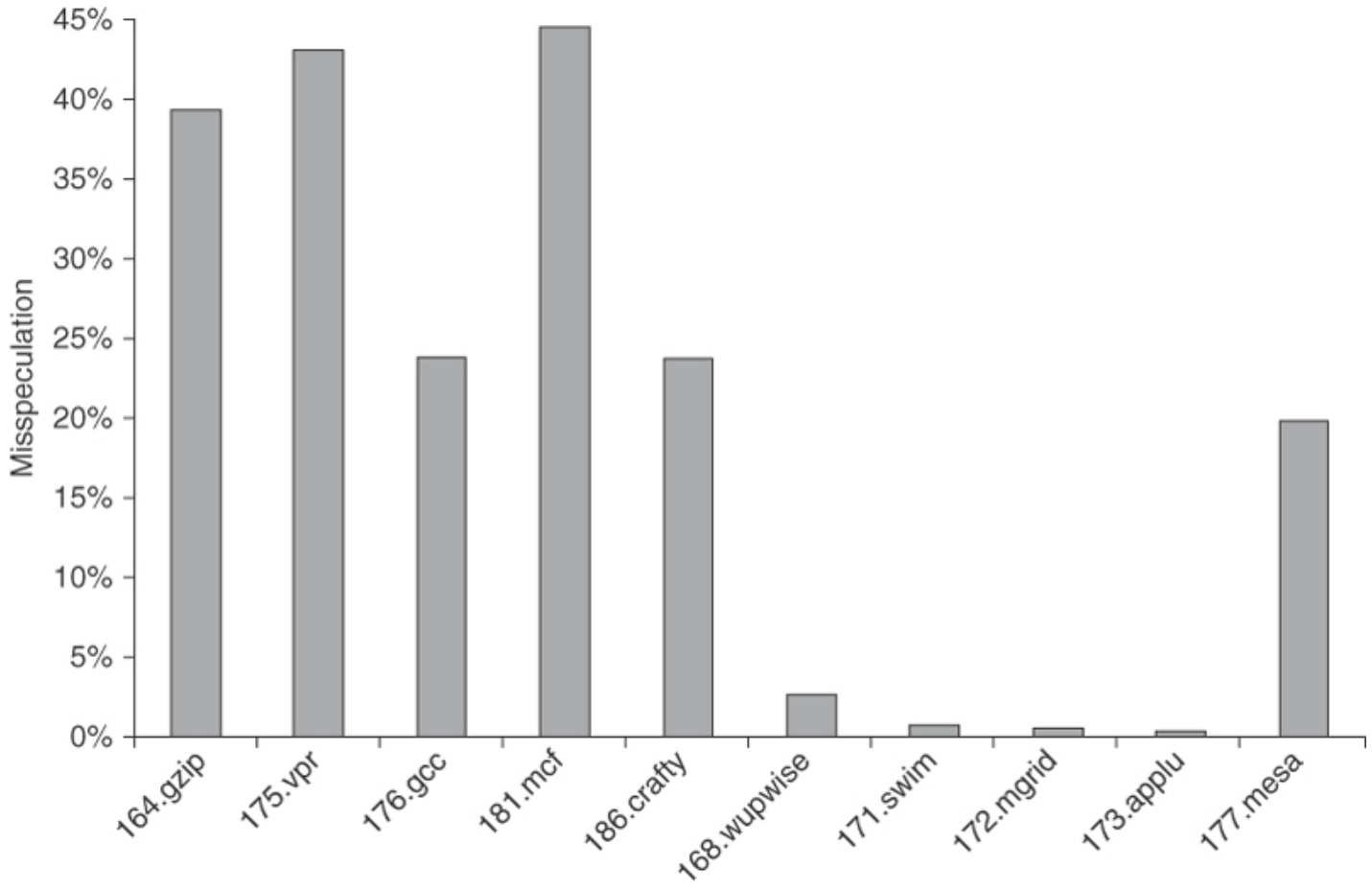

Figure 3.25 The fraction of instructions that are executed as a result of misspeculation is typically much higher for integer Programs (the first five) versus FP programs (the last five).

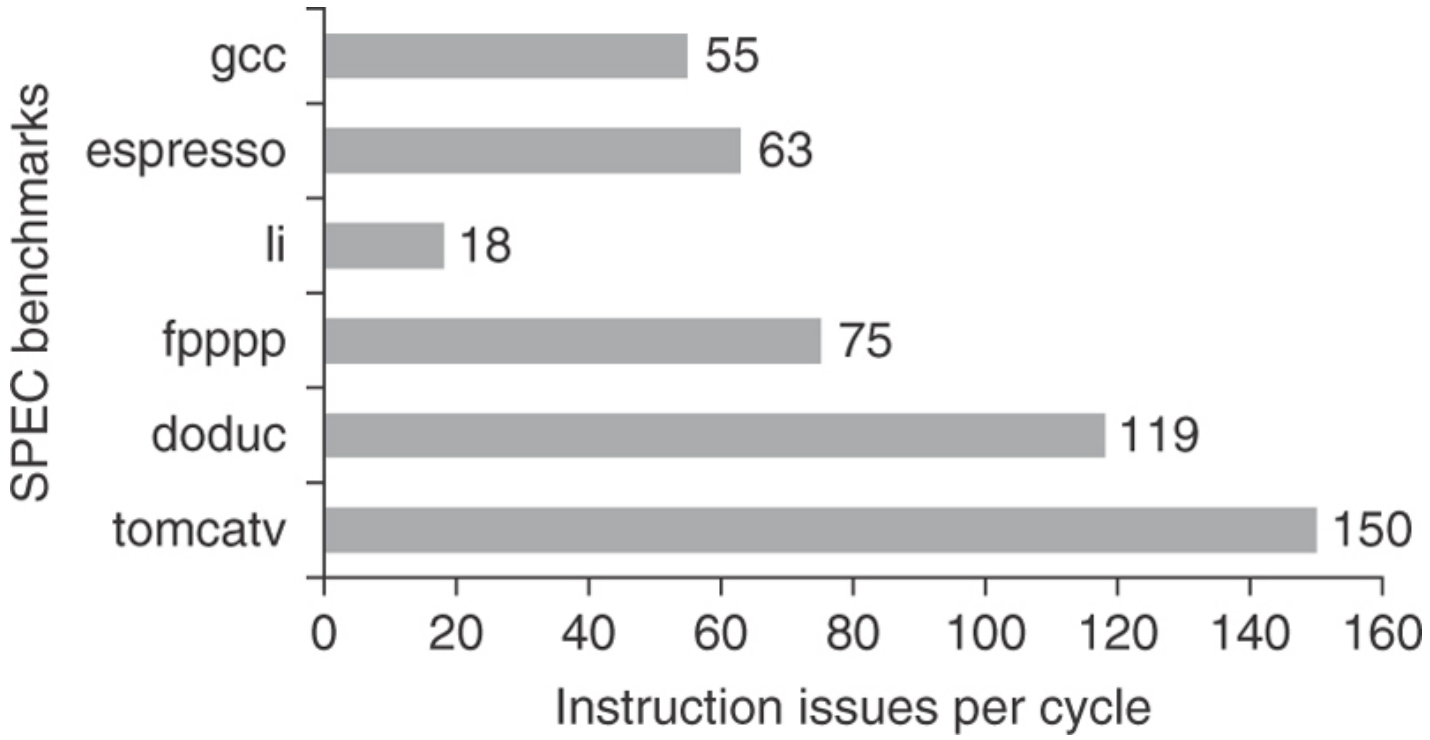

Figure 3.26 ILP available in a perfect processor for six of the SPEC92 benchmarks. The first three programs are integer programs, and the last three are floating-point programs. The floating-point programs are loop intensive and have large amounts of loop-level parallelism.

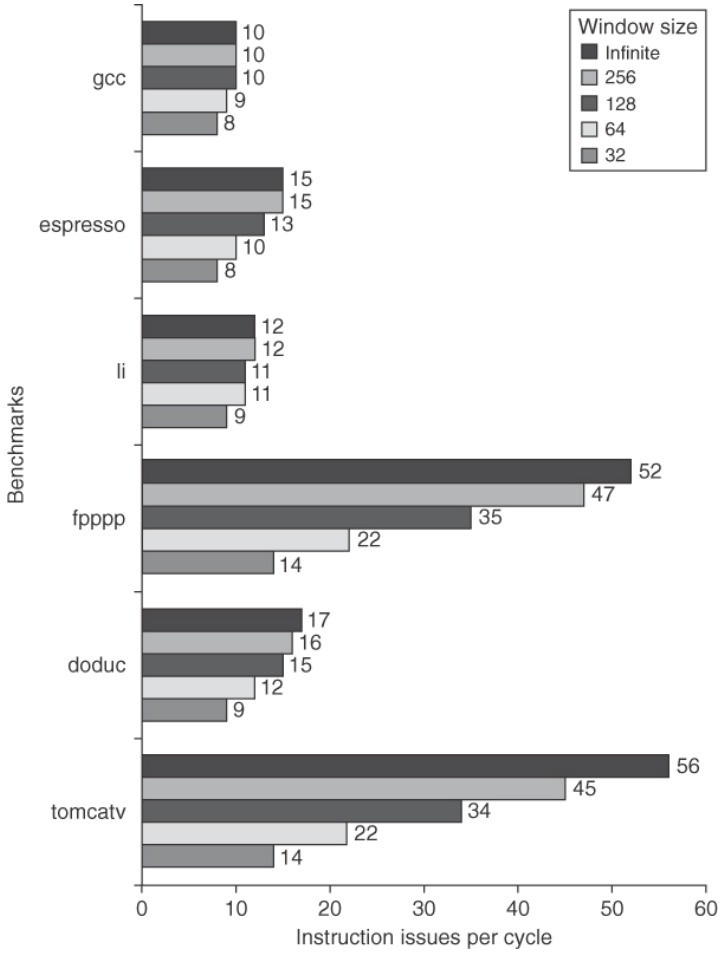

Figure 3.27 The amount of parallelism available versus the window size for a variety of integer and floating-point programs with up to 64 arbitrary instruction issues per clock. Although there are fewer renaming registers than the window size, the fact that all operations have one-cycle latency and the number of renaming registers equals the issue width allows the processor to exploit parallelism within the entire window. In a real implementation, the window size and the number of renaming registers must be balanced to prevent one of these factors from overly constraining the issue rate.

