Lecture 26: Parallel Processing

Spring 2024
Jason Tang

Topics

- Static multiple issue pipelines
- Dynamic multiple issue pipelines
- Hardware multithreading

Taxonomy of Parallel Architectures

- Flynn categories: SISD, MISD, SIMD, and MIMD
- Levels of Parallelism:
    - Bit level parallelism: 4-bit, 8-bit, 16-bit, 32-bit, now 64-bit processors
    - Instruction level parallelism (ILP): Pipelining, VLIW, superscalar, out of order execution
    - Process/thread level parallelism (TLP): Running different programs, or different parts of same program, on one processor

Multiple Issue Pipelining

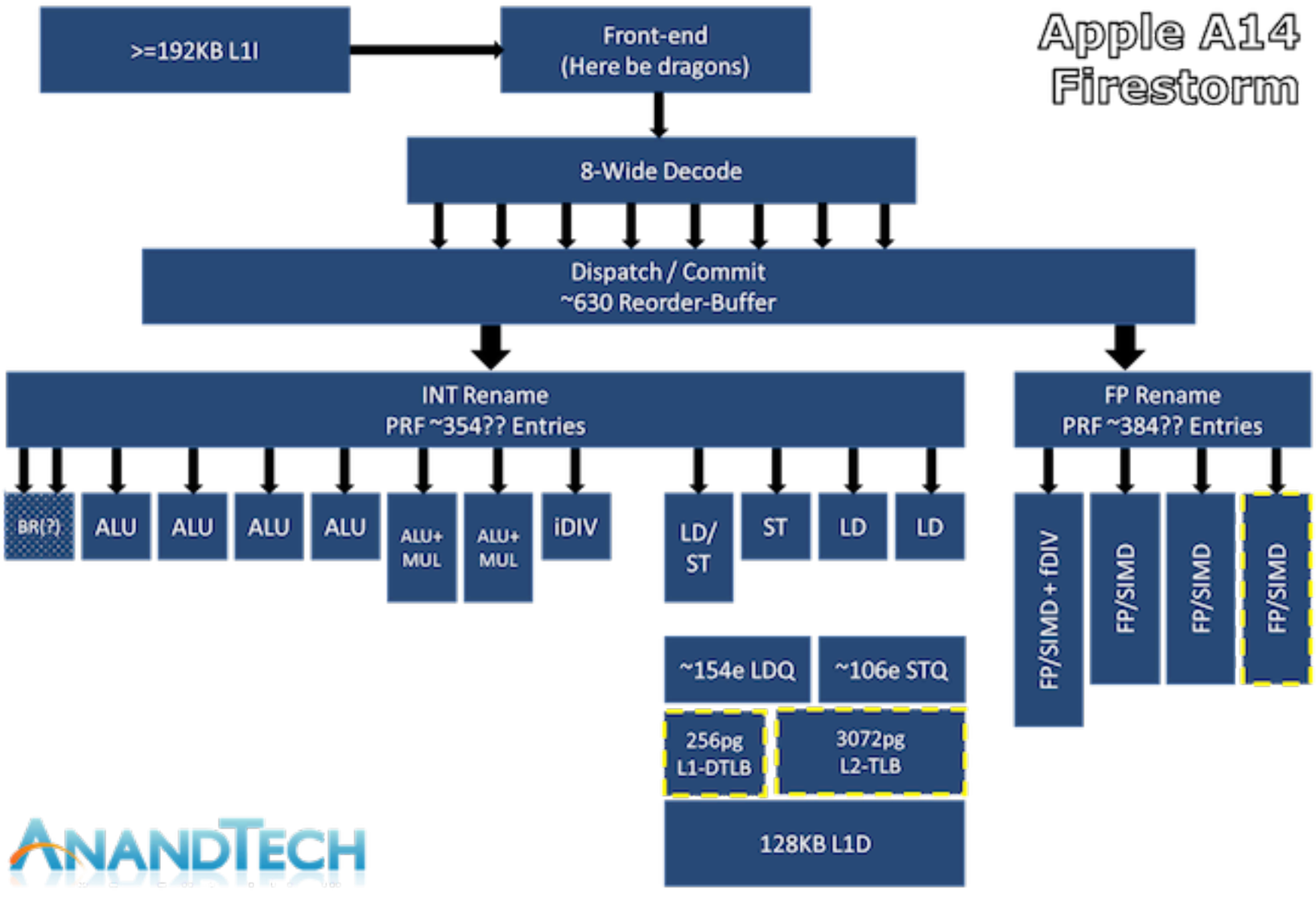
- In classic pipelining, processor fetches *one* instruction per cycle
- In multiple issue pipelining, processor fetches *two* (or more) instructions
  - Therefore, its CPI is *less* than 1.0
- Two ways to implement multiple issue pipelines:
  - Static multiple issue: software writer and compiler decide instruction issues, before execution
  - Dynamic multiple issue: hardware decides issues, during execution

