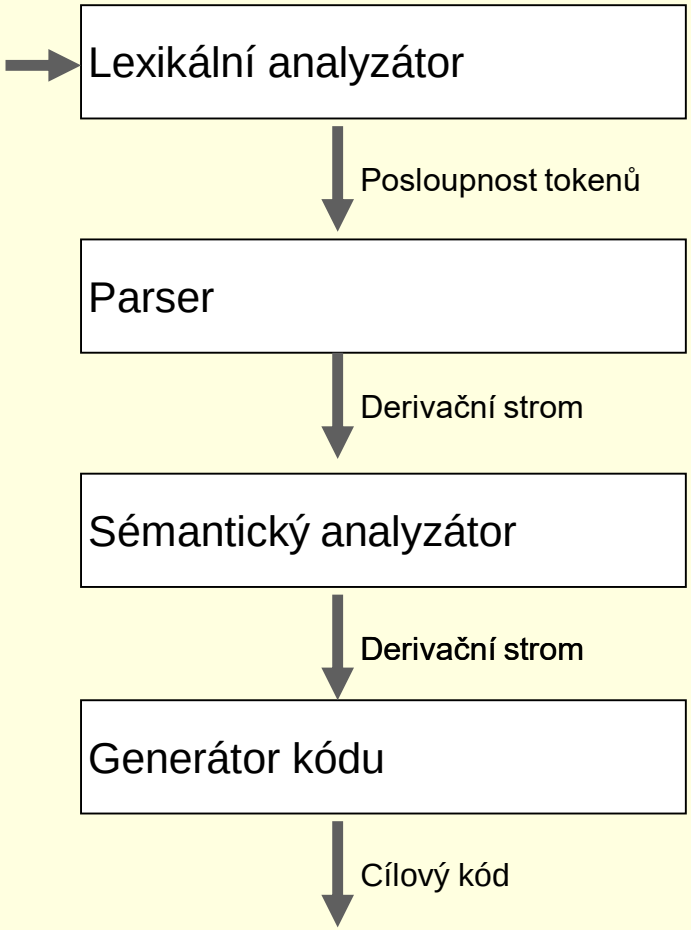
Architektura překladače

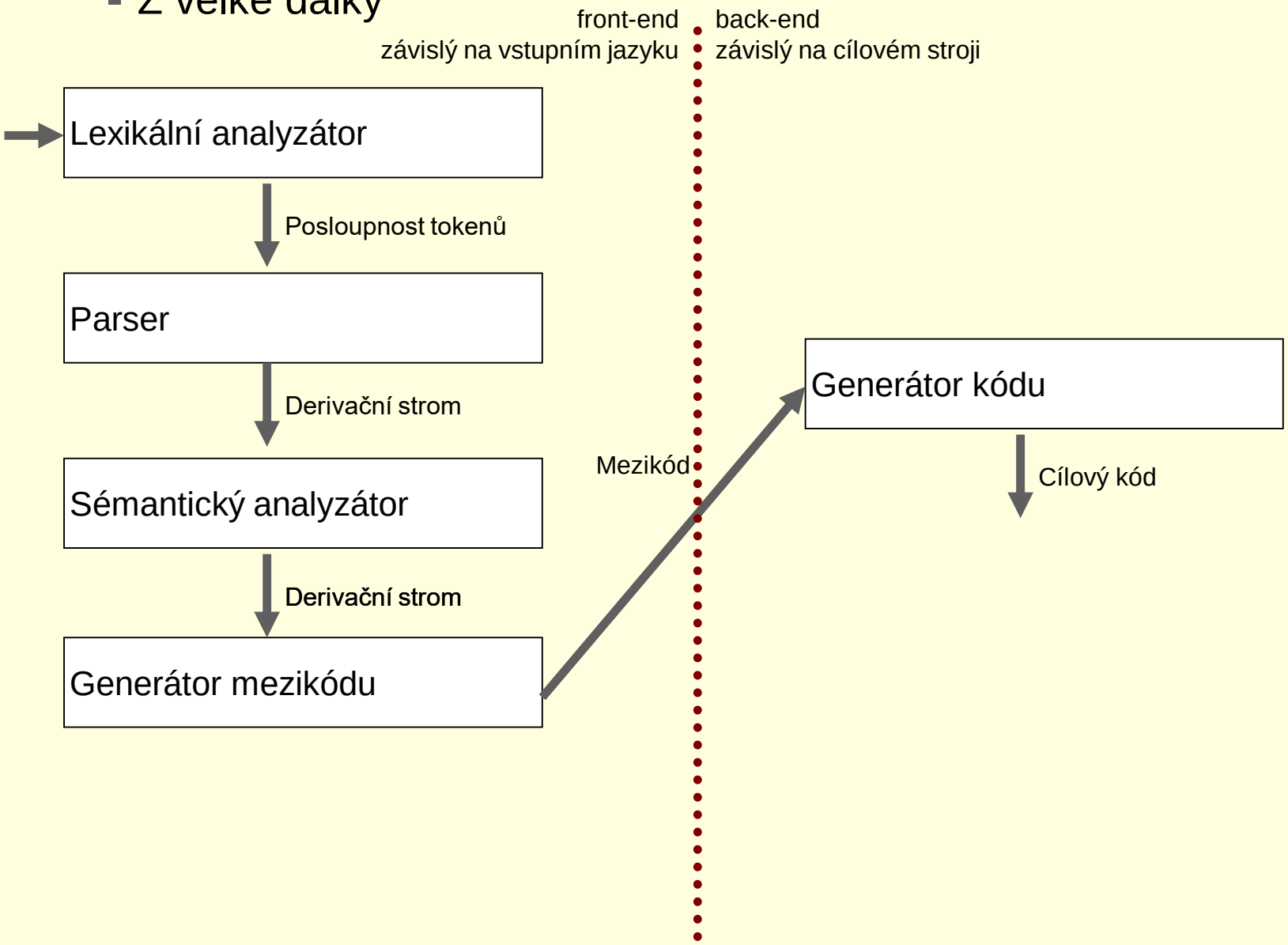
Architektura překladače