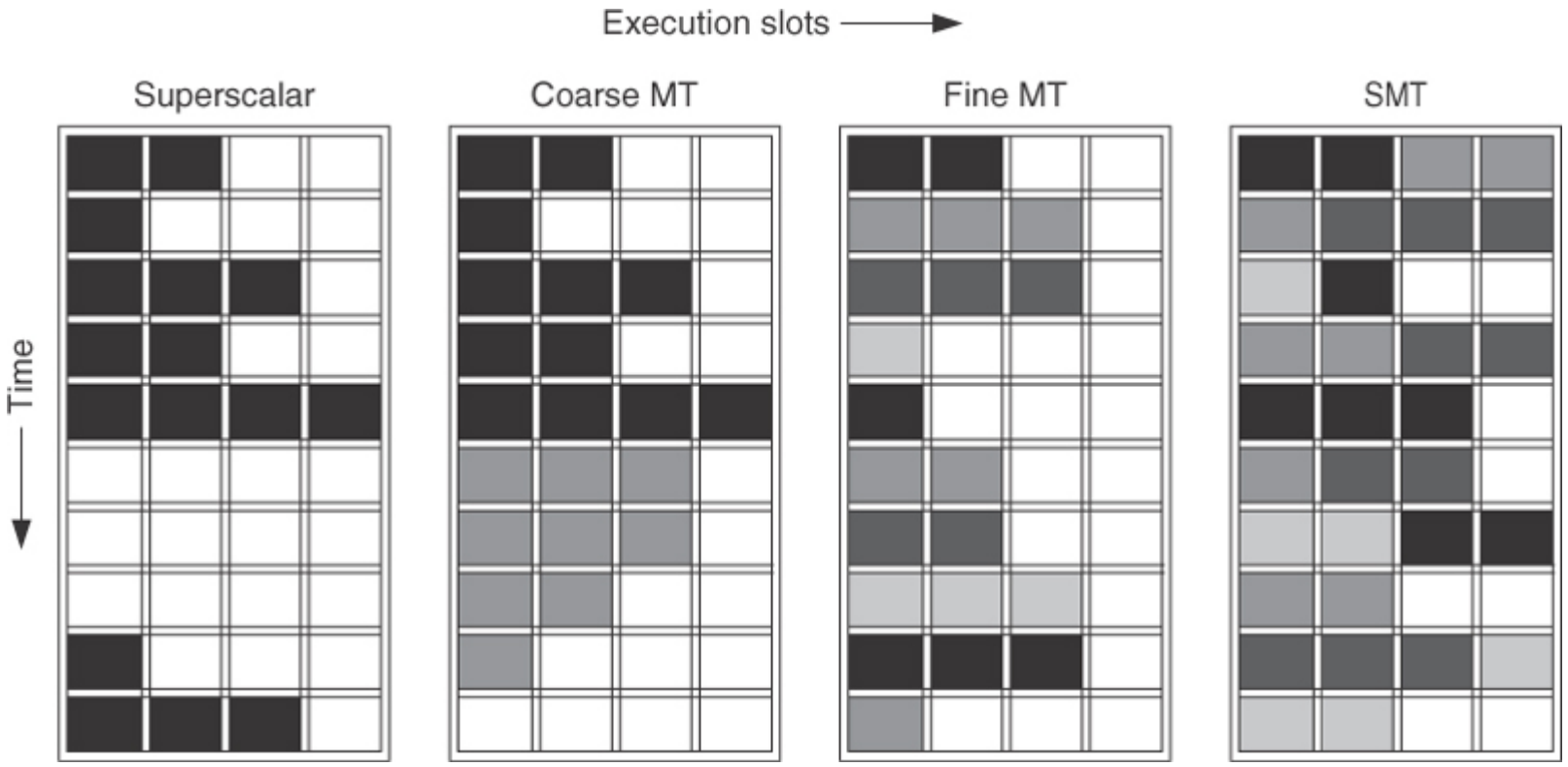

Figure 3.28 How four different approaches use the functional unit execution slots of a superscalar processor. The horizontal dimension represents the instruction execution capability in each clock cycle. The vertical dimension represents a sequence of clock cycles. An empty (white) box indicates that the corresponding execution slot is unused in that clock cycle. The shades of gray and black correspond to four different threads in the multithreading processors. Black is also used to indicate the occupied issue slots in the case of the superscalar without multithreading support. The Sun T1 and T2 (aka Niagara) processors are fine-grained multithreaded processors, while the Intel Core i7 and IBM Power7 processors use SMT. The T2 has eight threads, the Power7 has four, and the Intel i7 has two. In all existing SMTs, instructions issue from only one thread at a time. The difference in SMT is that the subsequent decision to execute an instruction is decoupled and could execute the operations coming from several different instructions in the same clock cycle.

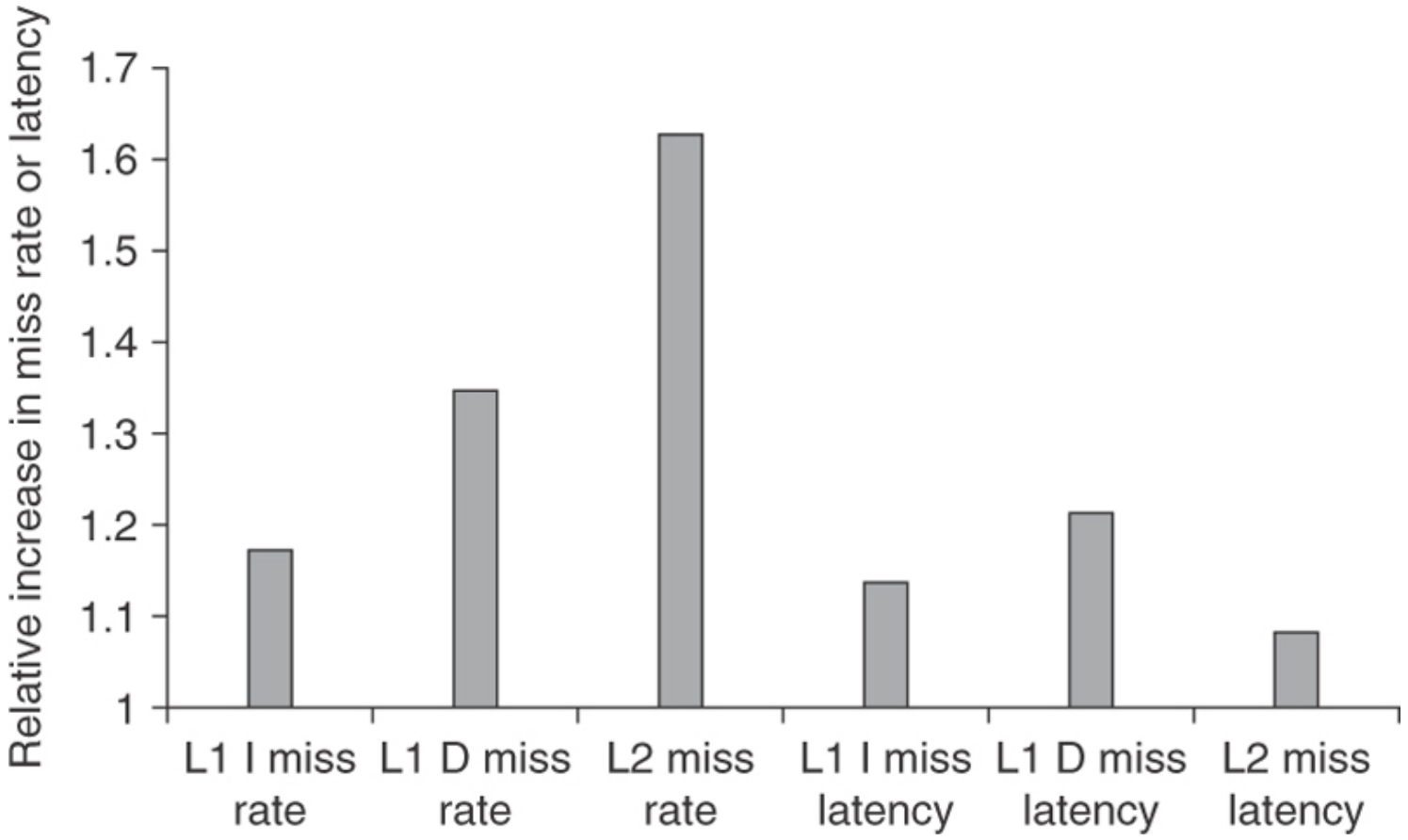

Figure 3.30 The relative change in the miss rates and miss latencies when executing with one thread per core versus four threads per core on the TPC-C benchmark. The latencies are the actual time to return the requested data after a miss. In the four-thread case, the execution of other threads could potentially hide much of this latency.

