The Microarchitecture of Superscalar Processors

Outline

- Requirements
- Features of the superscalar
- Superscalar design
- Hardware solutions
- Example processors
- Discussion

Requirements

- Fetch and decode several instructions at a time into multiple pipelines
- Binary backward compatible
  - Sequential execution model (appearance)
  - Precise states

Features

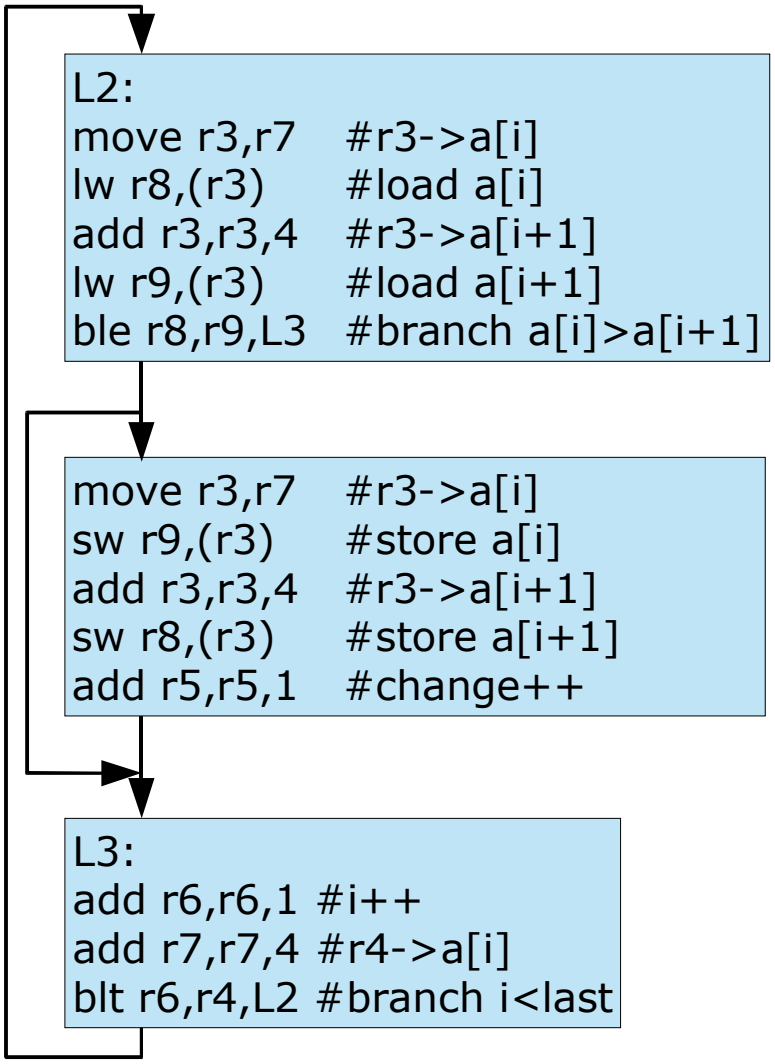
- Superscalar design does:
  - Fetch multiple instructions simultaneously
  - Find dependences and forward data to dependent instructions
  - Issue instructions in parallel
  - Serve multiple memory refs. per cycle
  - Handle unpredictable memory performance
  - Reorder writebacks to appear as executing sequentially

Data Dependence

- True dependence
  - RAW, uses value computed earlier
  - Dependant must wait until result is ready
- Artificial dependence
  - WAR and WAW, register reuse
  - Solved by register renaming

