Microprocessor	Year	Clock Rate	Pipeline Stages	Issue Width	Out-of-Order/ Speculation	Cores/ Chip	Power	
Intel 486	1989	25 MHz	5	1	No	1	5	W
Intel Pentium	1993	66 MHz	5	2	No	1	10	W
Intel Pentium Pro	1997	200 MHz	10	3	Yes	1	29	W
Intel Pentium 4 Willamette	2001	2000 MHz	22	3	Yes	1	75	W
Intel Pentium 4 Prescott	2004	3600 MHz	31	3	Yes	1	103	W
Intel Core	2006	2930 MHz	14	4	Yes	2	75	W
Sun UltraSPARC III	2003	1950 MHz	14	4	No	1	90	W
Sun UltraSPARC T1 (Niagara)	2005	1200 MHz	6	1	No	8	70	W

FIGURE 4.73 Record of Intel and Sun Microprocessors in terms of pipeline complexity, number of cores, and power. The Pentium 4 pipeline stages do not include the commit stages. If we included them, the Pentium 4 pipelines would be even deeper. Copyright © 2009 Elsevier, Inc. All rights reserved.

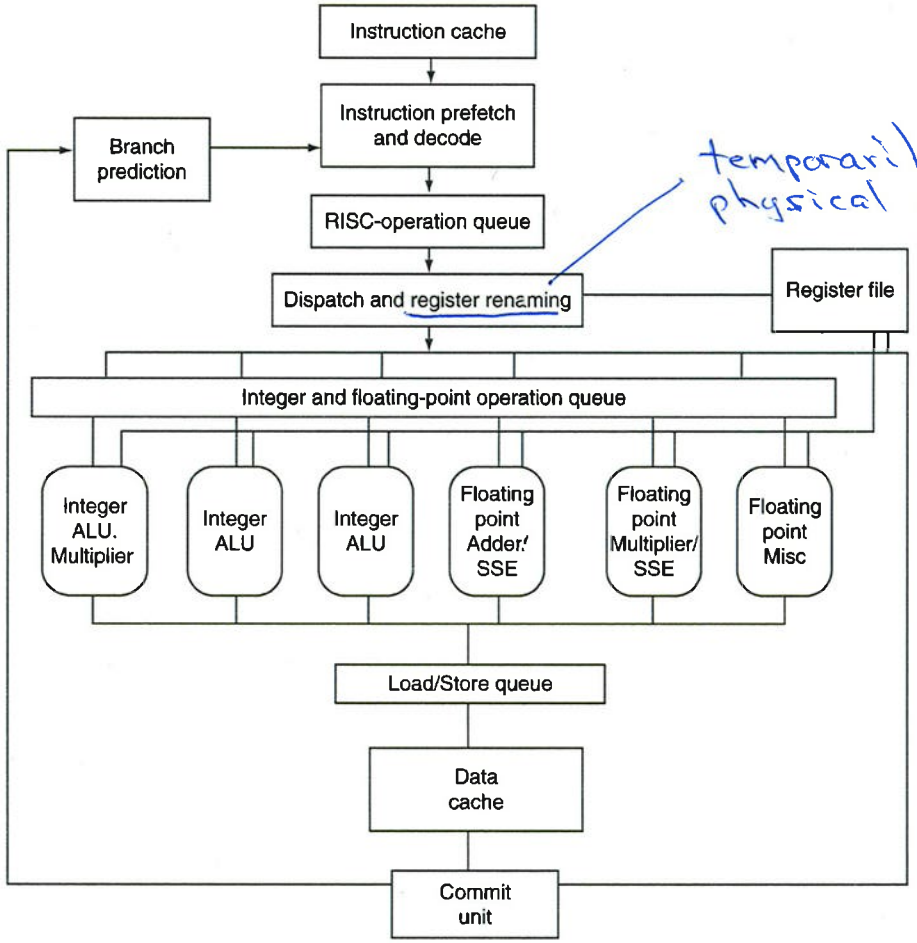


FIGURE 4.74 The microarchitecture of AMD Opteron X4. The extensive queues allow up to 106 RISC operations to be outstanding, including 24 integer operations, 36 floating point/SSE operations, and 44 loads and stores. The load and store units are actually separated into two parts, with the first part handling address calculation in the Integer ALU units and the second part responsible for the actual memory reference. There is an extensive bypass network among the functional units; since the pipeline is dynamic rather than static, bypassing is done by tagging results and tracking source operands, so as to allow a match when a result is produced for an instruction in one of the queues that needs the result.
Copyright © 2009 Elsevier, Inc. All rights reserved.