- Amatérský pohled



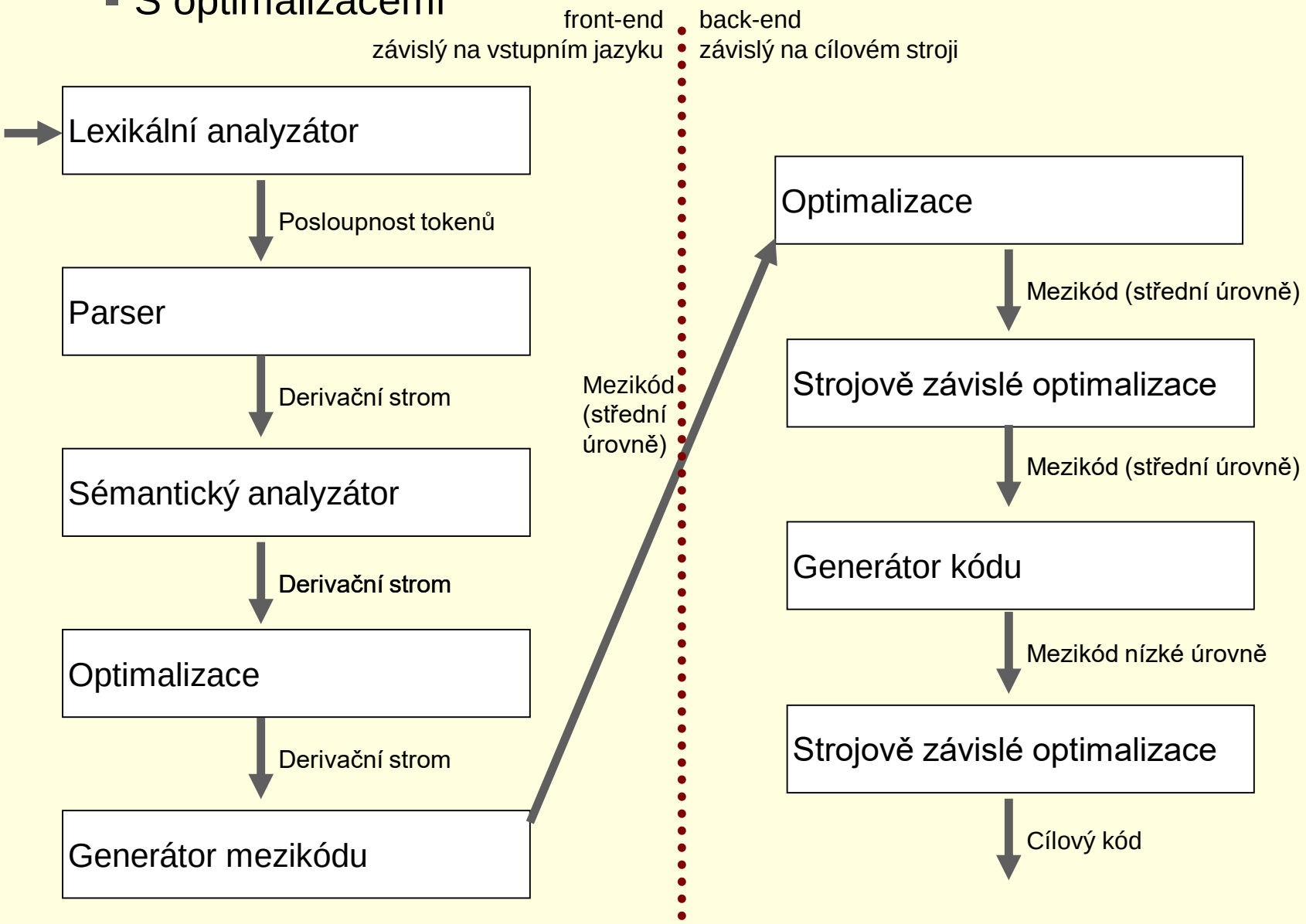
Architektura překladače

- Z velké dálky