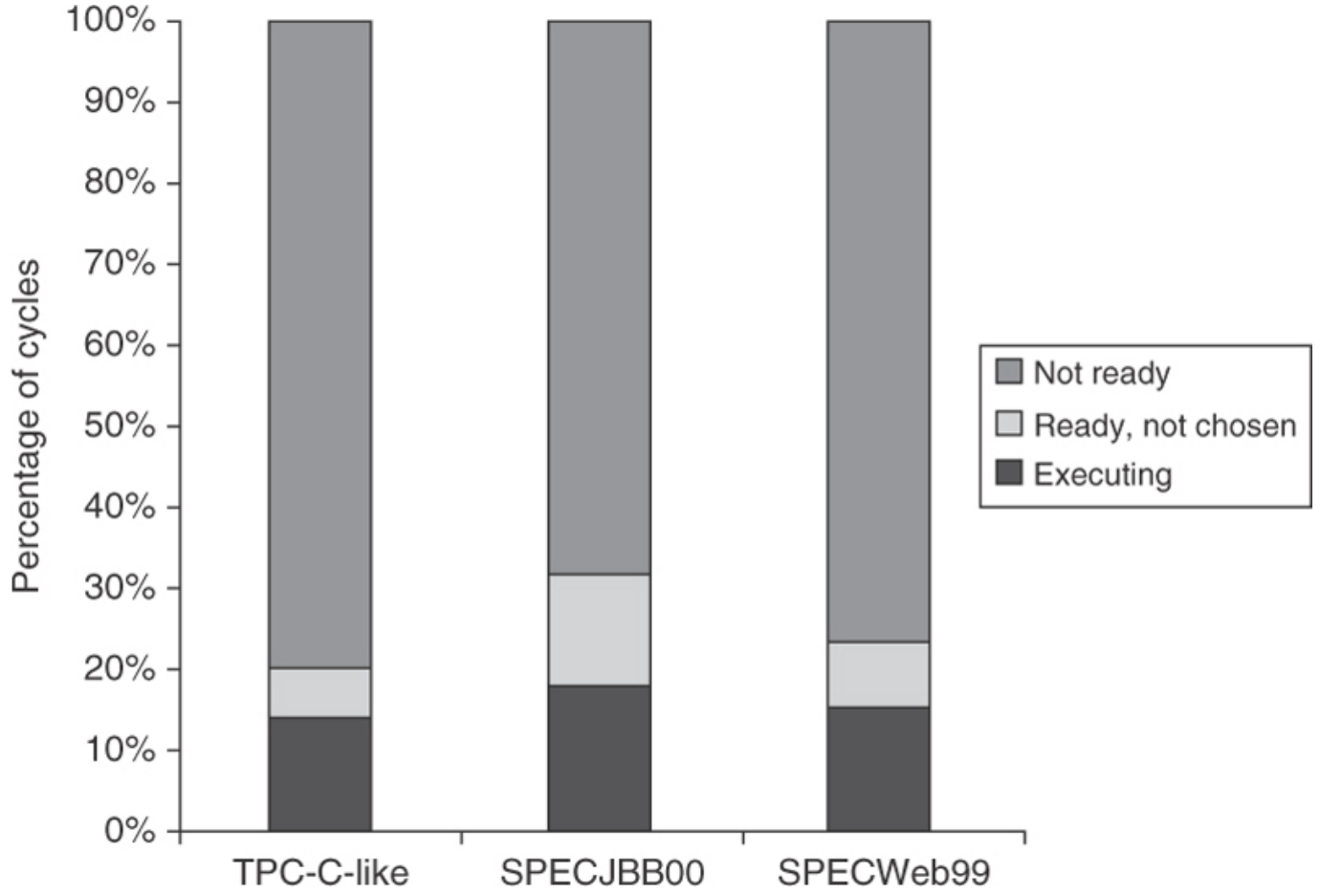

Figure 3.31 Breakdown of the status on an average thread. "Executing" indicates the thread issues an instruction in that cycle. "Ready but not chosen" means it could issue but another thread has been chosen, and "not ready" indicates that the thread is awaiting the completion of an event (a pipeline delay or cache miss, for example).

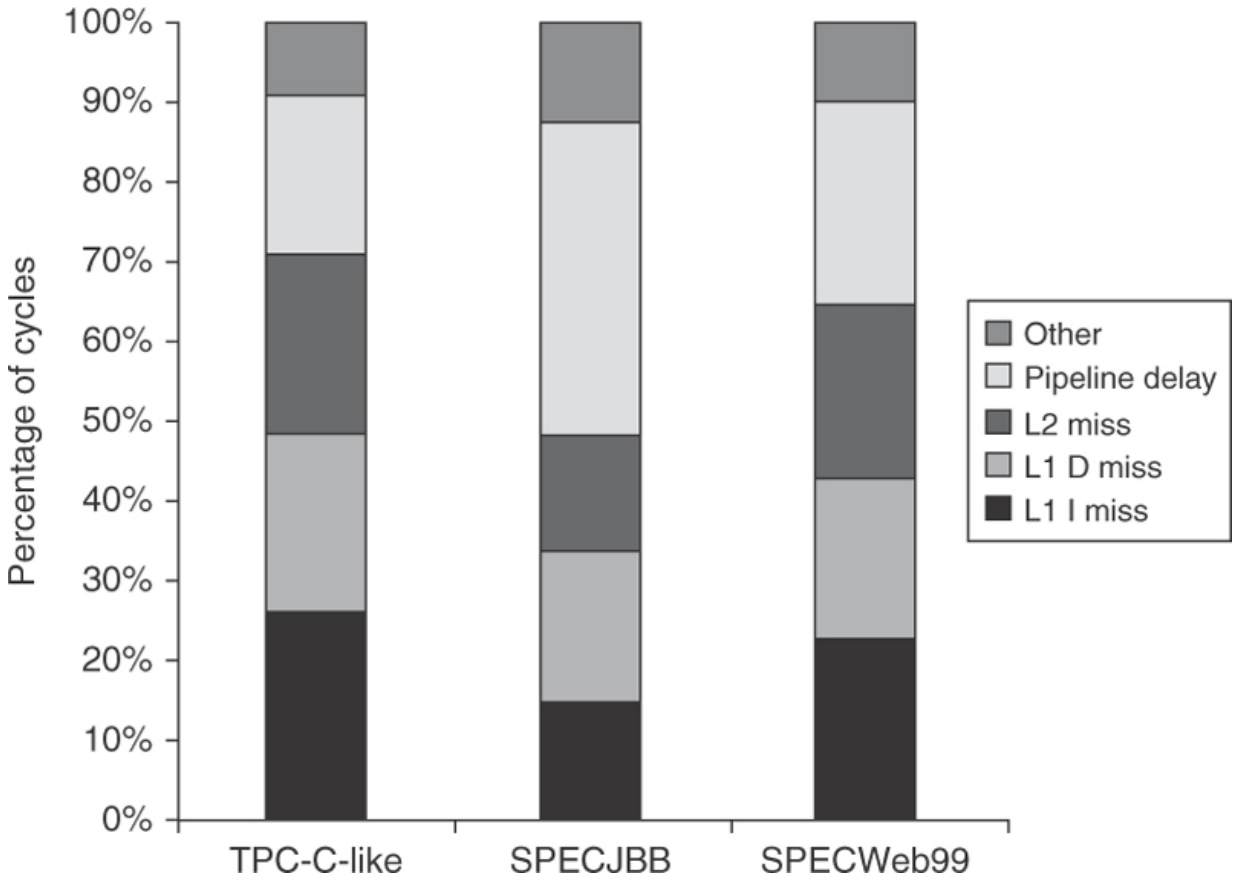

Figure 3.32 The breakdown of causes for a thread being not ready. The contribution to the "other" category varies. In PC-C, store buffer full is the largest contributor; in SPEC-JBB, atomic instructions are the largest contributor; and in SPECWeb99, both factors contribute.