Issue Slots

- Portion within a processor from which an instruction can execute
- In simple processors, there is exactly one issue slot, which can perform any kind of instruction (integer arithmetic, floating point arithmetic, branching, etc)
- Modern processors have multiple issue slots, but not all slots are equal

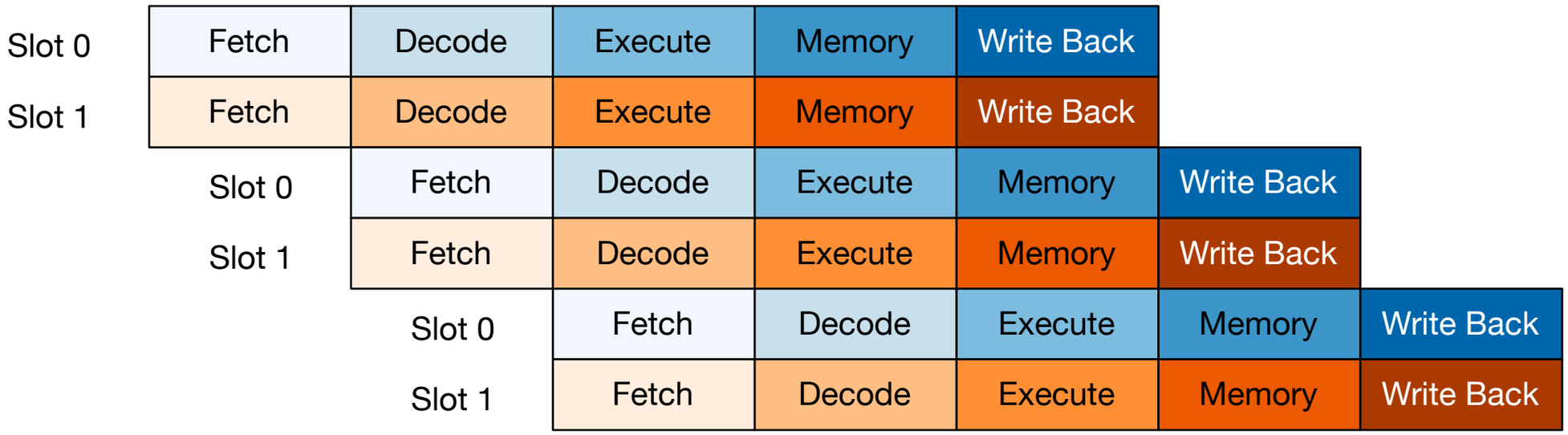
Issue Slots

- Example: Apple's A14 processor has different issue slots, for different purposes