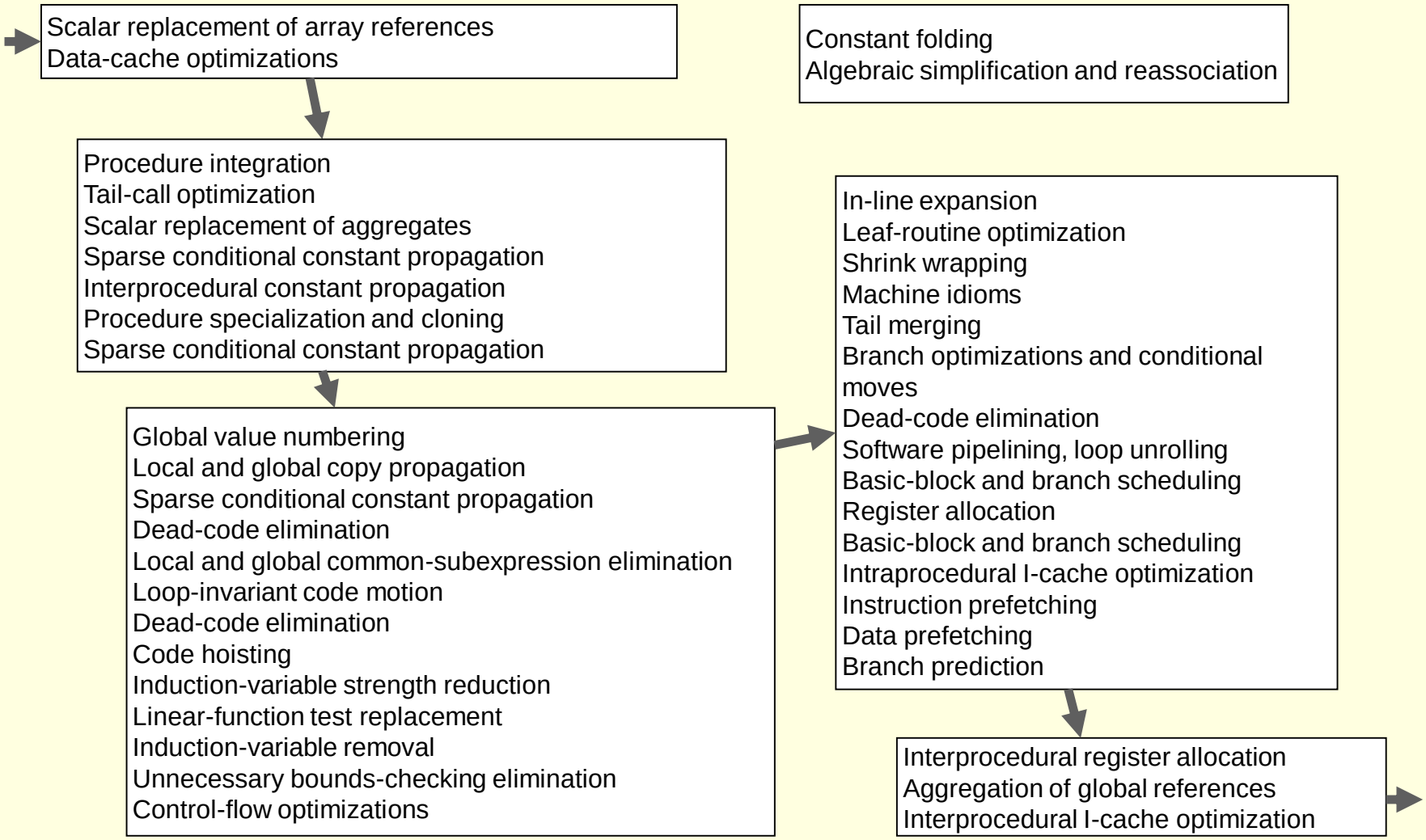
Architektura překladače

- S optimalizacemi



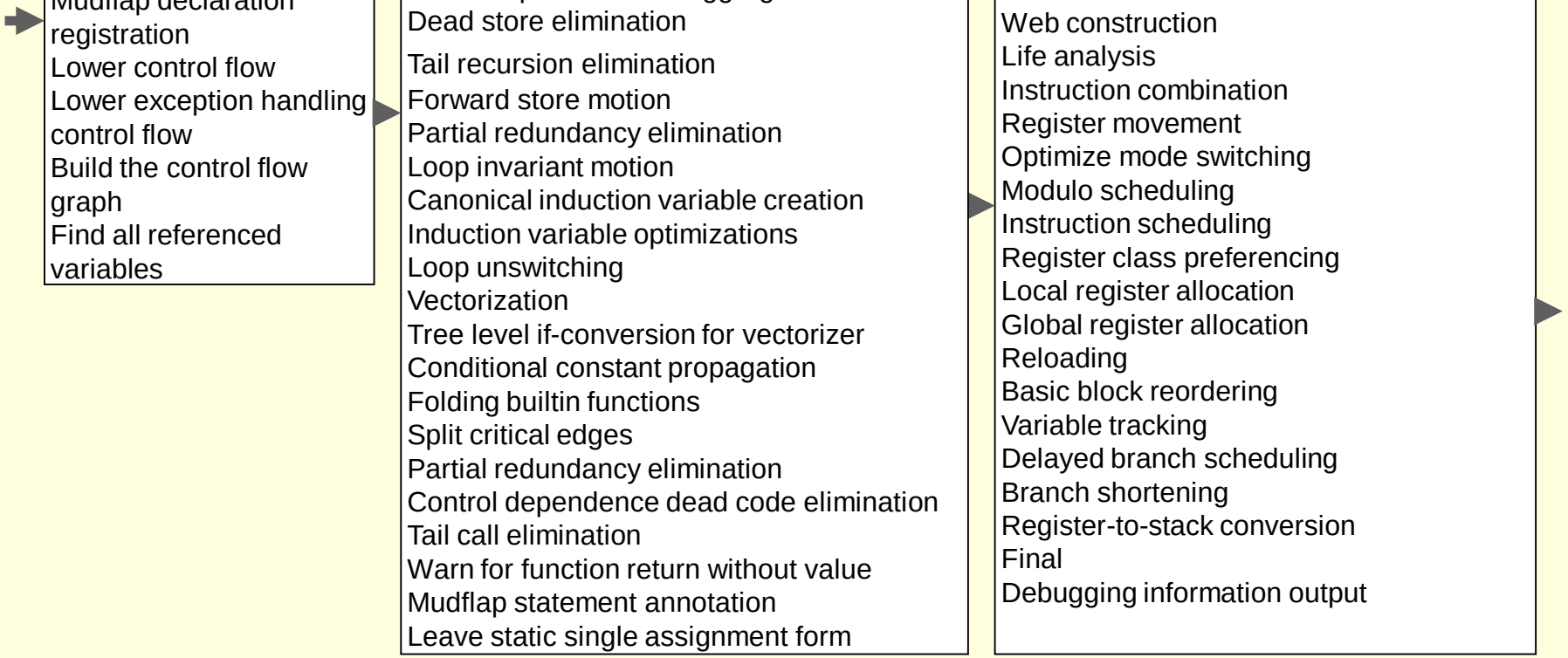
Architektura překladače

- Detailní pohled akademika (pouze optimalizace)
  - Muchnick: Advanced Compiler Design and Implementation