Control Dependence

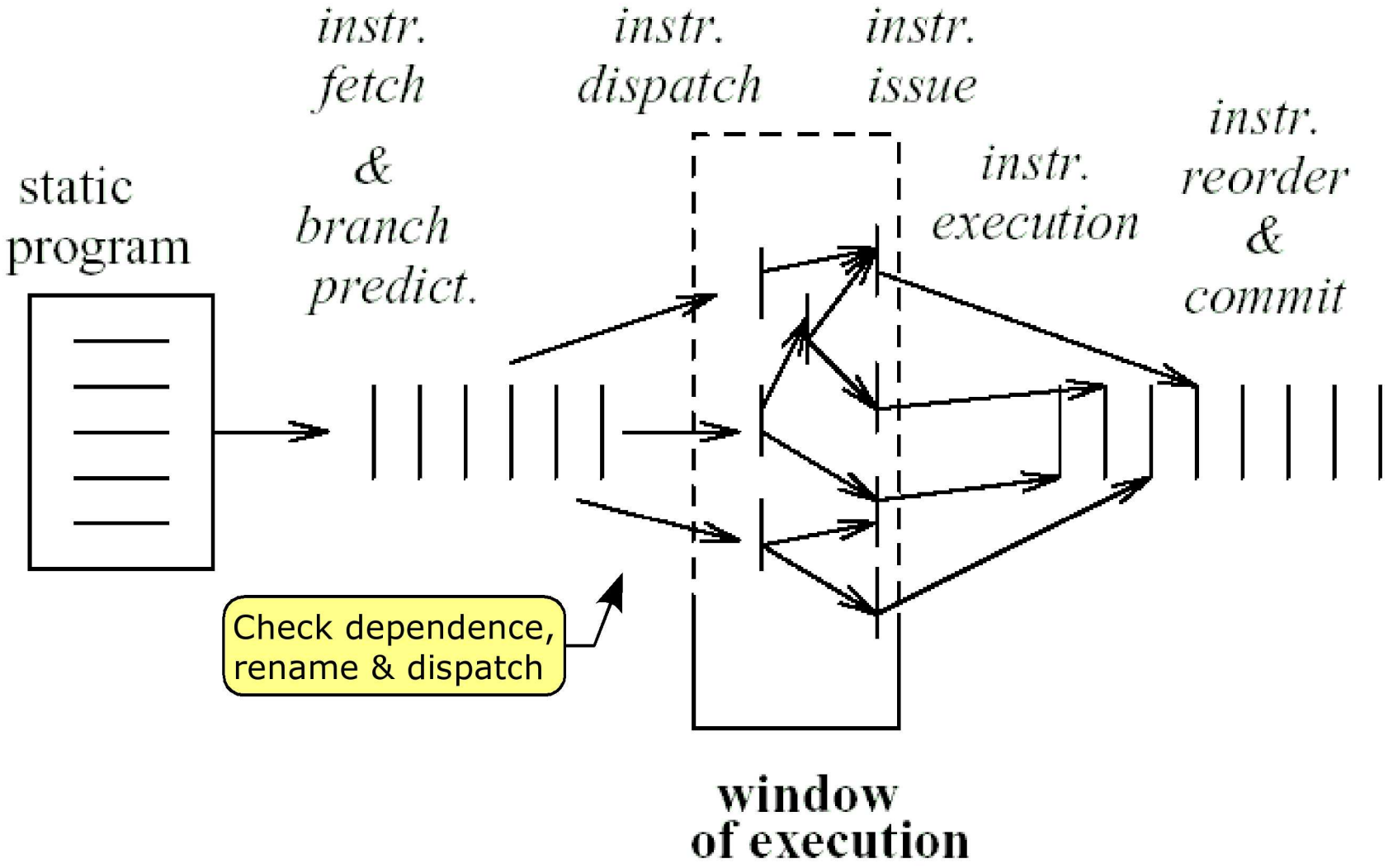
- Basic block can be dispatched in bulk
- Where do we proceed after conditional branch
  - Predict outcome and speculatively execute
  - If wrong, restore state