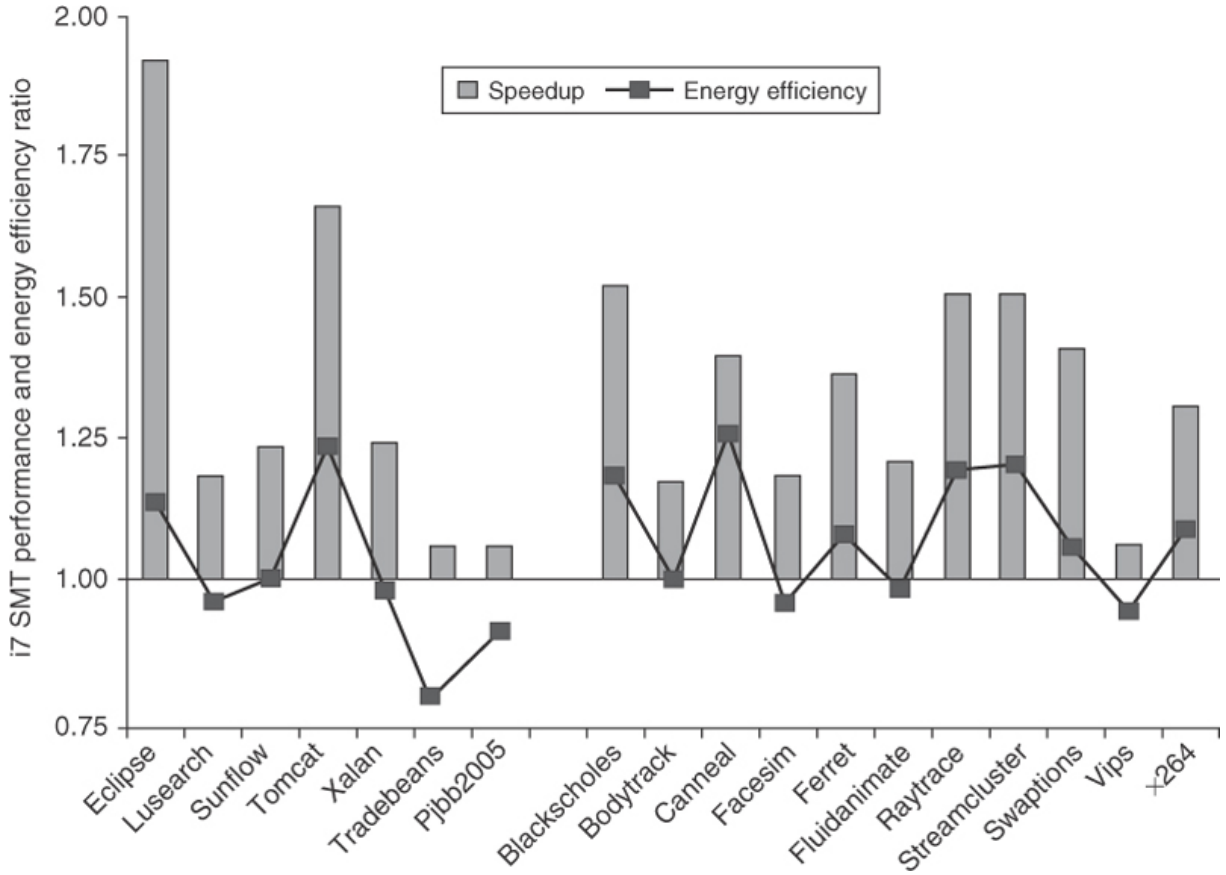

Figure 3.35 The speedup from using multithreading on one core on an i7 processor averages 1.28 for the Java benchmarks and 1.31 for the PARSEC benchmarks (using an unweighted harmonic mean, which implies a workload where the total time spent executing each benchmark in the single-threaded base set was the same). The energy efficiency averages 0.99 and 1.07, respectively (using the harmonic mean). Recall that anything above 1.0 for energy efficiency indicates that the feature reduces execution time by more than it increases average power. Two of the Java benchmarks experience little speedup and have significant negative energy efficiency because of this. Turbo Boost is off in all cases. These data were collected and analyzed by Esmaeilzadeh et al. [2011] using the Oracle (Sun) HotSpot build 16.3-b01 Java 1.6.0 Virtual Machine and the gcc v4.4.1 native compiler.

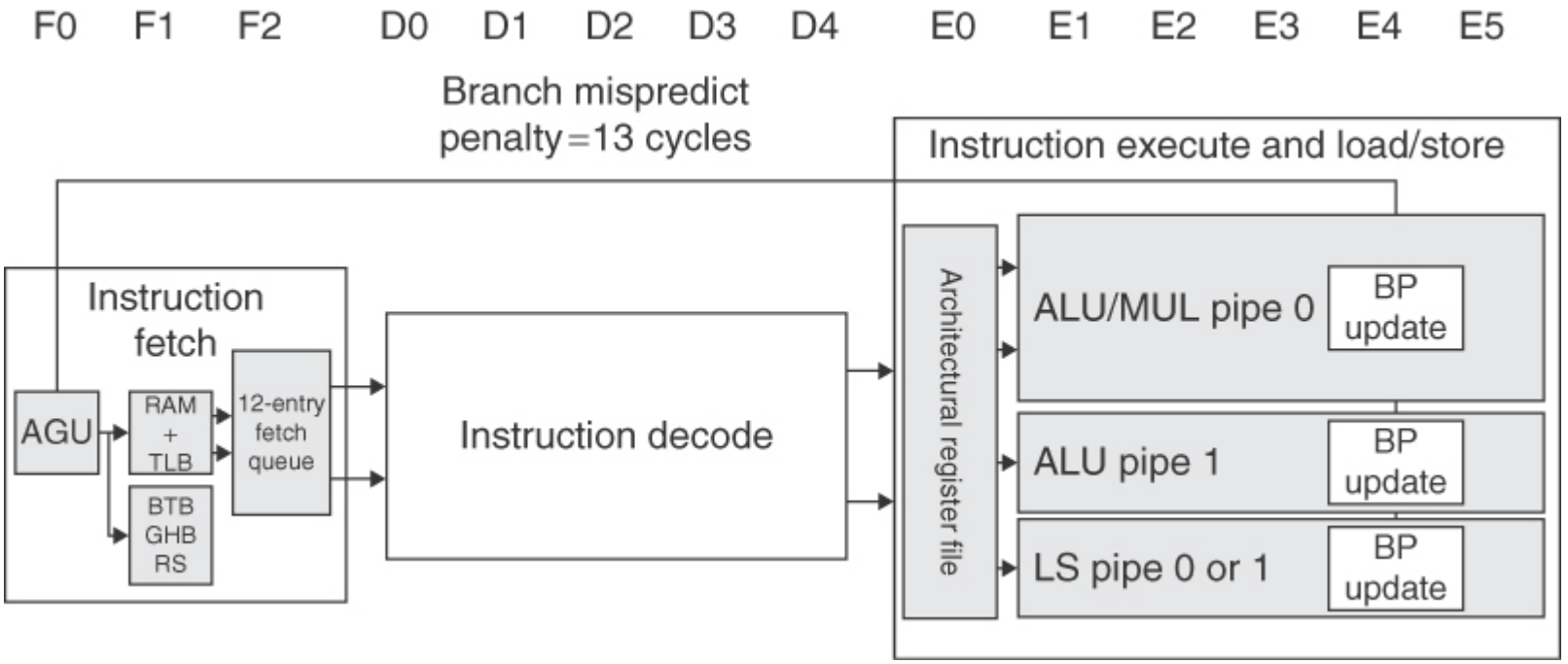

Figure 3.36 The basic structure of the A8 pipeline is 13 stages. Three cycles are used for instruction fetch and four for instruction decode, in addition to a five-cycle integer pipeline. This yields a 13-cycle branch misprediction penalty. The instruction fetch unit tries to keep the 12-entry instruction queue filled.