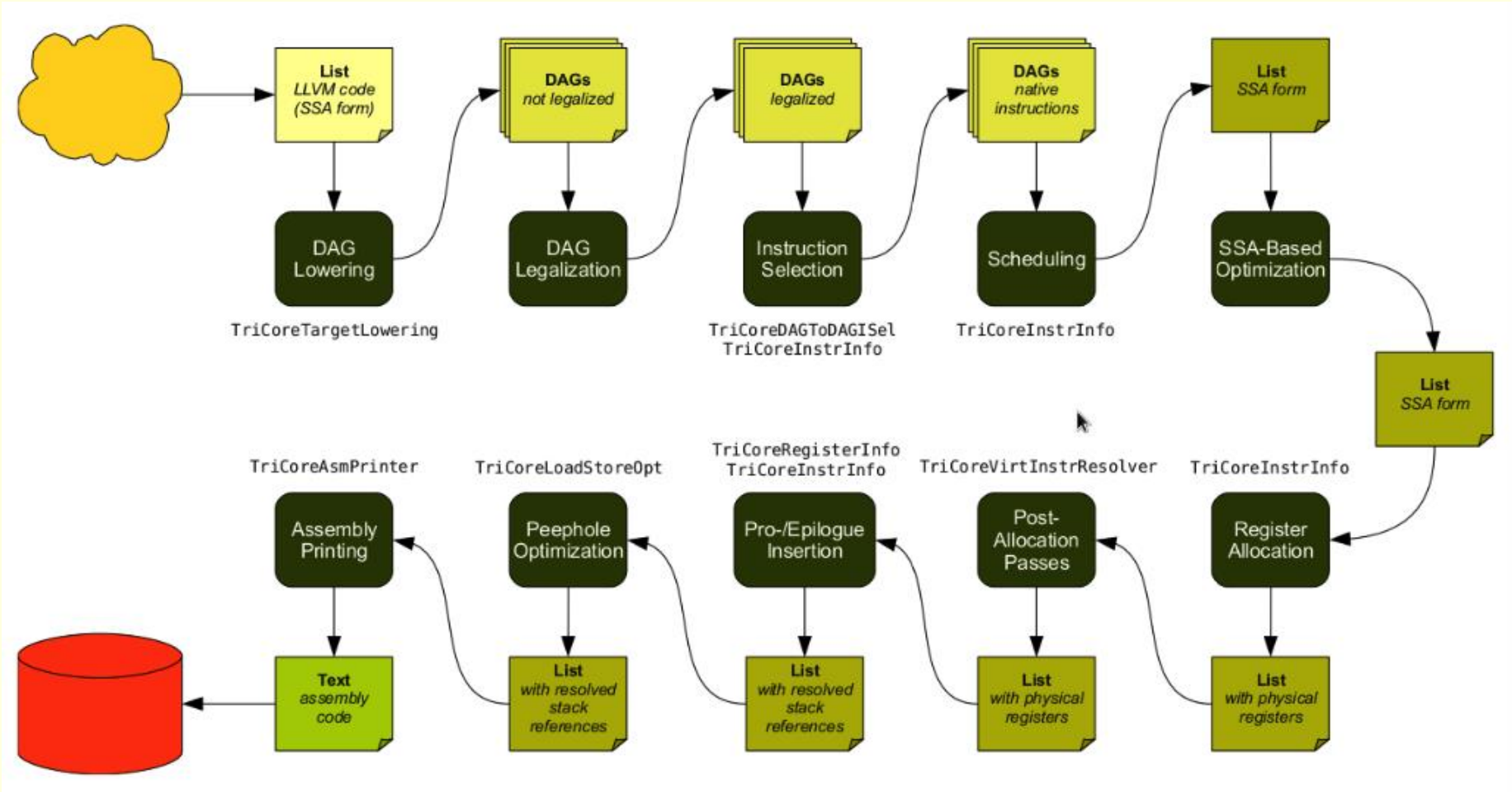
Architektura překladače

- Realita
  - GNU Compiler Collection Internals



Architektura překladače

- Realita
  - LLVM back-end

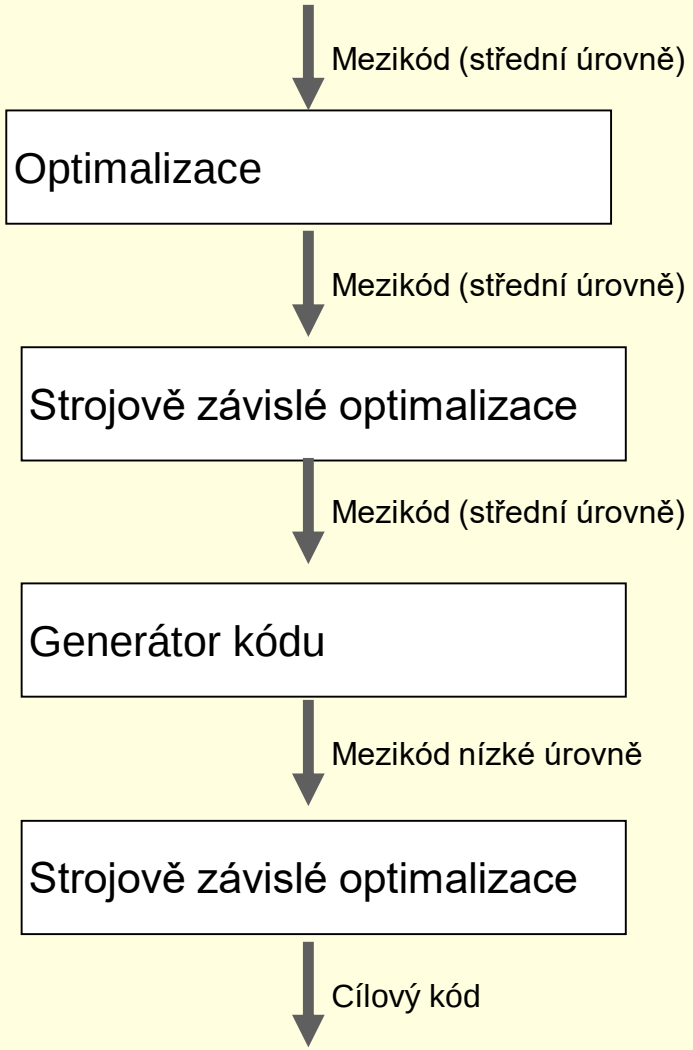


LLVM 6.0 code generator (AMD64, -O3 -mavx2)