Opteron X4 pipeline

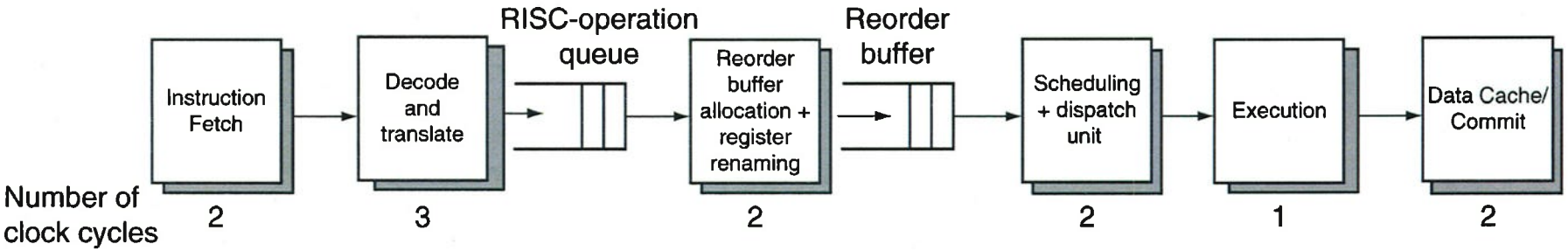


FIGURE 4.75 The Opteron X4 pipeline showing the pipeline flow for a typical instruction and the number of clock cycles for the major steps in the 12-stage pipeline for integer RISC-operations. The floating point execution queue is 17 stages long. The major buffers where RISC-operations wait are also shown. Copyright © 2009 Elsevier, Inc. All rights reserved.