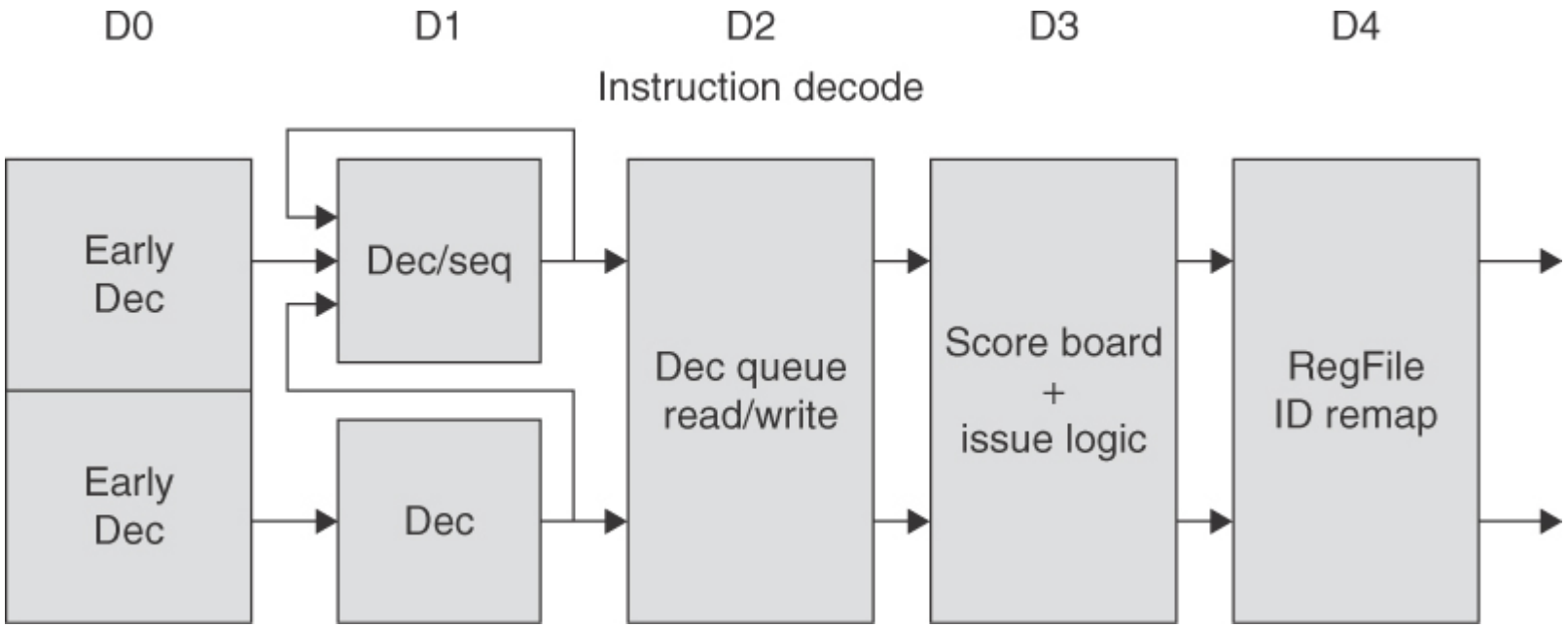

Figure 3.37 The five-stage instruction decode of the A8. In the first stage, a PC produced by the fetch unit (either from the branch target buffer or the PC incrementer) is used to retrieve an 8-byte block from the cache. Up to two instructions are decoded and placed into the decode queue; if neither instruction is a branch, the PC is incremented for the next fetch. Once in the decode queue, the scoreboard logic decides when the instructions can issue. In the issue, the register operands are read; recall that in a simple scoreboard, the operands always come from the registers. The register operands and opcode are sent to the instruction execution portion of the pipeline.

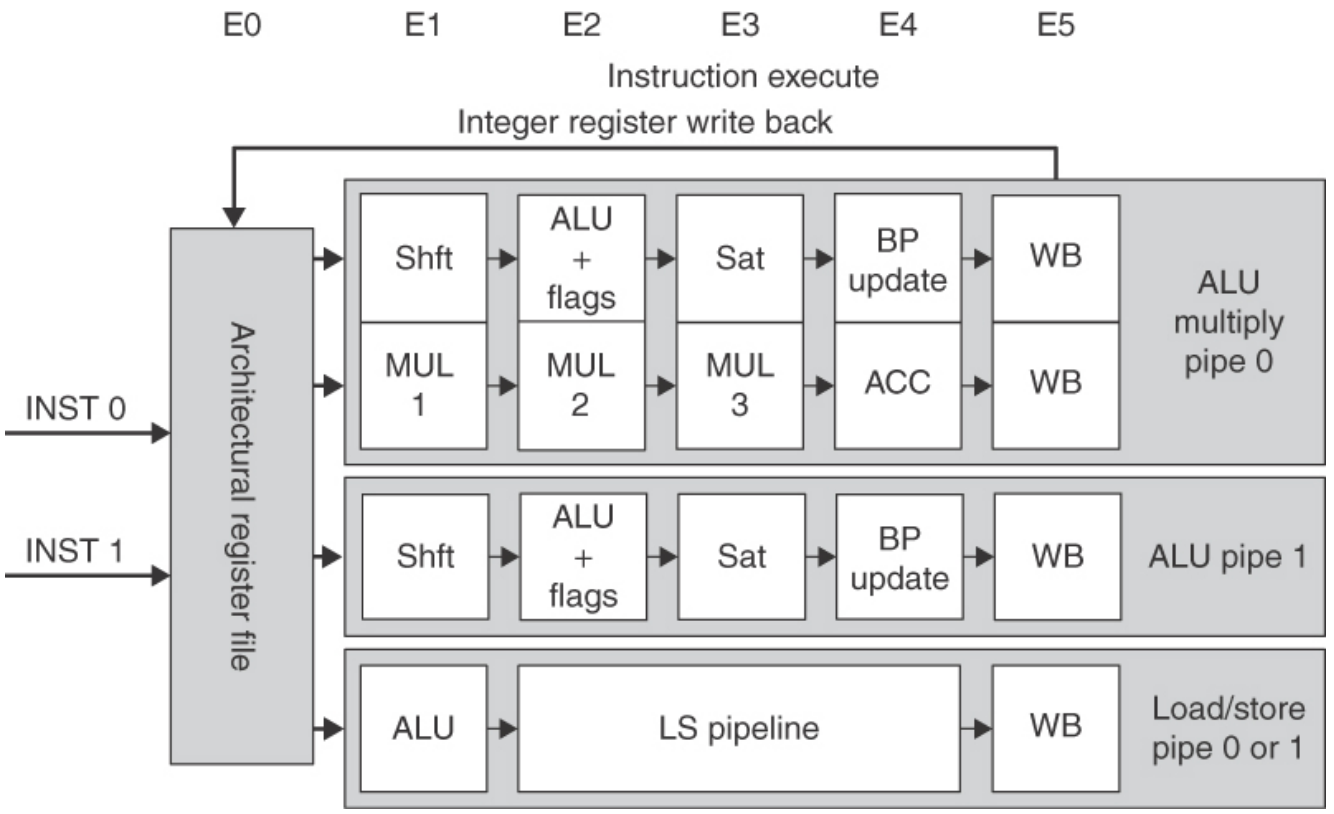

Figure 3.38 The five-stage instruction decode of the A8. Multiply operations are always performed in ALU pipeline 0.

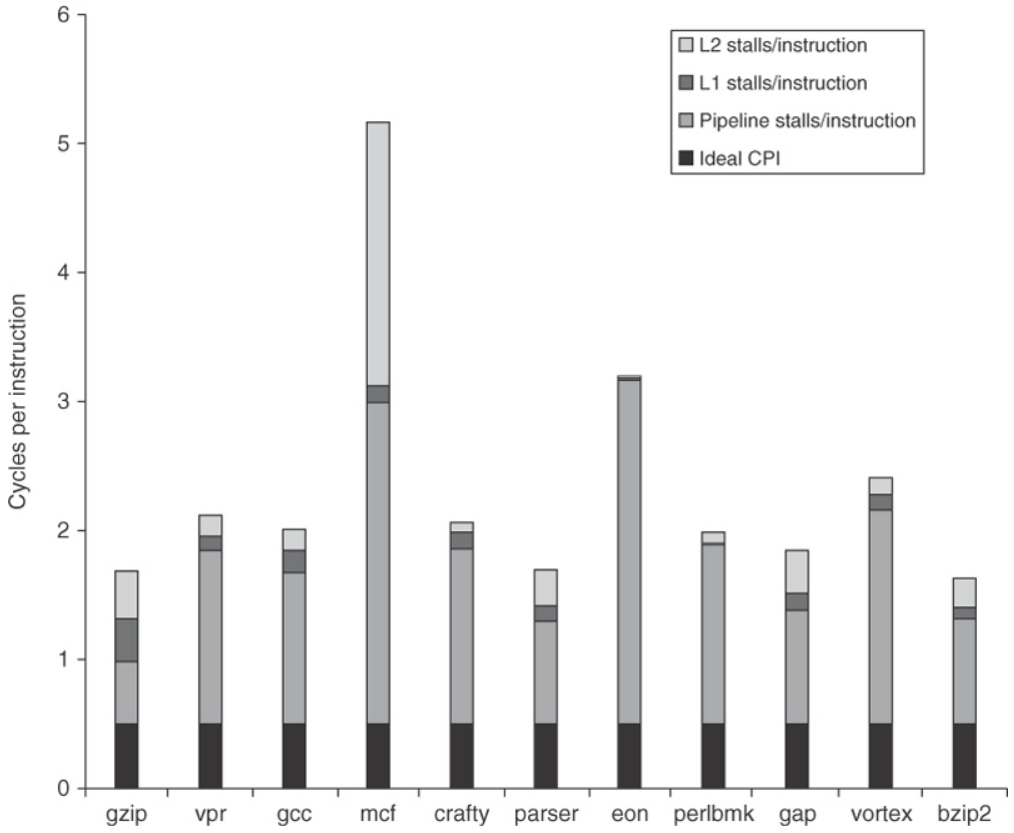

Figure 3.39 The estimated composition of the CPI on the ARM A8 shows that pipeline stalls are the primary addition to the base CPI. eon deserves some special mention, as it does integer-based graphics calculations (ray tracing) and has very few cache misses. It is computationally intensive with heavy use of multiples, and the single multiply pipeline becomes a major bottleneck. This estimate is obtained by using the L1 and L2 miss rates and penalties to compute the L1 and L2 generated stalls per instruction. These are subtracted from the CPI measured by a detailed simulator to obtain the pipeline stalls. Pipeline stalls include all three hazards plus minor effects such as way misprediction.