https://www.anandtech.com/show/16226/apple-silicon-m1-a14-deep-dive/2

Static Multiple Issue Pipeline

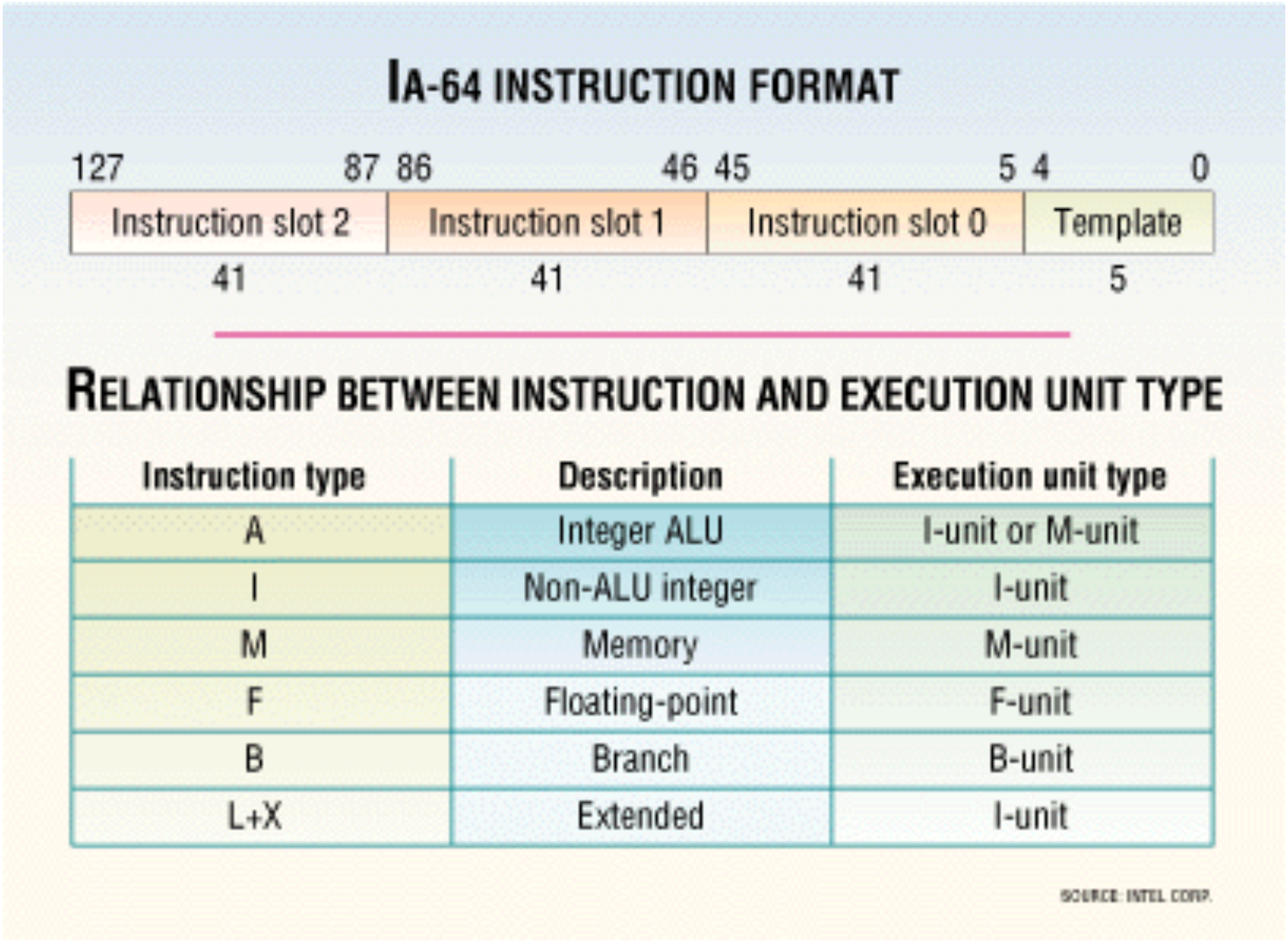


- Multiple instructions execute simultaneously, in different slots
- Issue packet: set of instructions executed together
- Compiler is responsible for generating contents of issue packet
  - Compiler is responsible for preventing data and control hazards

Very Long Instruction Word (VLIW)

- Compiler generates a single “instruction” that commands each issue slot
    - If the instruction cannot be parallelized or if a data/control hazard is to be resolved via a bubble, then place a no-op for that issue slot
- Example: Itanium was Intel’s first attempt at a 64-bit processor
    - **Not** in any way related to x86-64 (more properly “*AMD64*” or “*Intel 64*”)
    - EPIC: Explicitly Parallel Instruction Computing, term invented by Intel for its VLIW implementation
    - Each IA-64 issue packet is 128-bits, containing three instructions within