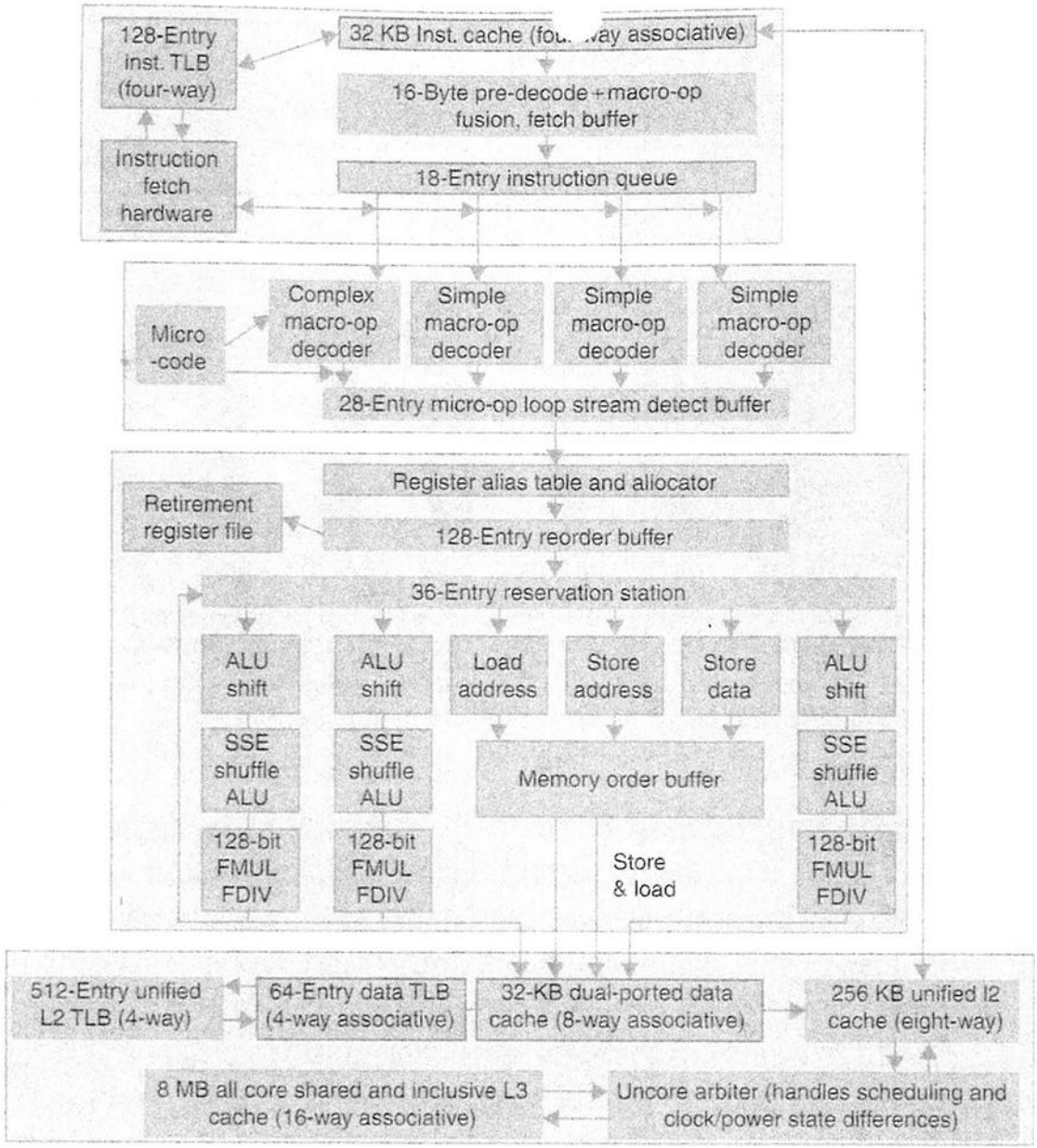


FIGURE 4.77 The Core i7 pipeline with memory components. The total pipeline depth is 14 stages, with branch mispredictions costing 17 clock cycles. This design can buffer 48 loads and 32 stores. The six independent units can begin execution of a ready RISC operation each clock cycle.

Processor	ARM A8	Intel Core i7 920
Market	Personal Mobile Device	Server, Cloud
Thermal design power	2 Watts	130 Watts
Clock rate	1 GHz	2.66 GHz
Cores/Chip	1	4
Floating point?	No	Yes
Multiple Issue?	Dynamic	Dynamic
Peak instructions/clock cycle	2	4
Pipeline Stages	14	14
Pipeline schedule	Static In-order	Dynamic Out-of-order with Speculation
Branch prediction	2-level	2-level
1st level caches / core	32 KiB I, 32 KiB D	32 KiB I, 32 KiB D
2nd level cache / core	128 - 1024 KiB	256 KiB
3rd level cache (shared)	–	2 - 8 MiB

FIGURE 4.74 Specification of the ARM Cortex-A8 and the Intel Core i7 920.