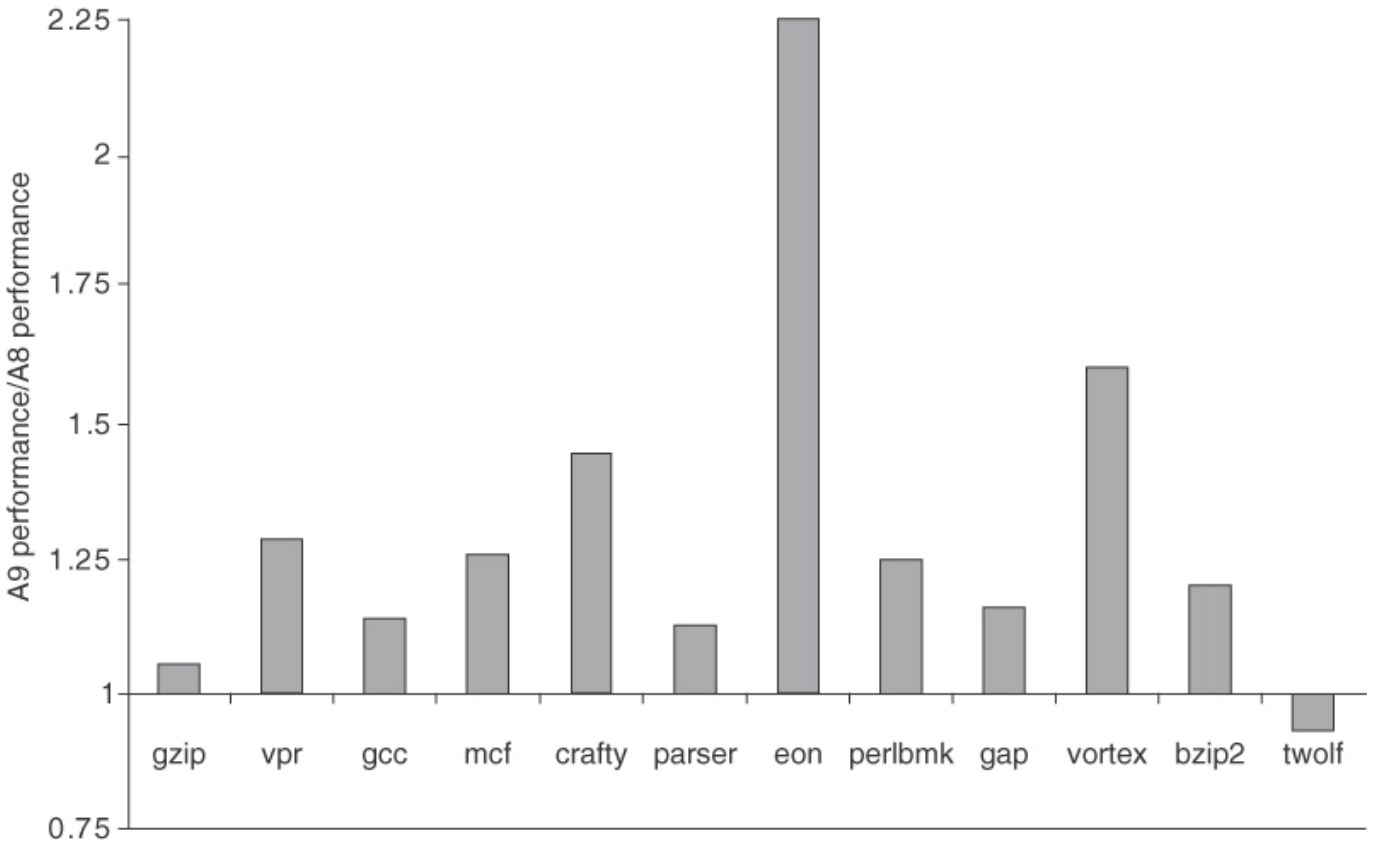

Figure 3.40 The performance ratio for the A9 compared to the A8, both using a 1 GHz clock and the same size caches for L1 and L2, shows that the A9 is about 1.28 times faster. Both runs use a 32 KB primary cache and a 1 MB secondary cache, which is 8-way set associative for the A8 and 16-way for the A9. The block sizes in the caches are 64 bytes for the A8 and 32 bytes for the A9. As mentioned in the caption of Figure 3.39, eon makes intensive use of integer multiply, and the combination of dynamic scheduling and a faster multiply pipeline significantly improves performance on the A9. twolf experiences a small slowdown, likely due to the fact that its cache behavior is worse with the smaller L1 block size of the A9.

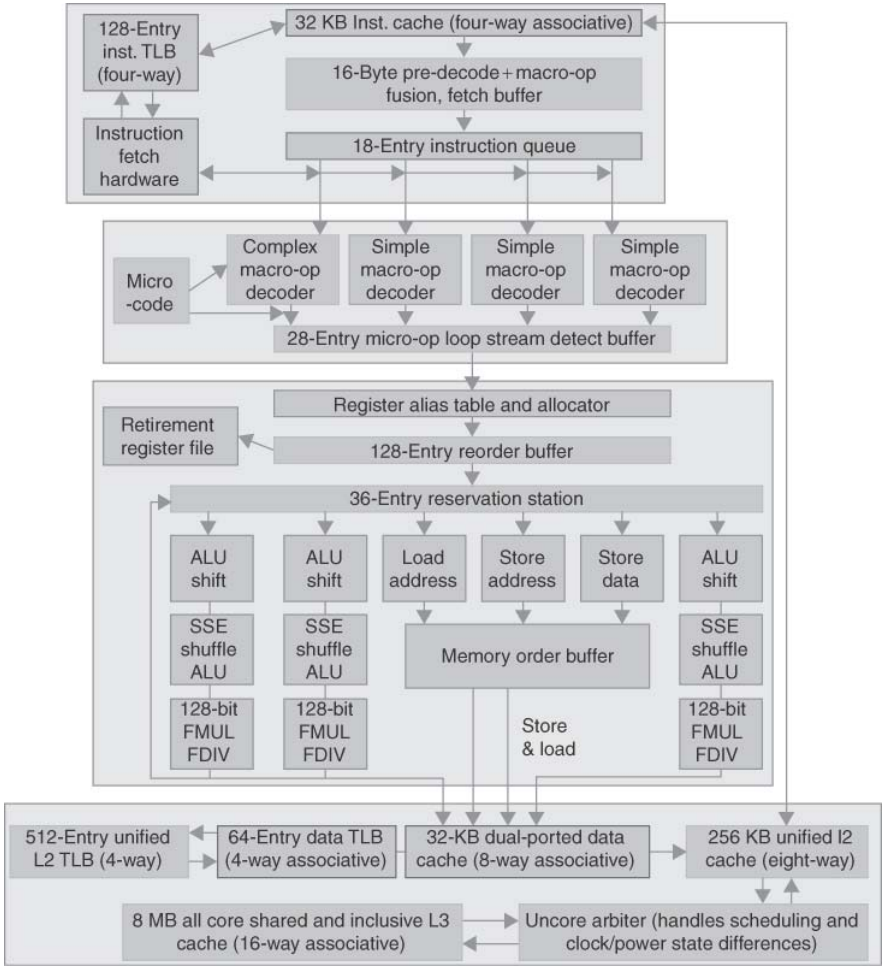

Figure 3.41 The Intel Core i7 pipeline structure shown with the memory system components. The total pipeline depth is 14 stages, with branch mispredictions costing 17 cycles. There are 48 load and 32 store buffers. The six independent functional units can each begin execution of a ready micro-op in the same cycle.