IA-64 Instruction Format



IA-64 INSTRUCTION FORMAT

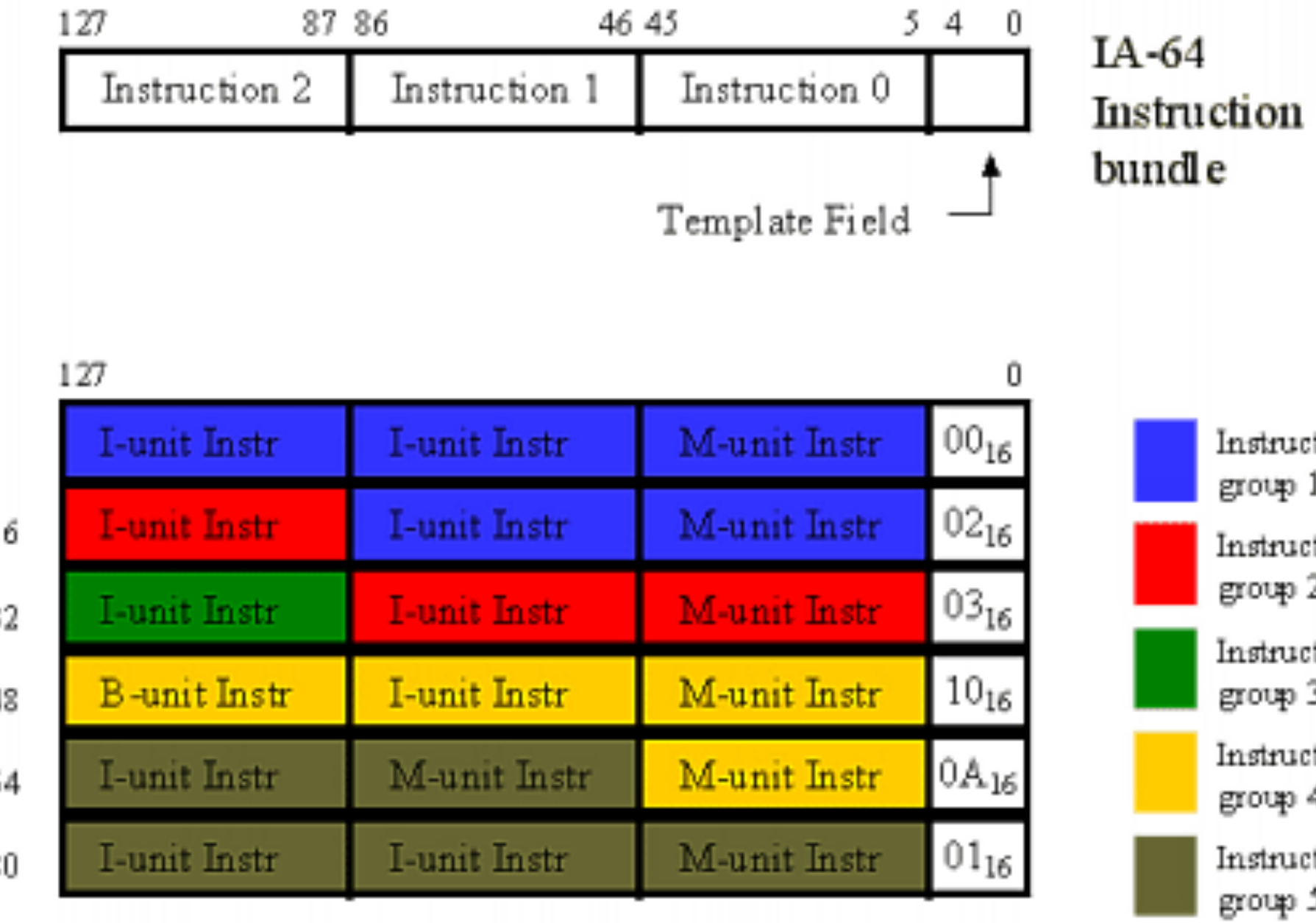
RELATIONSHIP BETWEEN INSTRUCTION AND EXECUTION UNIT TYPE

Instruction type	Description	Execution unit type
A	Integer ALU	I-unit or M-unit
I	Non-ALU integer	I-unit
M	Memory	M-unit
F	Floating-point	F-unit
B	Branch	B-unit
L+X	Extended	I-unit

SOURCE: INTEL CORP.