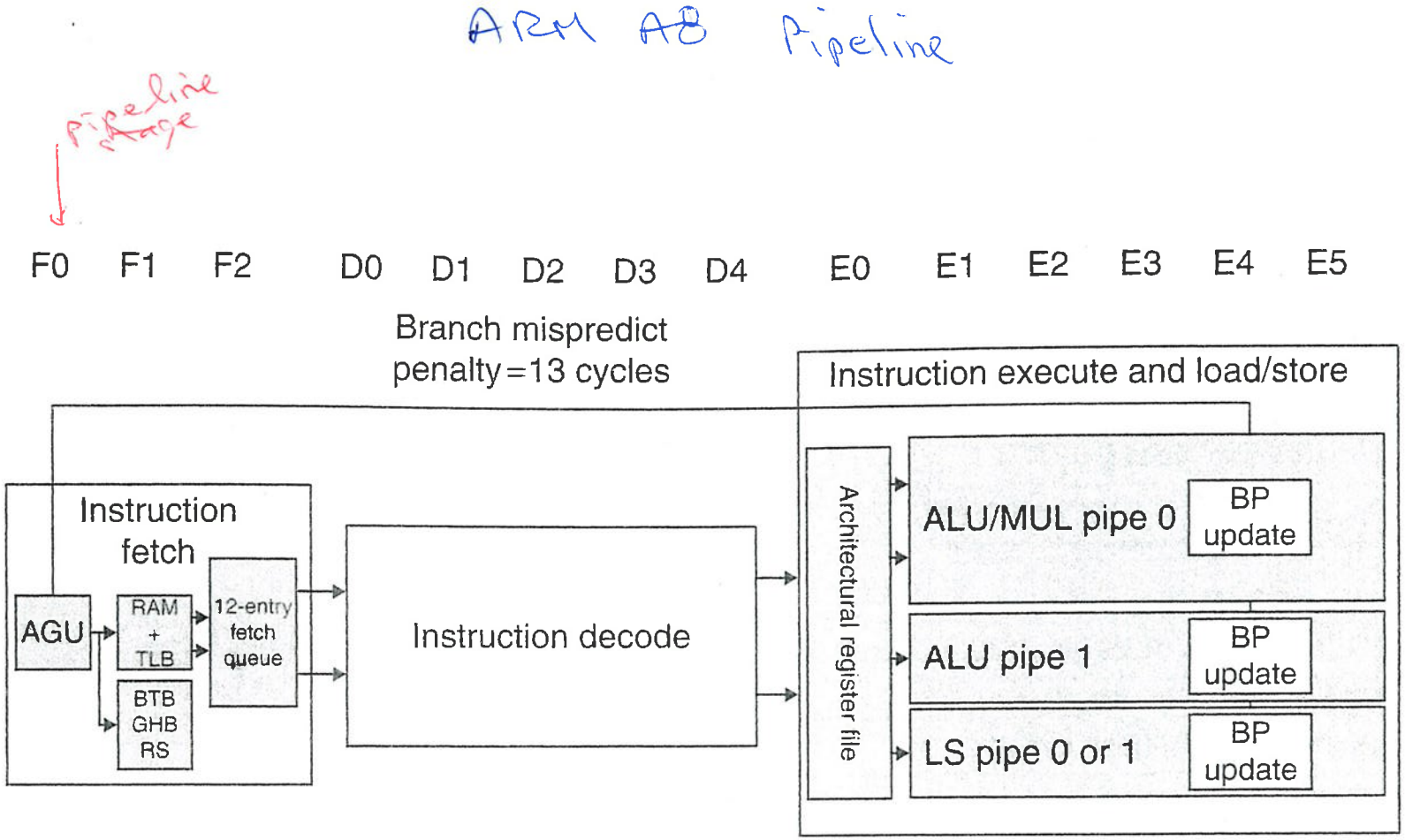