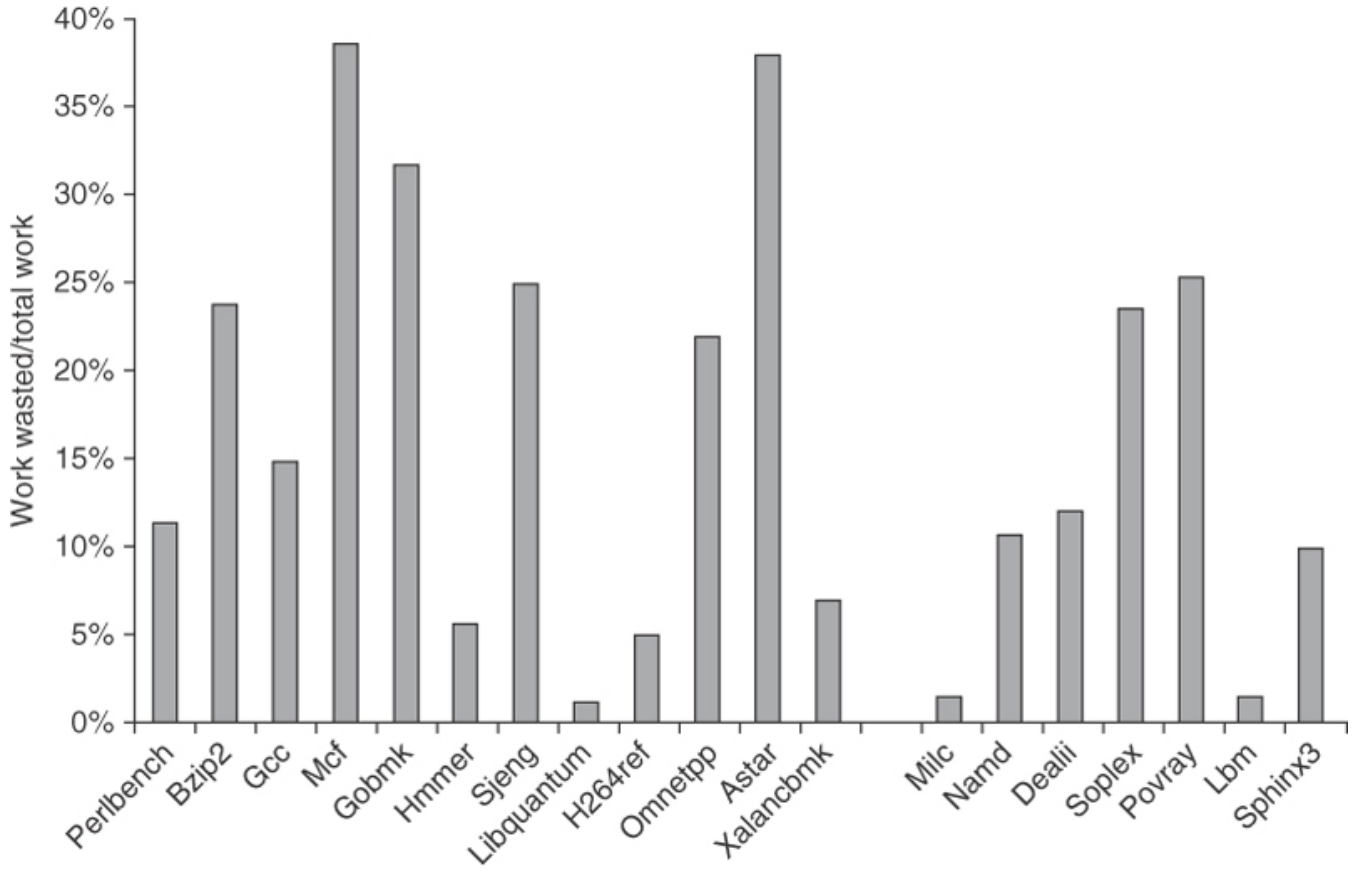

Figure 3.42 The amount of "wasted work" is plotted by taking the ratio of dispatched micro-ops that do not graduate to all dispatched micro-ops. For example, the ratio is 25% for sjeng, meaning that 25% of the dispatched and executed micro-ops are thrown away. The data in this section were collected by Professor Lu Peng and Ph.D. student Ying Zhang, both of Louisiana State University.

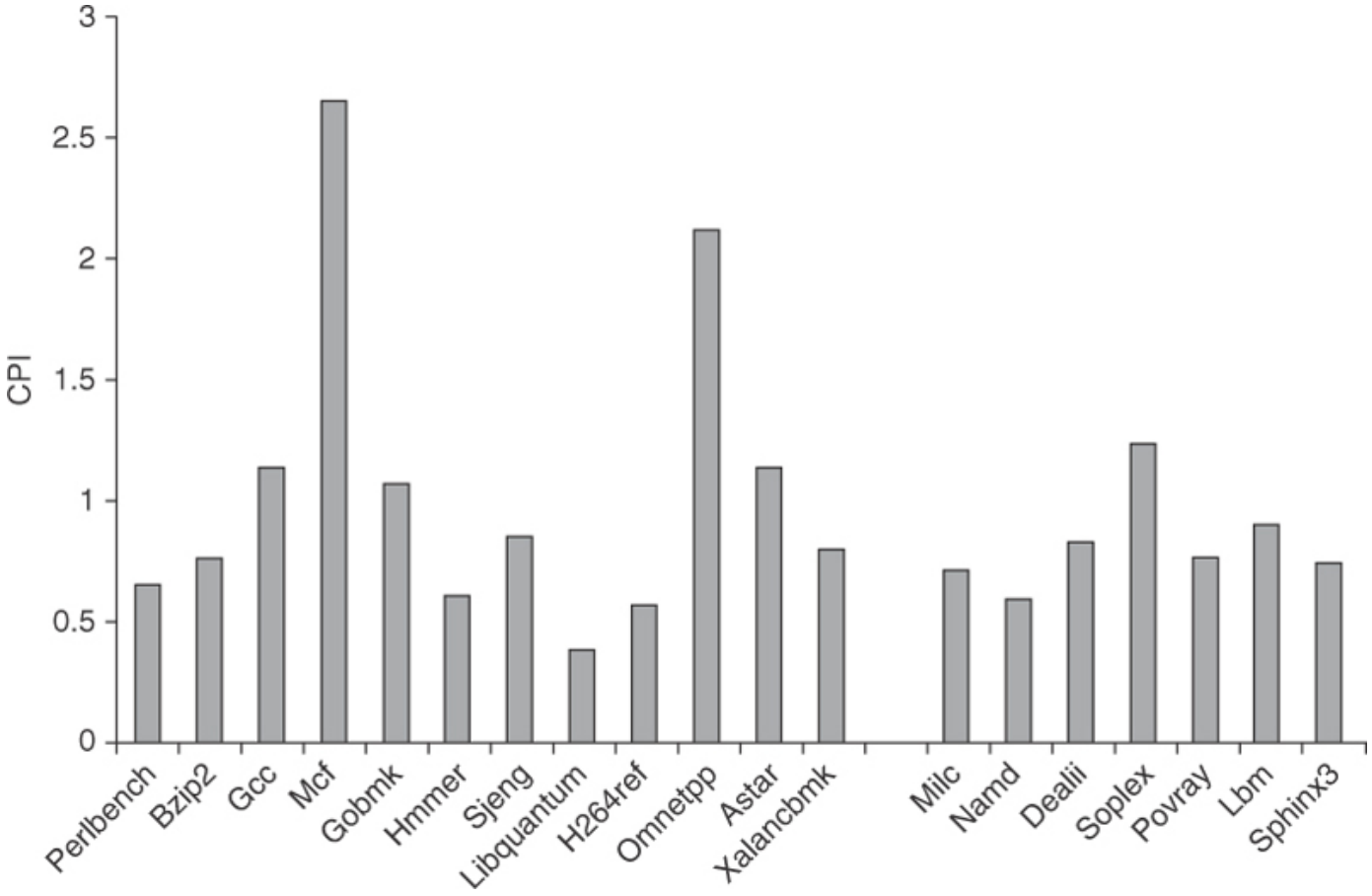

Figure 3.43 The CPI for the 19 SPECCPU2006 benchmarks shows an average CPI for 0.83 for both the FP and integer benchmarks, although the behavior is quite different. In the integer case, the CPI values range from 0.44 to 2.66 with a standard deviation of 0.77, while the variation in the FP case is from 0.62 to 1.38 with a standard deviation of 0.25. The data in this section were collected by Professor Lu Peng and Ph.D. student Ying Zhang, both of Louisiana State University.

