
State Recovery

Wen-mei Hwu

ECE

University of Illinois, Urbana-Champaign

Needs for Recovery

- Branch Prediction Misses
 - recover from speculative execution of post-branching instructions
- Exceptions
 - memory management and protection
 - advanced control flows

Exceptions

- Conditions
 - not handled by main program logic
 - too expensive to check with explicit instructions
 - too cumbersome to model with branches
 - must be handled before resuming execution

Exception Types

- Fault:
 - violating instruction retried after handling exception
- Trap:
 - execution resumes after the violating instruction (like calls).

Precise State (PS)

- Fault PS
 - the CS that corresponds to dynamic PC just before the faulting instr.
- Trap PS
 - the CS that corresponds to dynamic PC just after the trapping instr

Why is PS Desirable?

- Diagnosis:
 - find the cause of the exception
- Handling:
 - give the handler a correct context
- Resuming:
 - simply re-activate the architectural state after exception handling

Implementation Model

- Pending Consistent State (PCS)
 - An architectural state that evolves with execution towards a CS.
- Current space:
 - the architectural space used by the out-of-order execution engine
 - the only space if no checkpoint repair

Implementation Model

- Backup space:
 - A logical copy of the memory space and the architectural regs which holds one PCS during execution
 - Could be explicitly implemented as physical copies
 - Could be explicitly implemented with difference lists

Checkpoint

- A dynamic program counter at which CS can be delivered during recovery.
- a PCS is maintained for each checkpoint

Repair Range

- Mapping of each checkpoint to a sequence of dynamic instructions
 - any exception caused by instructions in the repair range of a checkpoint will cause a repair back to that checkpoint
- A check point is active when an instruction in its repair range is still pening

Practical Implementations

- Difference lists (ROB, HB)
 - only one space physically exists
 - all others are expressed as delta from the physical space
- Physical copies (CKPT)
 - straightforward implementation
 - attractive for very large windows

Reorder Buffer

- Proposed by Smith and Plezskun
- Register file and memory space exist as the backup(c) space.
- All other spaces are represented by differences from the backup(c) space.

Reorder Buffer

- A reorder buffer of m entries is provided: each entry buffers
 - one state modification (register/memory write)
 - one program counter
- The reorder buffer is logically an m -stage queue

Reorder Buffer Issues

- Bypassing
 - must bypass result from reorder buffer to subsequent instructions to avoid unnecessary interlocking
- Tagging
 - execution results must be delivered to ROB in Tomasulo's algorithm

P6-like Example

- Commit instructions in-order using Reorder Buffer (ROB)
 - Keep Retirement Register File (RRF) in precise state
 - Be able to clear contents of RS, ROB, pipes, etc. and start over
 - Retire uops for short x86 instructions together

P6-like Example (cont.)

- Retire uops for x86 string ops so always at a restartable point
- Commit stores at uop retirement
- Delay exception handling until uop retired
 - May be speculatively executed
- Update RAT with retirement info

Fixing Mispredicted Branches

- Start fetching from correct path as soon as misprediction detected
 - Takes eight cycles for correct-path ops to reach the RS

Fixing Mispredicted Branches (cont.)

- Prevent wrong-path ops from affecting state
 - Branches should not redirect Ifetch
 - Stores should not affect memory state
 - Exceptions should be ignored
 - Most cases handled during in-order retirement from ROB

Fixing Mispredicted Branches (cont.)

- Fix RAT
 - Correct-path ops must be stalled at renaming until RAT fixed
 - Could fix when misprediction detected
 - Complex implementation
 - Take into account pending correct-path ops
 - Ops could be delayed at renaming stage

Fixing Mispredicted Branches (cont.)

- Could fix when mispredicted branch retires
 - Simple reset of RAT state
 - correct-path ops not delayed at renaming stage

Fixing Mispredicted Branches (cont.)

- Remove wrong-path operations from RS, pipelines, ROB, etc.
 - Could nullify as soon as misprediction detected
 - Complex implementation required to selective nullify ops
 - Minimizes performance lost to resource contention

Fixing Mispredicted Branches (cont.)

- Could wait until all pending operations are off-path
 - Reset of RS, pipelines, ROB, etc. then can be done
 - Some performance may be lost due to resource contention

History Buffer

- Register file and memory space exist as the Current space
- All other spaces are represented by differences from the Current space

History Buffer

- A buffer of m entries is provided:
each entry
 - preserves the previous value before one state modification
 - buffers one program counter
- The history buffer is logically an m -stage stack.

History Buffer Issues

- Tagging
 - execution results must be delivered to HB in Tomasulo's algorithm to ensure ability to recover

Checkpoint Copies

- Multiple copies of register files
- much faster branch misprediction recovery
- State recovery via physical information transfer or reset of *current* pointer
- Used in MIPS R10000 to reduce branch prediction miss penalty