https://www.eetimes.com/document.asp?doc_id=1139305

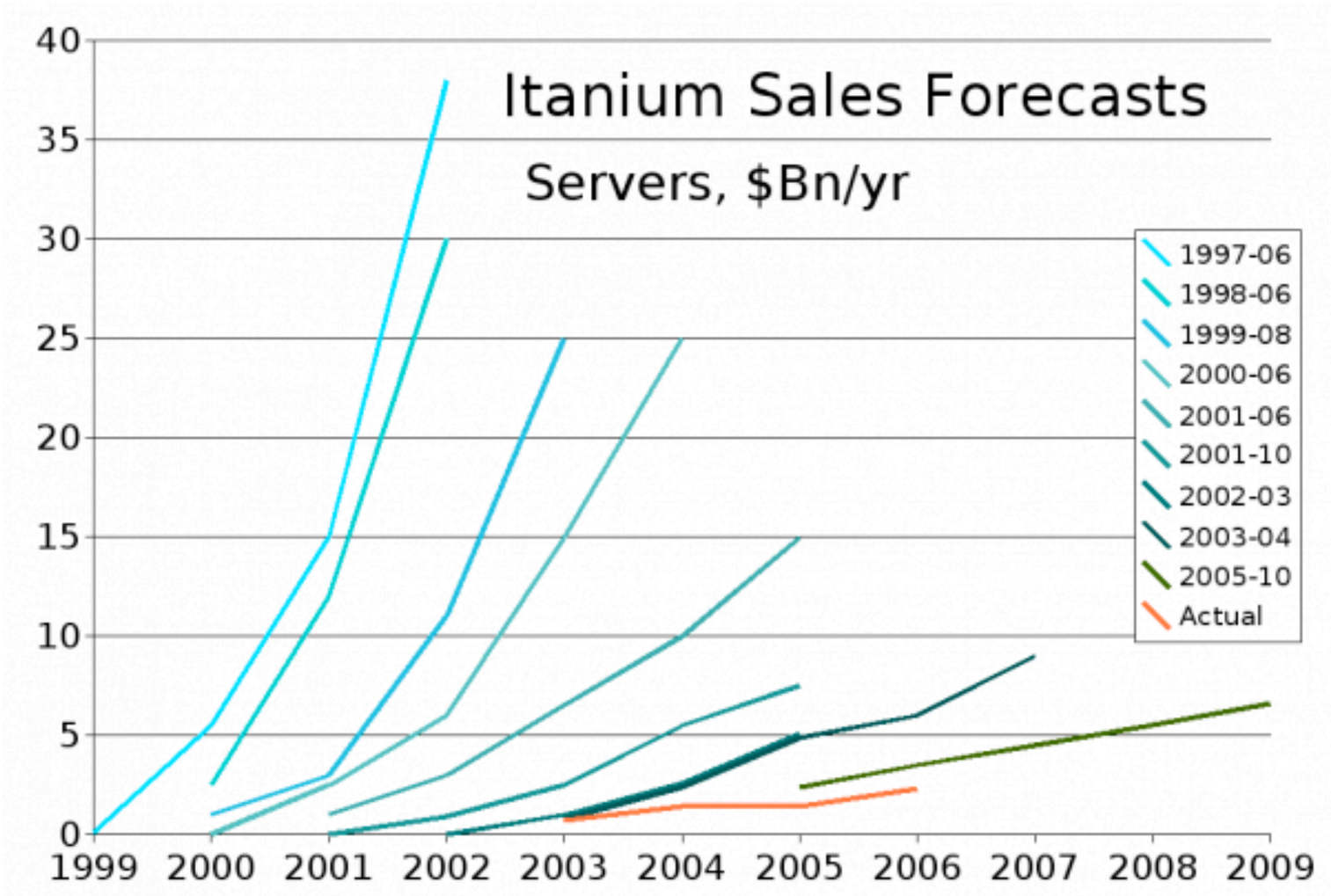
IA-64 Example Instruction Flow



https://www.realworldtech.com/intel-history-lesson/4/

End of the “Itanic”

- Effectively discontinued within ten years of release

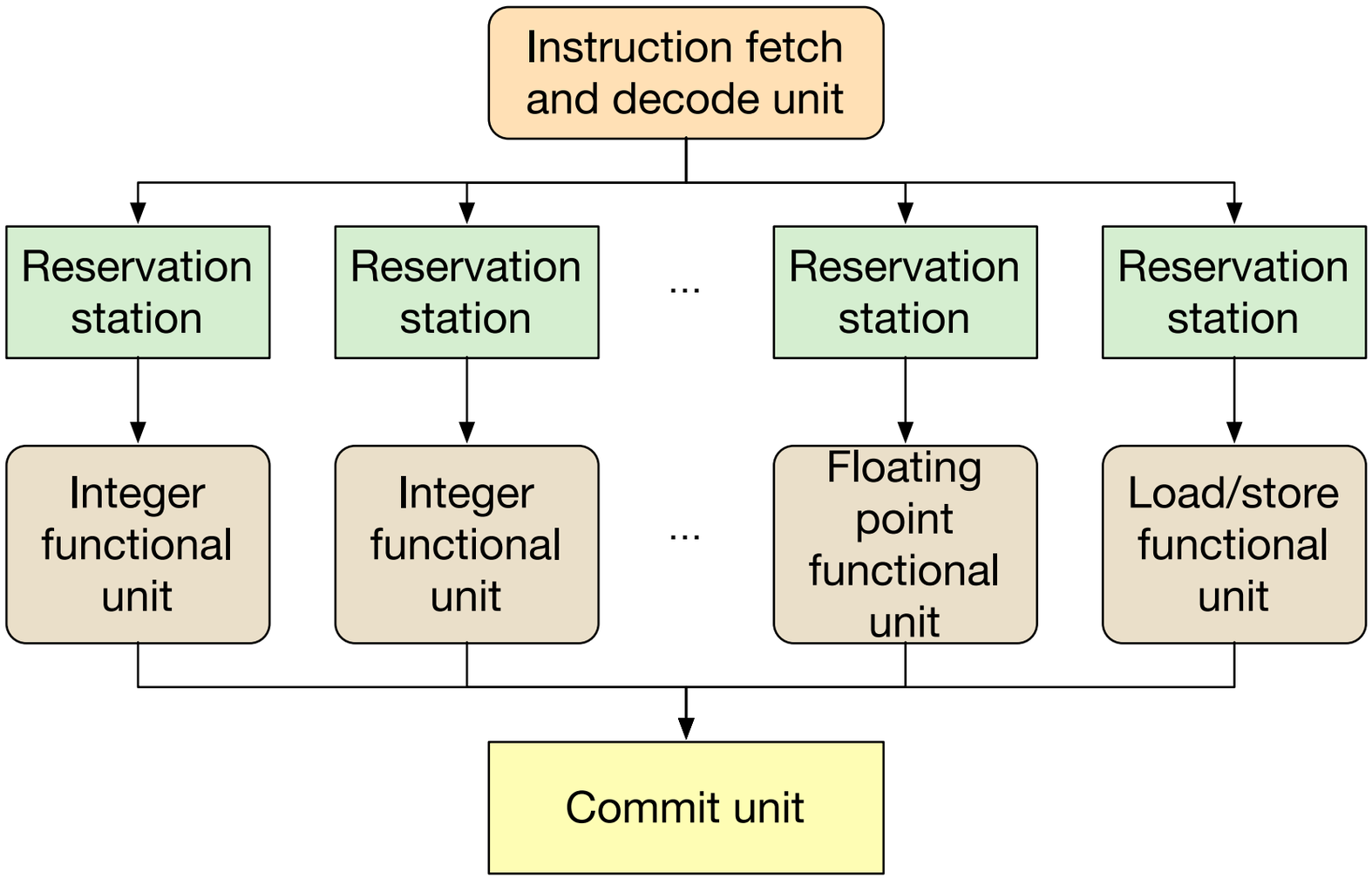


https://helgeklein.com/blog/itanium-ia-64-an-obituary/

Dynamic Multiple Issue Pipeline

- Superscalar processor fetches multiple instructions at once
- Dynamic pipeline schedule: Processor decides which instruction(s) to execute each cycle, to avoid hazards
- Out of order execution: Processor may execute instructions in a different order than given by compiler, to minimize stalls
- Requires duplicate and much more complex hardware