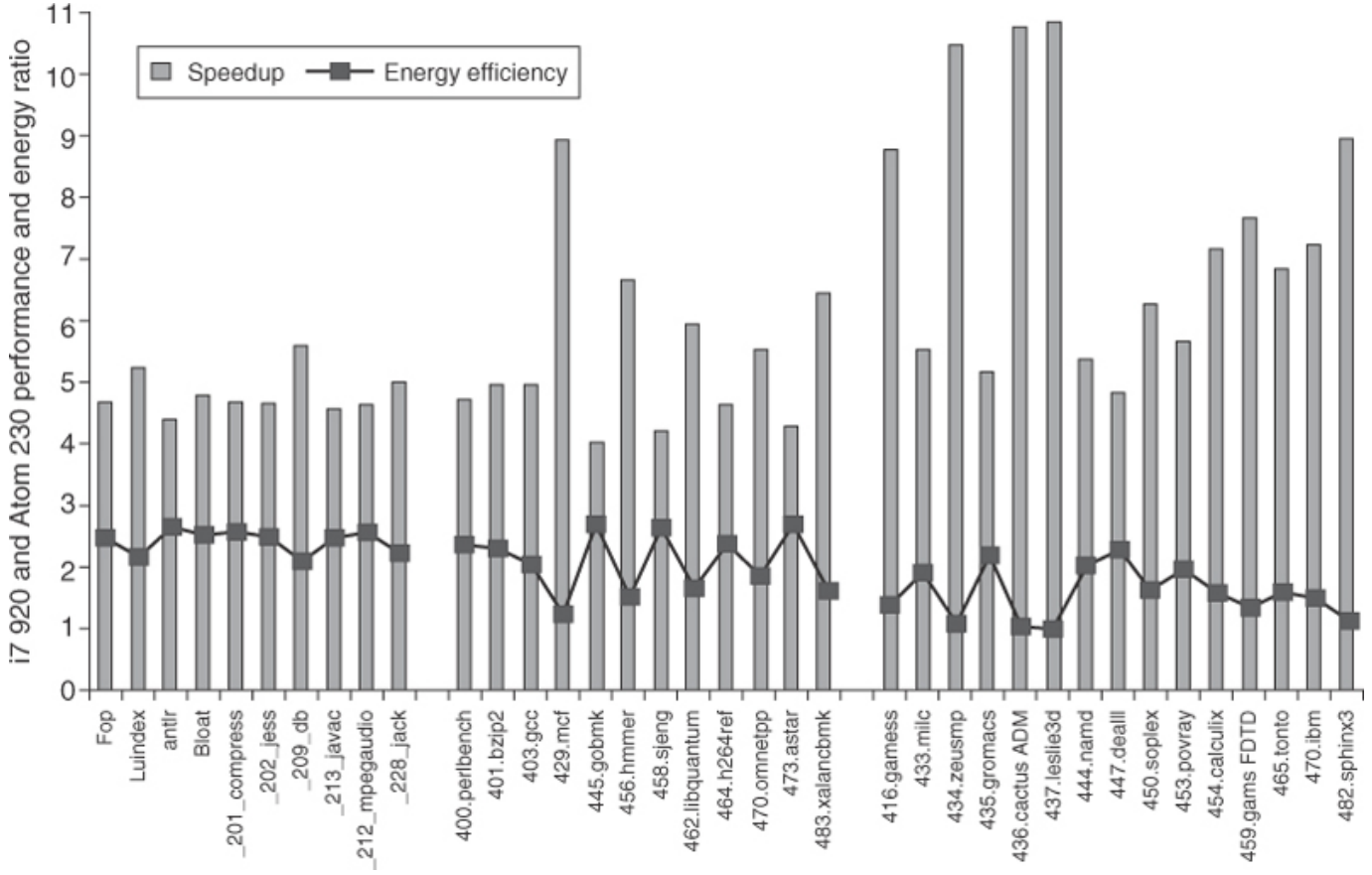

Figure 3.45 The relative performance and energy efficiency for a set of single-threaded benchmarks shows the i7 920 is 4 to over 10 times faster than the Atom 230 but that it is about 2 times *less* power efficient on average! Performance is shown in the columns as i7 relative to Atom, which is execution time (i7)/execution time (Atom). Energy is shown with the line as Energy (Atom)/Energy (i7). The i7 never beats the Atom in energy efficiency, although it is essentially as good on four benchmarks, three of which are floating point. The data shown here were collected by Esmaeilzadeh et al. [2011]. The SPEC benchmarks were compiled with optimization on using the standard Intel compiler, while the Java benchmarks use the Sun (Oracle) Hotspot Java VM. Only one core is active on the i7, and the rest are in deep power saving mode. Turbo Boost is used on the i7, which increases its performance advantage but slightly decreases its relative energy efficiency.

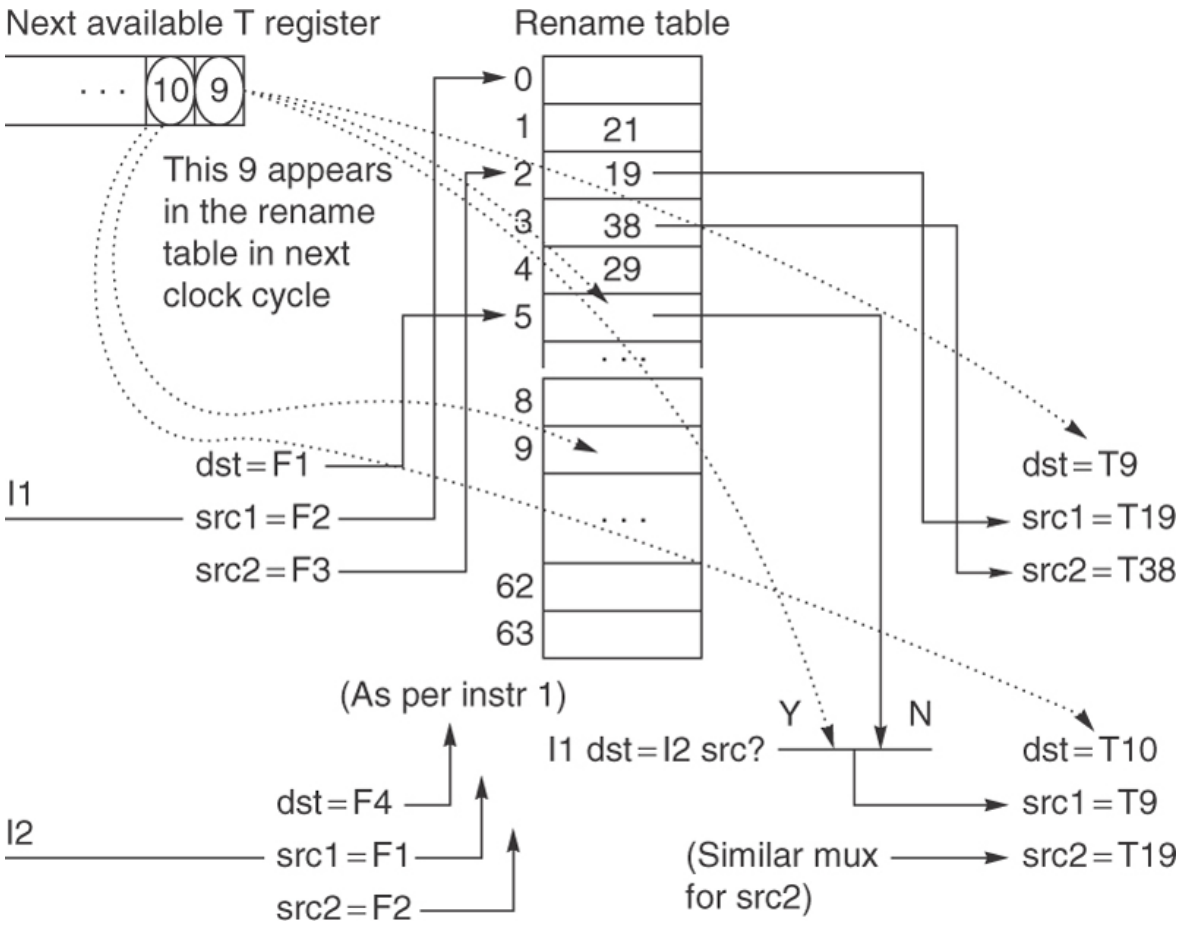

Figure 3.52 Rename table and on-the-fly register substitution logic for superscalar machines.
(Note that src is source, and dest is destination.)

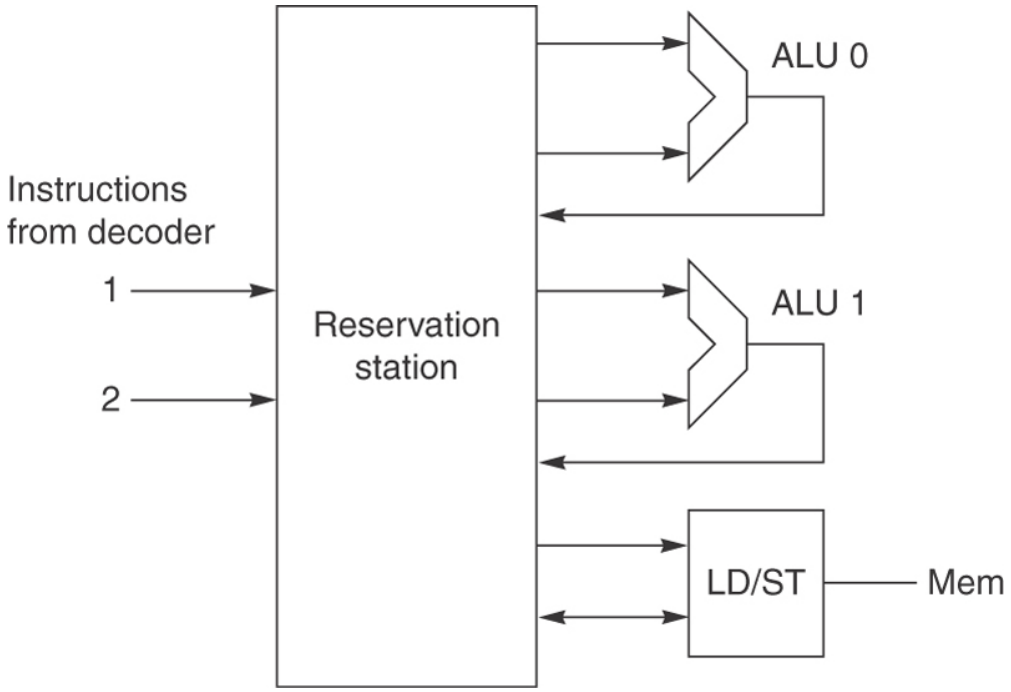

Figure 3.55 An out-of-order microarchitecure.