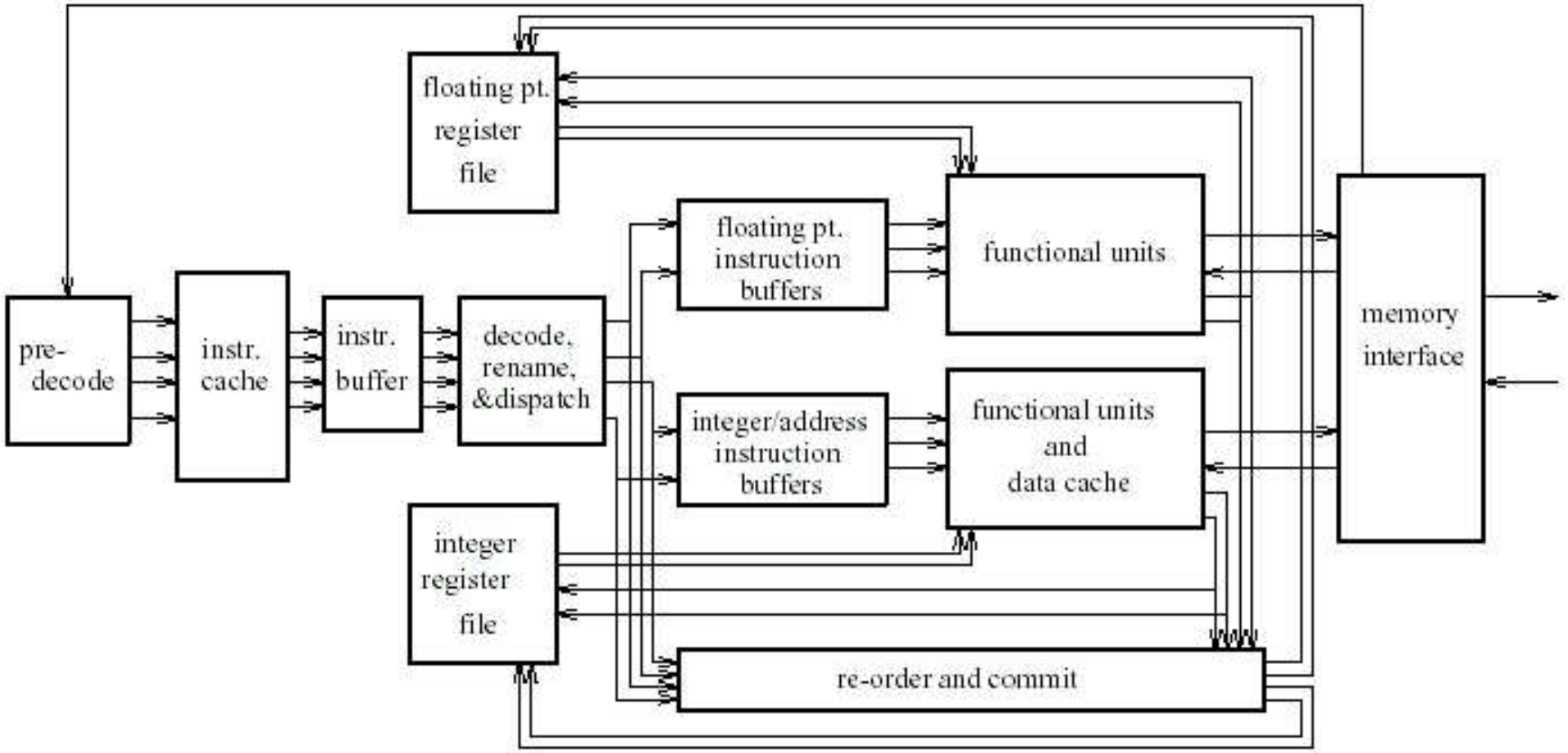
Branch Prediction

- Prediction methods
  - static = special/certain operations or compiler
  - dynamic = use past history in prediction table (often counters for taken branch)
- Redirection takes some time
  - Branch target address cache
  - Possible pipeline bubble