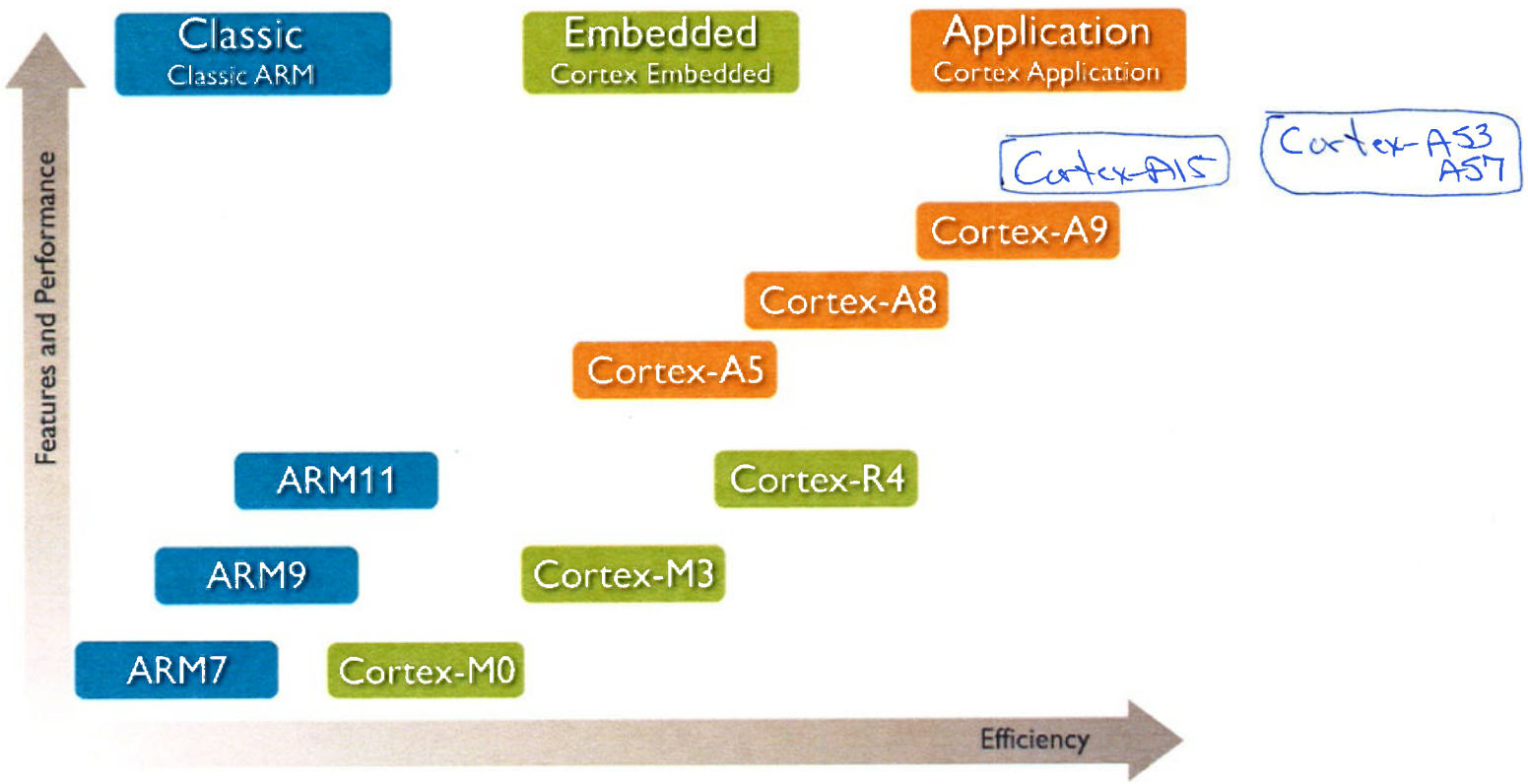