- Instruction Selection
- Expand ISel Pseudo-instructions
- X86 Domain Reassignment Pass
- Tail Duplication
- Optimize machine instruction PHIs
- Merge disjoint stack slots
- Local Stack Slot Allocation
- Remove dead machine instructions
- Early If-Conversion
- Machine InstCombiner
- X86 cmov Conversion
- Machine Loop Invariant Code Motion
- Machine Common Subexpression Elimination
- Machine code sinking
- Peephole Optimizations
- Remove dead machine instructions
- Live Range Shrink
- X86 Fixup SetCC
- X86 LEA Optimize
- X86 Optimize Call Frame
- X86 WinAlloca Expander
- Detect Dead Lanes
- Process Implicit Definitions
- Live Variable Analysis
- Machine Natural Loop Construction
- Eliminate PHI nodes for register allocation
- Two-Address instruction pass
- Simple Register Coalescing
- Rename Disconnected Subregister Components

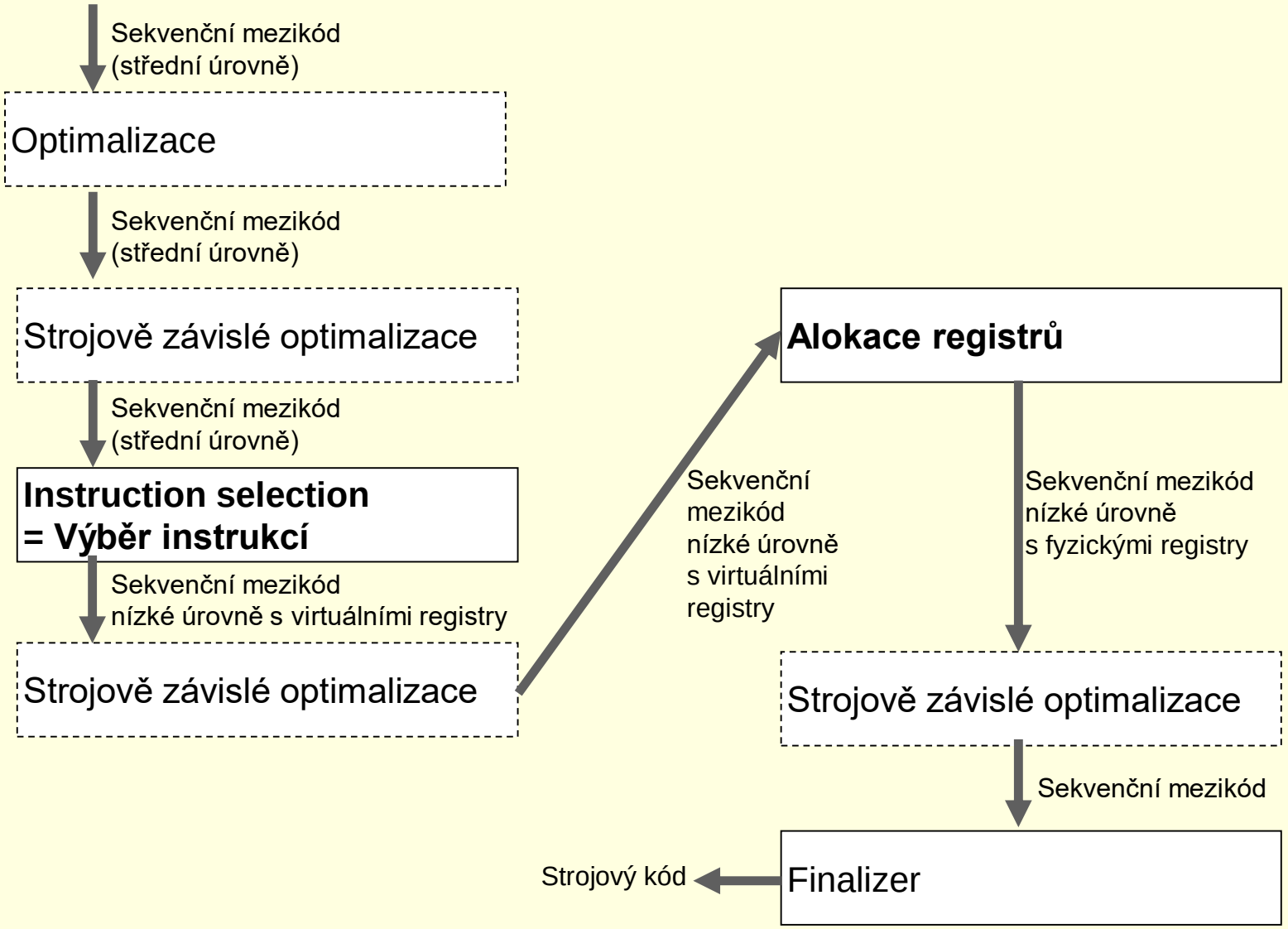
- Machine Instruction Scheduler
- Greedy Register Allocator
- Virtual Register Rewriter
- Stack Slot Coloring
- Machine Loop Invariant Code Motion
- X86 FP Stackifier
- Shrink Wrapping analysis
- Prologue/Epilogue Insertion & Frame Finalization
- Control Flow Optimizer
- Tail Duplication
- Machine Copy Propagation Pass
- Post-RA pseudo instruction expansion pass
- X86 pseudo instruction expansion pass
- Post RA top-down list latency scheduler
- Analyze Machine Code For Garbage Collection
- Branch Probability Basic Block Placement
- X86 Execution Dependency Fix
- X86 vzeroupper inserter
- X86 Byte/Word Instruction Fixup
- X86 Atom pad short functions
- X86 LEA Fixup
- Compressing EVEX instrs to VEX encoding when possible
- Contiguously Lay Out Funclets
- StackMap Liveness Analysis
- Live DEBUG_VALUE analysis
- Insert fentry calls
- Insert XRay ops
- Implement the 'patchable-function' attribute
- X86 Retpoline Thunks