Dynamic Pipeline Scheduling



- Instructions are fetched in-order, and later committed ("retired") in-order
- Functional units execute instructions out-of-order

Out of Order Execution

- With standard pipelining, if an instruction stalls, all instructions after it are also stalled

- With dynamic scheduling, split *Decode* into substages:
  - Issue: decode instruction, checking for structural hazards
  - Dispatch: wait until all data hazards have been resolved, then fetch operands; hold instruction until a functional unit can execute it

- Commit unit is responsible for reordering functional units' results to achieve in-order results

Dynamic Scheduling Example

- For a modern Intel Skylake processor, 64-bit integer division can take up to 90 clock cycles, while integer addition takes just one cycle

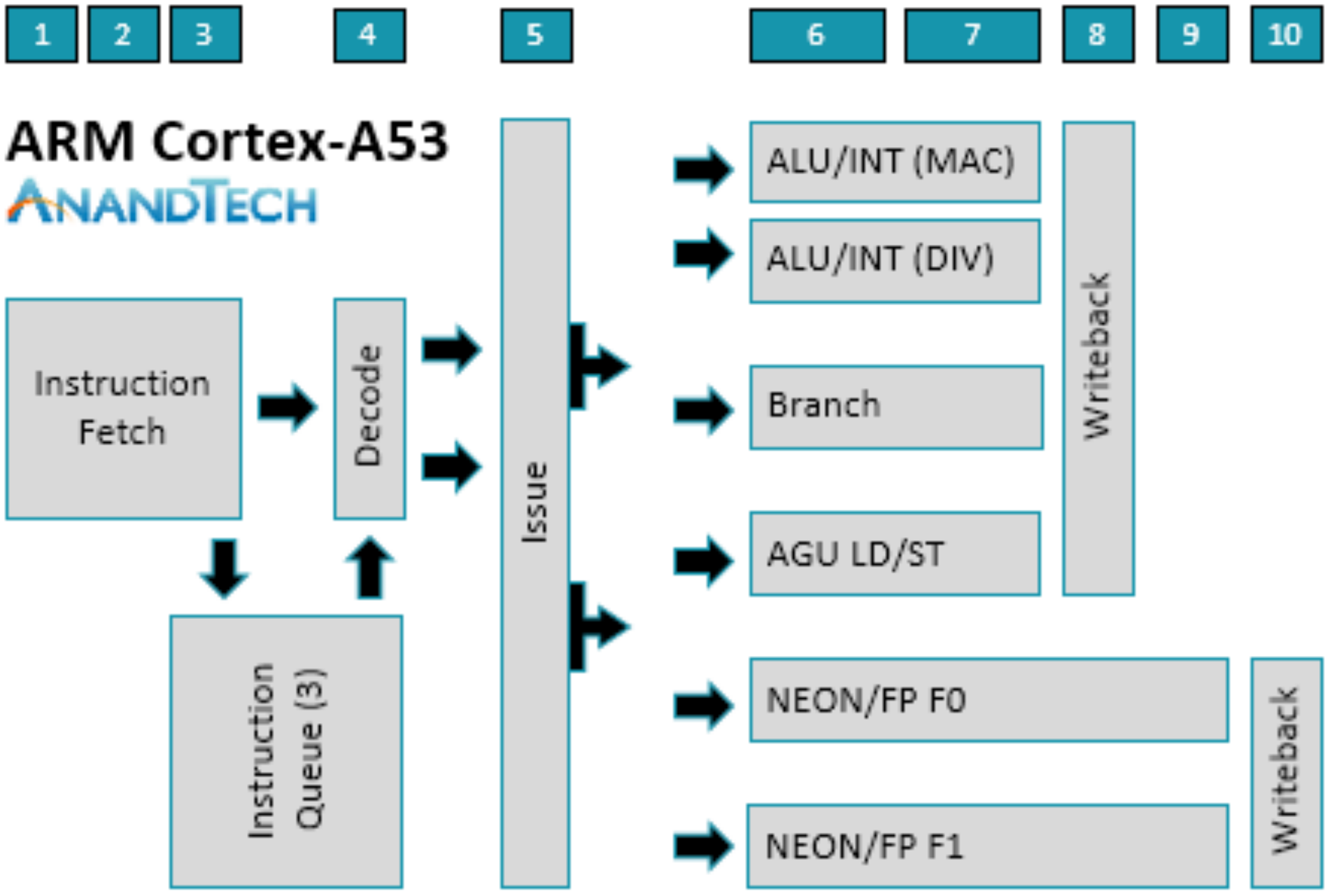
- With a single pipelined processor, a division followed by 8 additions would take 98 cycles

