MI-SP-SP-13 (MI-#GEN) (lecture 10)

Generování kódu: HW závislé optimalizace, optimalizace pro instrukční pipelining

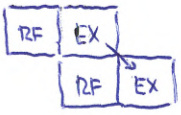
Pipelined processors

- each functional unit is independent → multicycle design
- each tick one instruction executed (ideally)

→ pipeline hazards

① - instruction needs result of previous one → stall

→ feed-forward



- some combinations can still result in stall

lw
add

STALLED

② - instruction reordering - we must respect data dependencies
- ▷ read after write
- ▷ write after read
- ▷ write after write

Instruction scheduling → create DAG between instructions representing dependencies
→ sorting the graph with any topological sort is ok
- choosing best order is NP-hard problem → heuristics

- emit instruction that: - does not conflict with previous one
  - is likely to conflict
  - as far as possible from possible conflict

→ construct scheduling DAG $O(n^2)$
→ emit instructions from candidate list or either NOP (on some platforms) or inst satisfying at least last two rules

Dynamic scheduling

- modern CPUs (some) are scheduling instruction dynamically
  - compiler technique in hardware

- CPU sees only smaller group of instructions
  - sometimes can predict e.g. branches better than compiler could

[Out-of-order & Speculative execution]