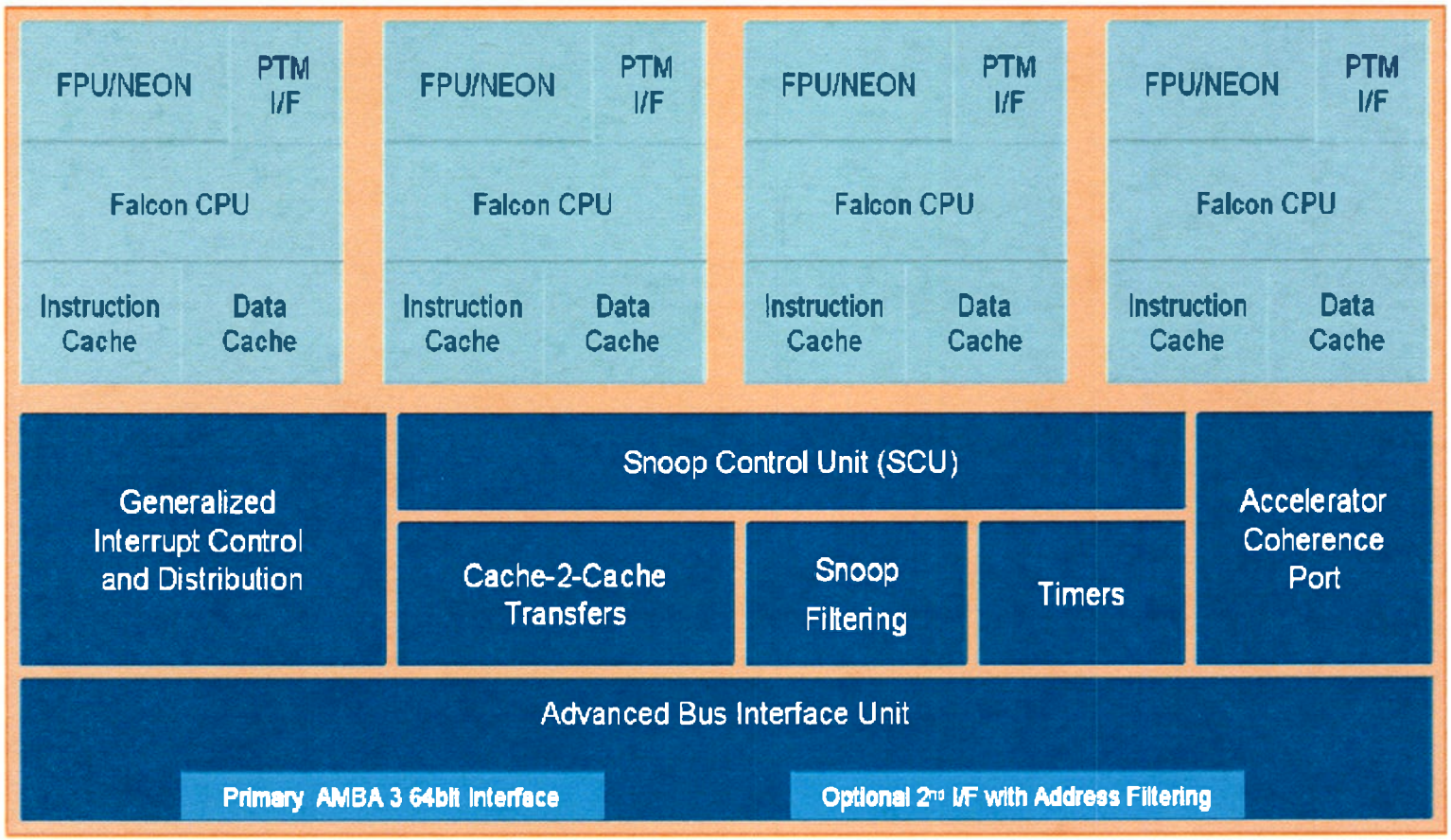