- With a single superscalar processor with 4 ALUs and a single FPU, and where there are no data dependencies between instructions, that same sequence would take 92 cycles

- If that superscalar processor also executes out of order, that same sequence would take 90 cycles

http://www.agner.org/optimize/instruction_tables.pdf

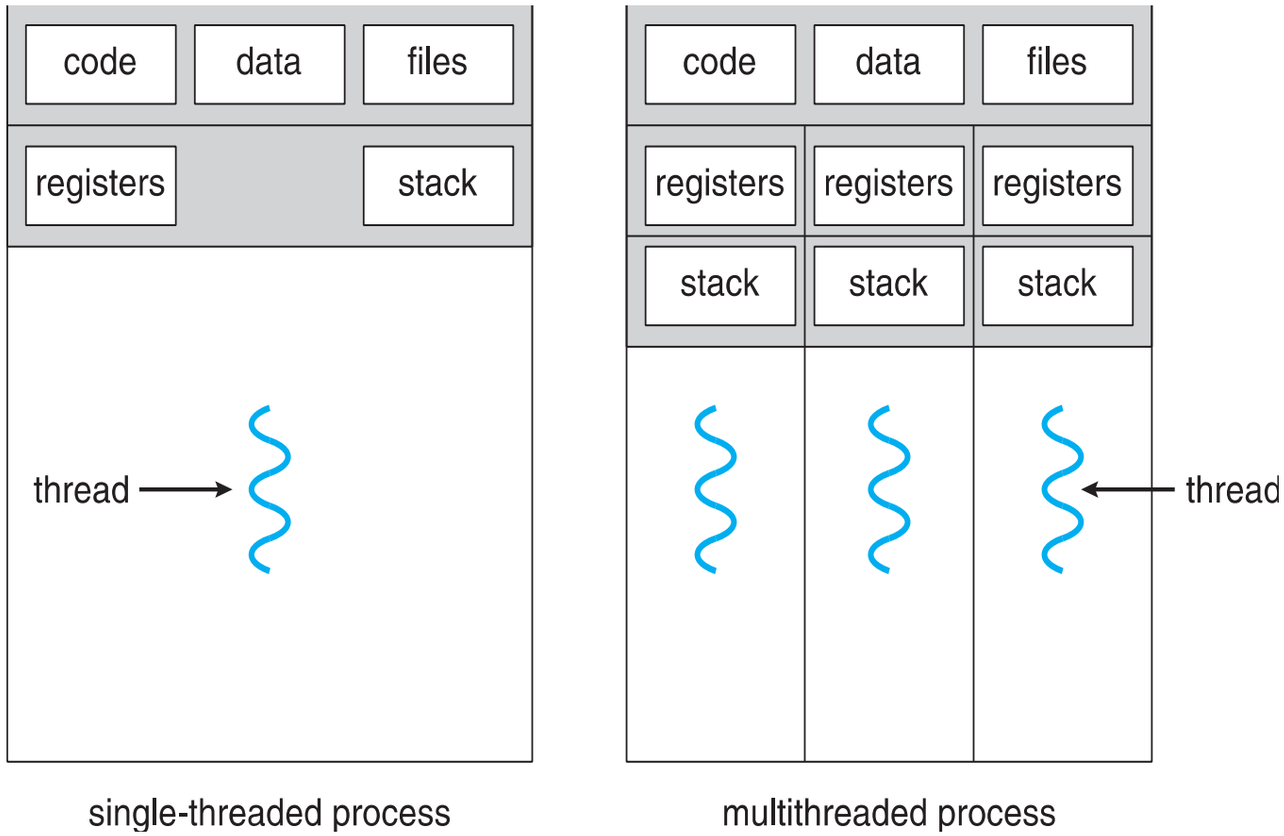
Cortex-A53 Pipeline Stages



https://www.anandtech.com/show/8542/cortexm7-launches-embedded-iot-and-wearables/2

Threading

- A software multithreaded system has a single program, but with multiple program counters and stacks
  - Operating system decides which thread to run on which processor, based upon the process scheduler



  - A processor normally executes only one thread at a time

- A hardware multithreaded system has a processor that can truly execute multiple threads simultaneously, via dynamic scheduling

Hardware Multithreading

- Processor fetches from 2 (or more) different program counters, where each program counter corresponds to a thread

- Superthreading: processor executes instruction(s) from one thread on one clock cycle, then instruction(s) from second thread on next clock cycle
  - Number of instructions to execute is based upon available issue slots
  - Instead of the operating system choosing how long to run each thread, the hardware switches threads after every clock cycle