Flow of Execution



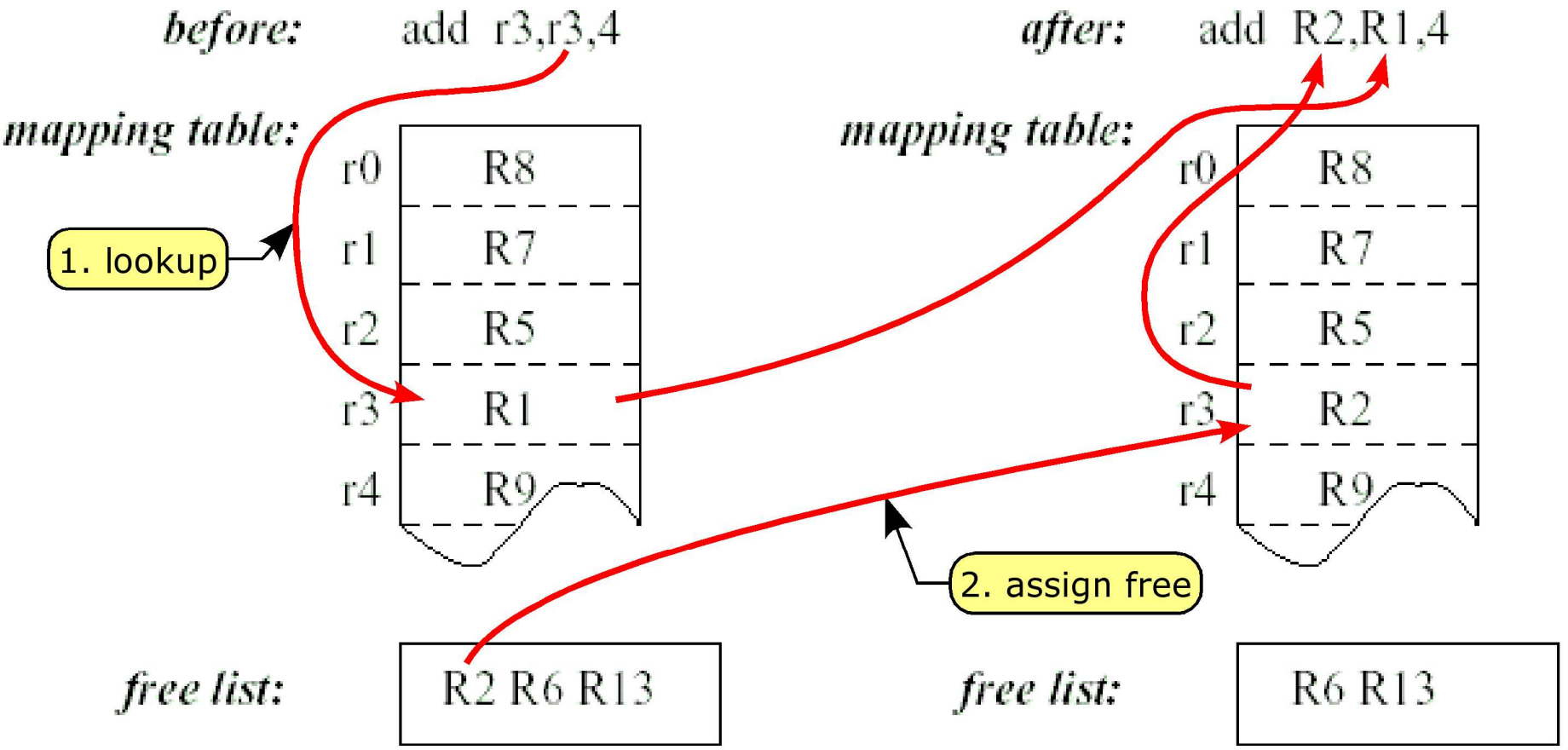
Organization



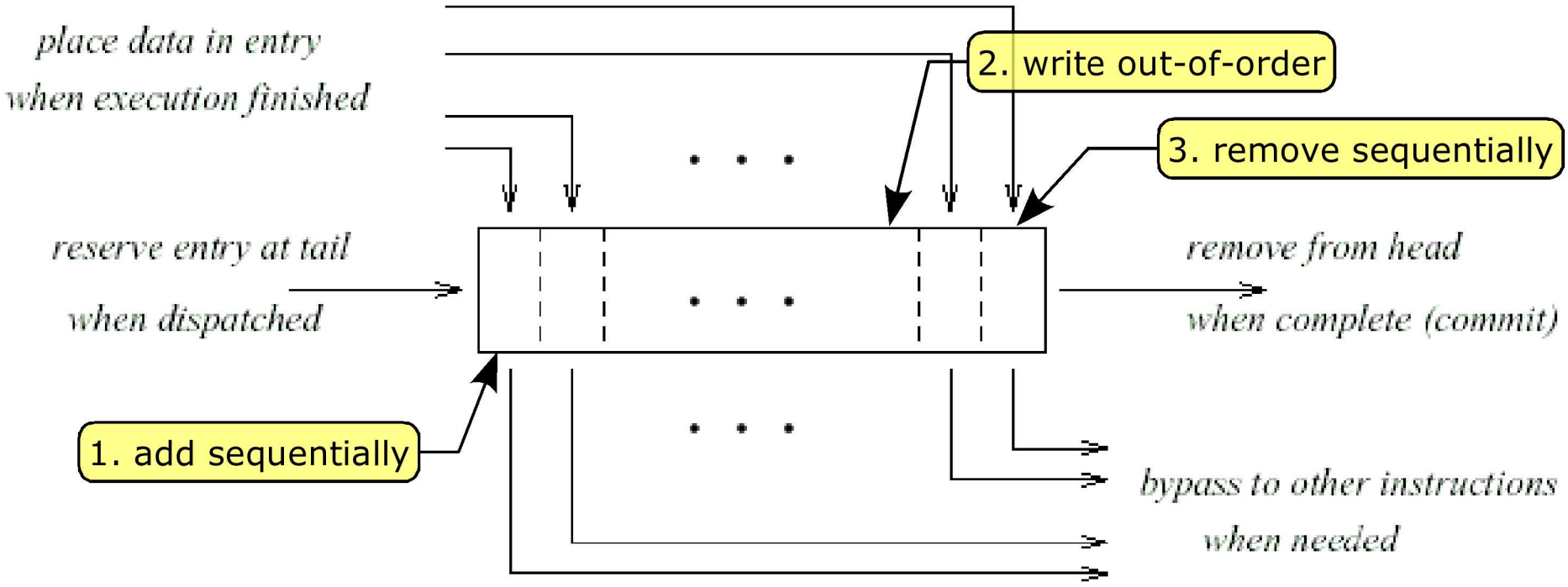
Register Renaming

- Two methods
  - Larger physical than logical register file
    - Lookup source registers, assign free physical register to result register (destination)
    - Count usage, return to free list on zero
  - Reorder buffer
    - One entry per active instruction in reorder buffer (also gives use precise interrupts)