FIGURE 4.75 The A8 pipeline. The first three stages fetch instructions into a 12-entry instruction fetch buffer. The *Address Generation Unit* (AGU) uses a *Branch Target Buffer* (BTB), *Global History Buffer* (GHB), and a *Return Stack* (RS) to predict branches to try to keep the fetch queue full. Instruction decode is five stages and instruction execution is six stages.

ARM Instruction Set Architectures.



ARM Cortex-A9 MPcore (Multicore)



ARM Cortex-A9 Microarchitecture

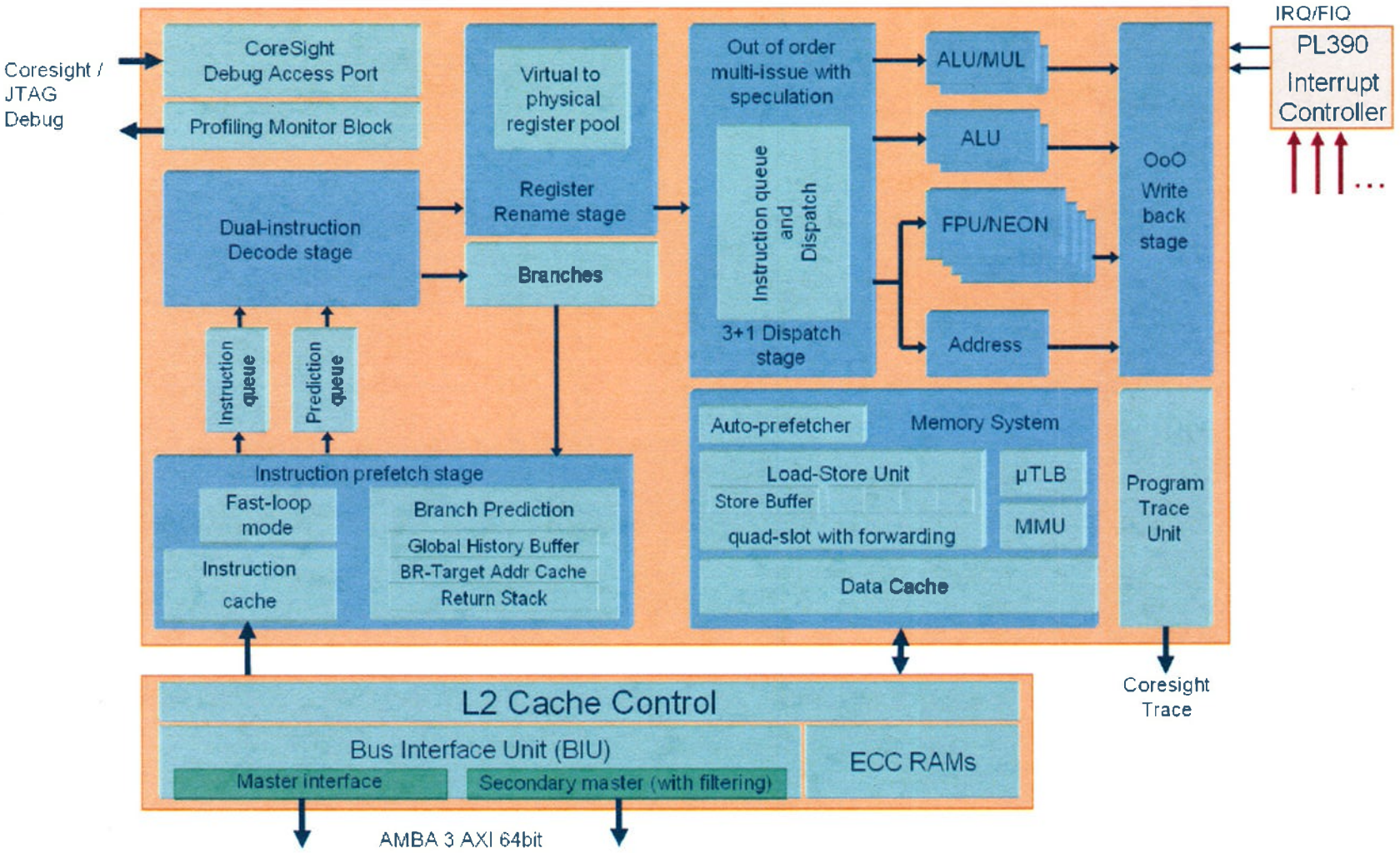


Fig. 1 Cortex-A9 microarchitecture structure and the single core interfaces.

Pipeline description

- Advanced processing of instruction fetch and branch prediction - unblocks branch resolution from potential memory latency-induced instruction stalls.
- Up to four instruction cache line prefetch-pending - further reduces the impact of memory latency so as to maintain instruction delivery.
- Between two and four instructions per cycle forwarded continuously into instruction decode -ensures efficient superscalar pipeline utilization.
- Fast-loop mode - provides low power operation while executing small loops.
- Superscalar decoder - capable of decoding two full instructions per cycle.
- Speculative execution of instructions - enabled by dynamic renaming of physical registers into an available pool of virtual registers.
- Increased pipeline utilization - removing data dependencies between adjacent instructions and reducing interrupt latency.
- Virtual renaming of registers - accelerating code through an effective hardware based unrolling of loops without the additional costs in code size and power consumption.
- Any of the four subsequent pipelines can select instructions from the issue queue - providing out of order dispatch further increasing pipeline utilization independent of developer or compiler instruction scheduling. Ensures maximum performance from code optimized for previous generation of processors and maintains existing software investments.
- Concurrent execution across full dual arithmetic pipelines, load-store or compute engine, plus resolution of any branch each cycle.
- Dependent load-store instructions can be forwarded for resolution within the memory system - further reduces pipeline stalls and significantly accelerating the execution of high level code accessing complex data structures or invoking C++ functions.
- Support for four data cache line fill requests - further reduces stalls due to memory latency with either automatic or user driven prefetching to ensure the availability of critical data.
- Out of order write back of instructions - enables the pipeline resources to be released independent of the order in which the system provides the required data.