- Simultaneous Multithreading (SMT): processor fills issue slots with mix of instructions from all threads, to keep all functional units busy

https://arstechnica.com/features/2002/10/hyperthreading/

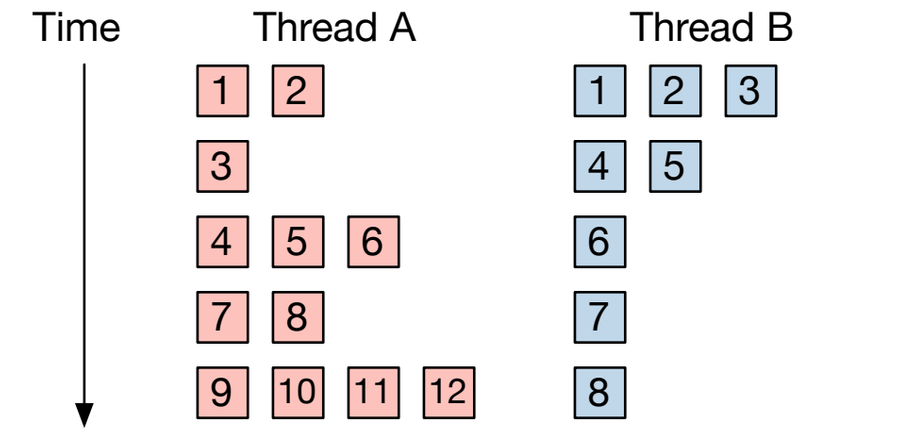
Hardware Multithreading

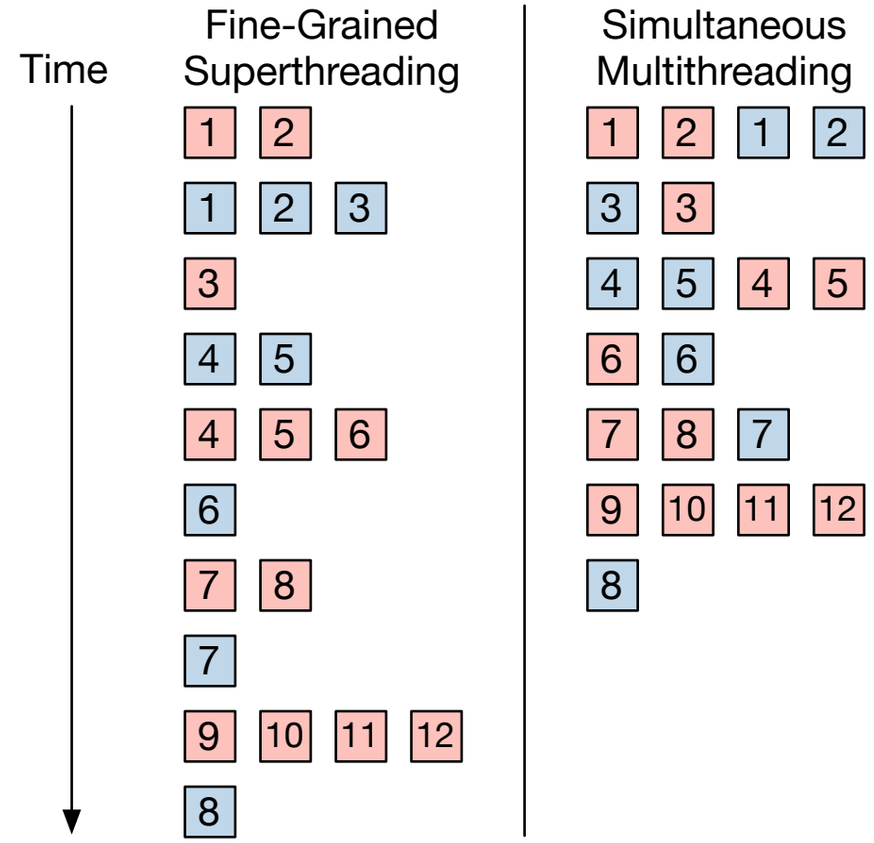
- Example: processor has 4 issue slots, and is to execute 2 threads

- Suppose that threads normally execute on a superscalar processor as per top right

- With a fine-grained superthreading (switch thread each clock cycle) and a SMT processor, execution would instead look like lower left

  - With course-grained superthreading, processor switches threads only on expensive stalls, instead of every cycle





Comparison of Hardware Multithreading

	Superthreading	Symmetric Multithreading
Number of different threads in a pipeline stage in a given cycle	Exactly 1	1+
Scheduling algorithm	Round-robin, or processor may have priority scheduling	Simultaneous
Processor usage	Same as superscalar	Fill every available issue slots
Stall penalty	Partially hidden, because other threads will execute during stall	Completely hidden, because other threads will fill issue slots
Power consumption	Same as superscalar	Uses more power