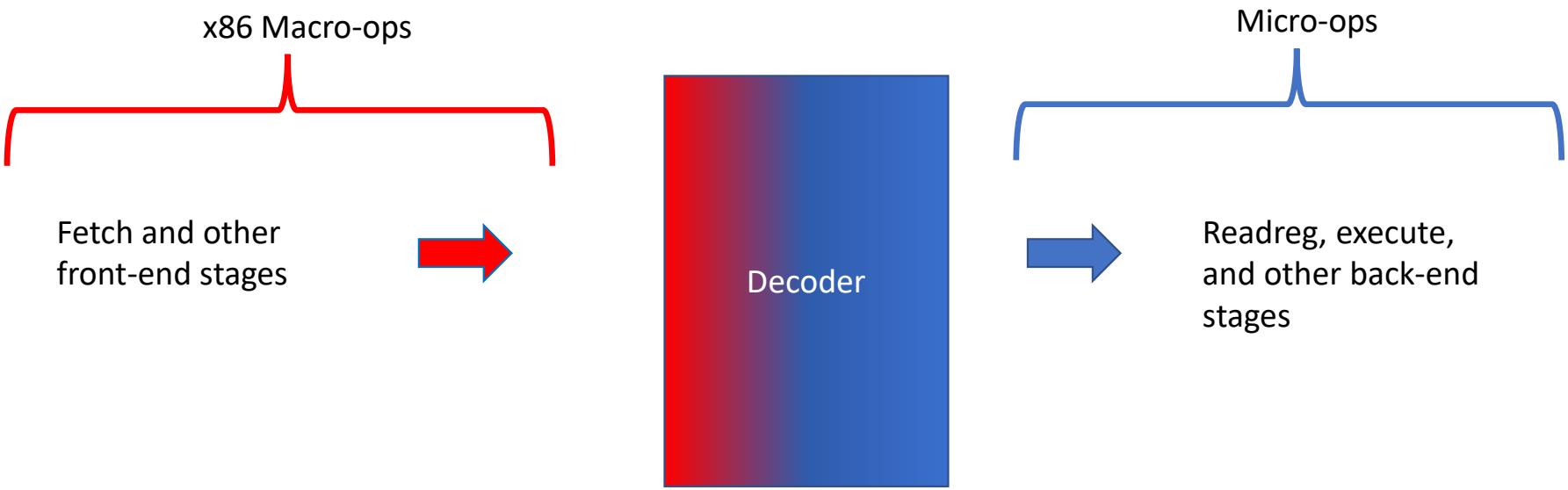
A 4-slide introduction to Intel micro-ops (uops).

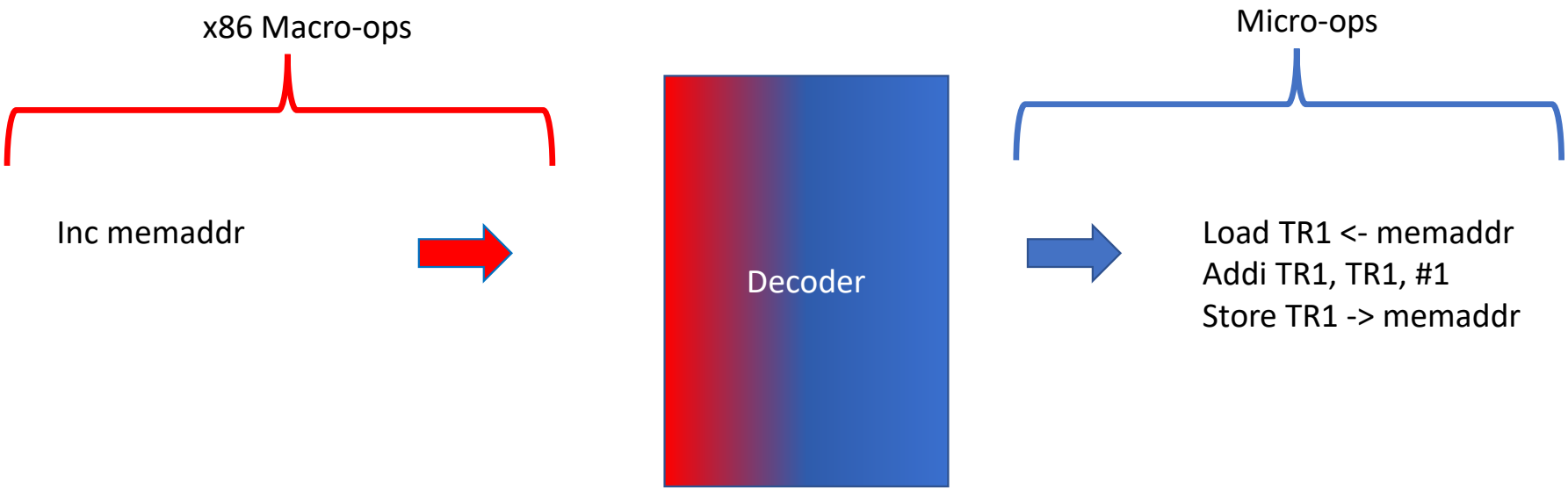
- A modern Intel processor maintains two distinct Instruction Set Architectures:
  - The external x86 ISA that has been around since the 1970s. That ISA is:
    - Publicly defined
    - User-, programmer-, compiler-visible
    - Changes rarely, to maintain binary compatibility with their code base
    - CISC, and *very* poorly structured for pipelining
  - An internal ISA that actually executes on the internal pipeline. That ISA is:
    - Secret and proprietary
    - Programmer-invisible
    - Can be changed each generation with no impact on binary compatibility
    - RISC, and well suited for pipelining

The translation between the x86 ISA instructions (sometimes called macro-ops) and micro-ops

- Is done completely in hardware
- Is done in a pipeline stage called the “decoder”
  - Note that this is different than the ID stage we know, which translates opcodes to control signals. They must also do this for micro-ops, but later in the pipeline
- Many macro-ops can each be translated to a single micro-op, but others translate to 2, 3, 4 micro-ops. More complex macro-ops require a more expensive (ie, slow) translation, sometimes requiring control flow and loops to complete a single macro-op (e.g., Intel’s REP LODS instruction on a null-terminated string, which executes until it discovers the end of the string)


x86 Macro-ops
Fetch and other front-end stages
Decoder
Micro-ops
Readreg, execute, and other back-end stages

For example…



Here, a single macro-op (increment memory) translates to 3 micro-ops, which use an internal register to temporarily hold the value, and each go through the pipeline independently.