Example (larger physical)



Example (reorder buffer)



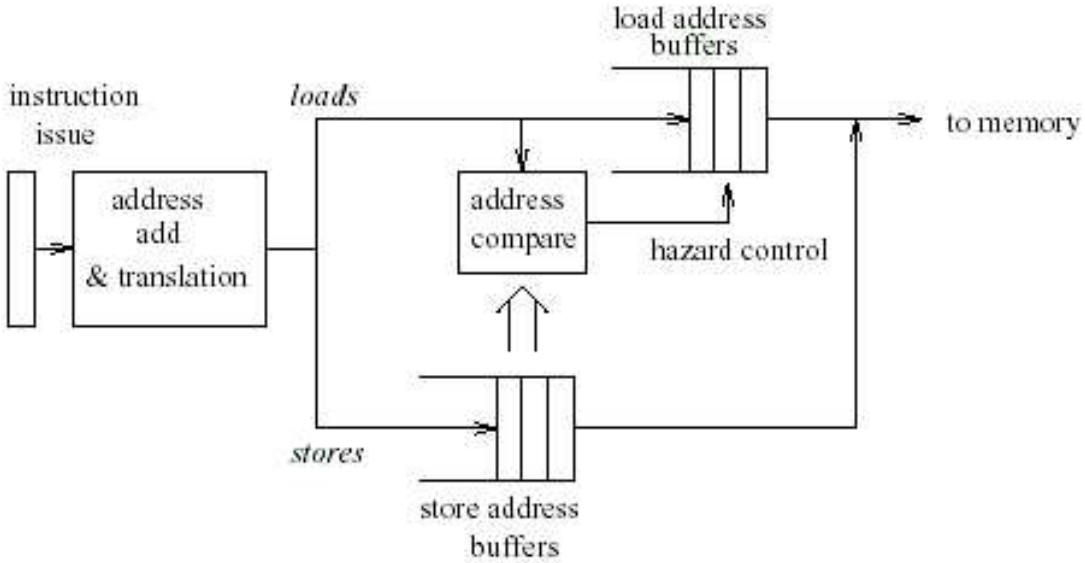
- Same style mapping table as previous example but it can contain references to values in reorder buffer

Issue Buffer

- Single queue
  - No out-of-order issue
  - No reordering necessary
- Multiple queues
  - One queue for each unit type
  - Restricted renaming (e.g. only mem. unit)
- Reservation stations
  - Full out-of-order execution
  - Reservation station ingates results needed
  - Designator could point to reorder buffer

Memory Handling

- Use memory hierarchies
- Memory system needs to serve multiple references per cycle
  - Allow operations to overlap or to be handled out-of-order, multiport lowest memory level
  - Use store address buffer for pending writes, check buffer on load and store



Example Processors

- MIPS R10000
  - Dynamic instruction scheduling
  - Branch prediction(counter) and resume cache
  - Reorder buffer
- Alpha 21164
  - No instruction scheduling, no OOO execution
  - Branch prediction (counter)
  - Single queue instruction buffer
- AMD K5
  - Pre-decode; Decode translate CISC → multiple ROP
  - Instruction scheduling and reorder buffer

Pros and Cons

+ Can execute more than one instruction per cycle
+ Handles varying memory speed
– Multiported units for greater parallelism increases hardware complexity
– Control complexity grows quadratically
– Memory speed still behind

Questions

- Seems that the limit for superscalars has been reached. How has modern architectures solved it?
- Alpha 21164 kind of seems like it goes against the spirit of superscalar by avoiding out of order instructions. How does it measure up to the others?
- Is a combination of hardware and software solutions the way to go for higher performance?
- Have there been any significant improvements/ revisions extending this microarchitecture?
- What was the outcome of the VLIW civil war?
- How do we know that introducing all of this complexity for better parallelism is worth it? Is it clear that this is better than trying to squeeze every last bit of speed out of a RISC processor?