- **Různé vnitřní reprezentace**
  - Mezikód střední úrovně
    - Nezávislá sada operací
      - ADD_I32 a,b,c
    - Forma
      - Nesekvenční
      - Částečně sekvenční
  - Mezikód nízké úrovně
    - Ekvivalenty strojových instrukcí
      - add r1,r2
    - Forma
      - Nesekvenční
      - Částečně sekvenční
      - Sekvenční



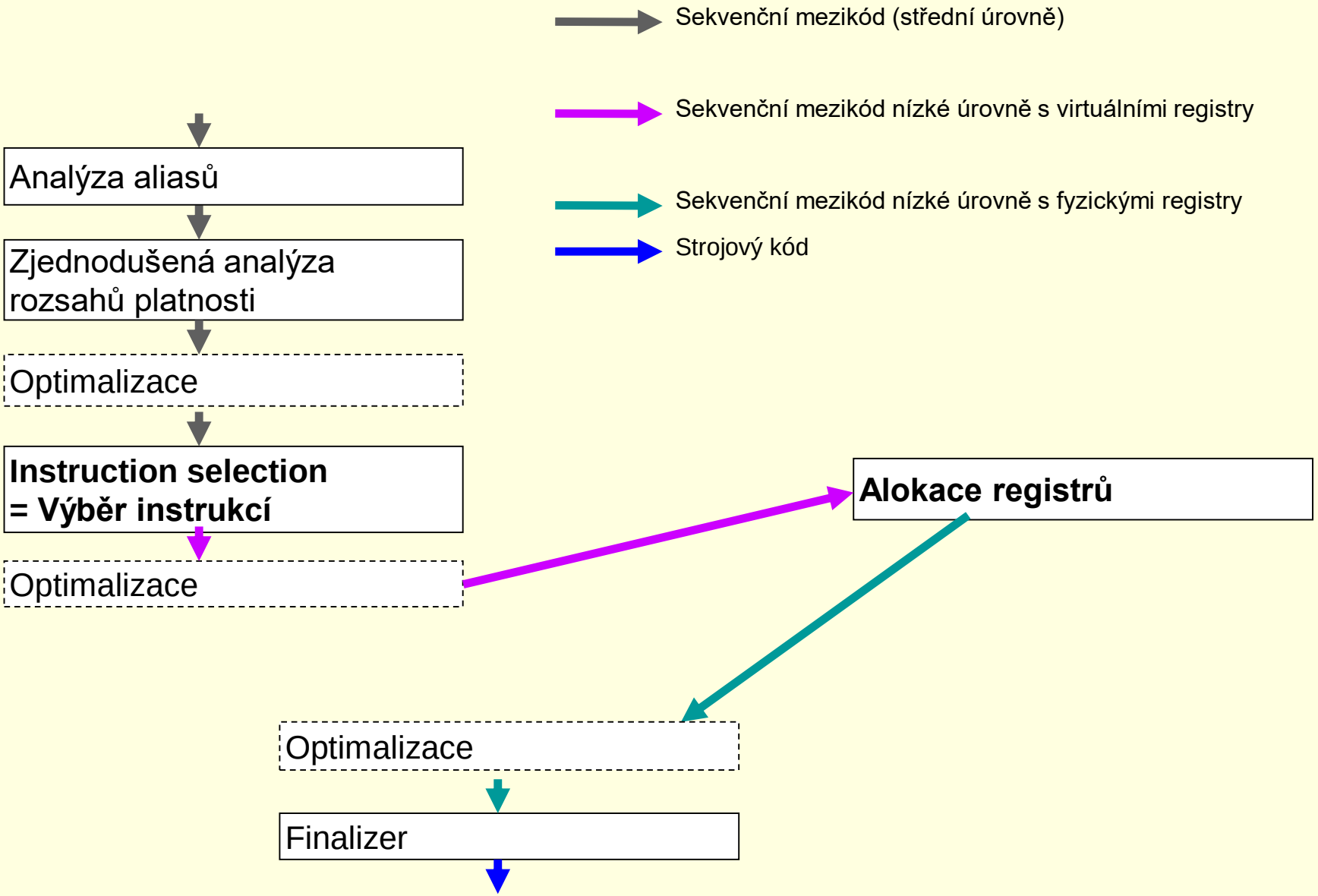
Architektura back-endu

- Sekvenční mezikód



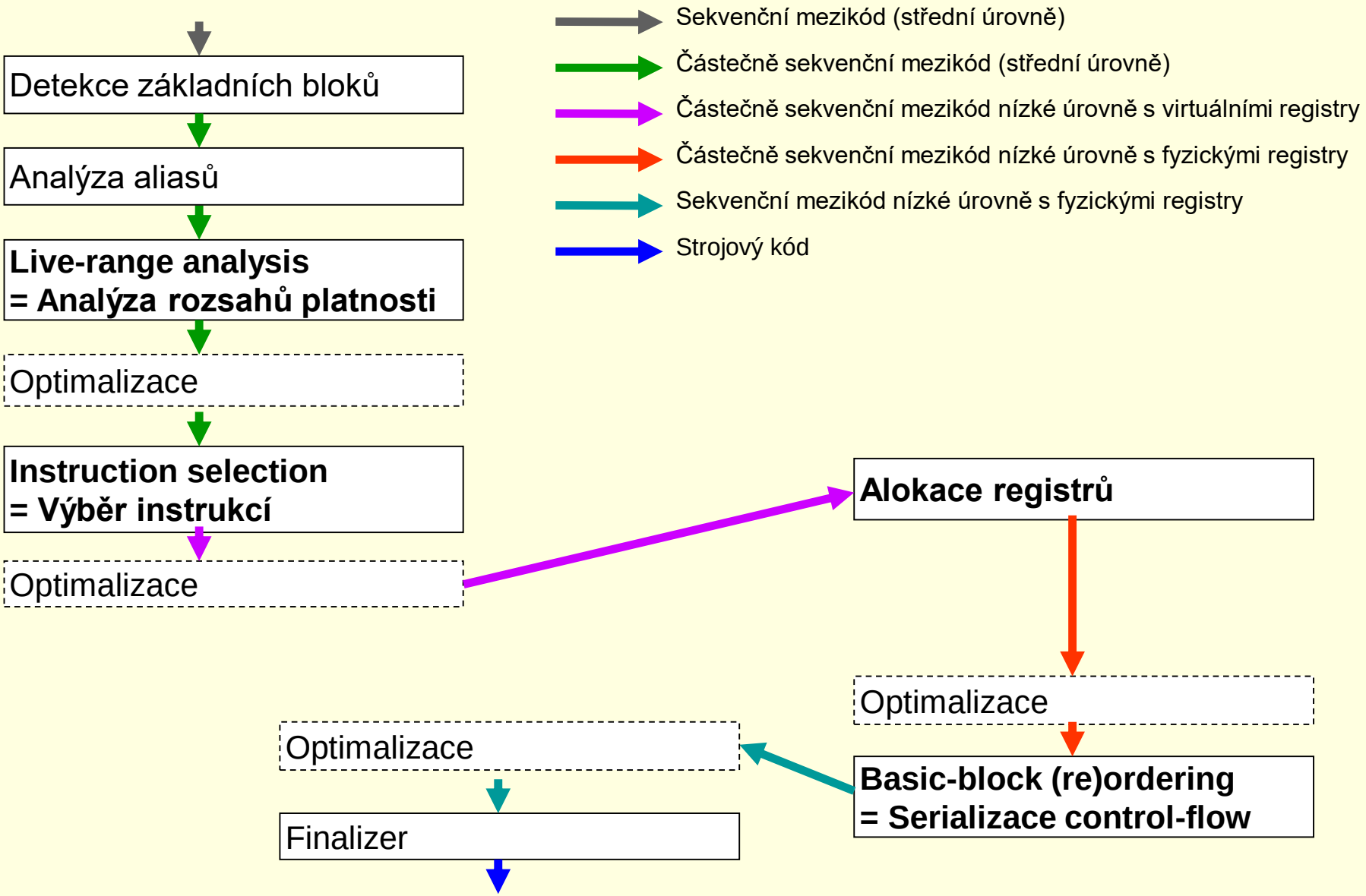
Architektura back-endu

- Sekvenční mezikód



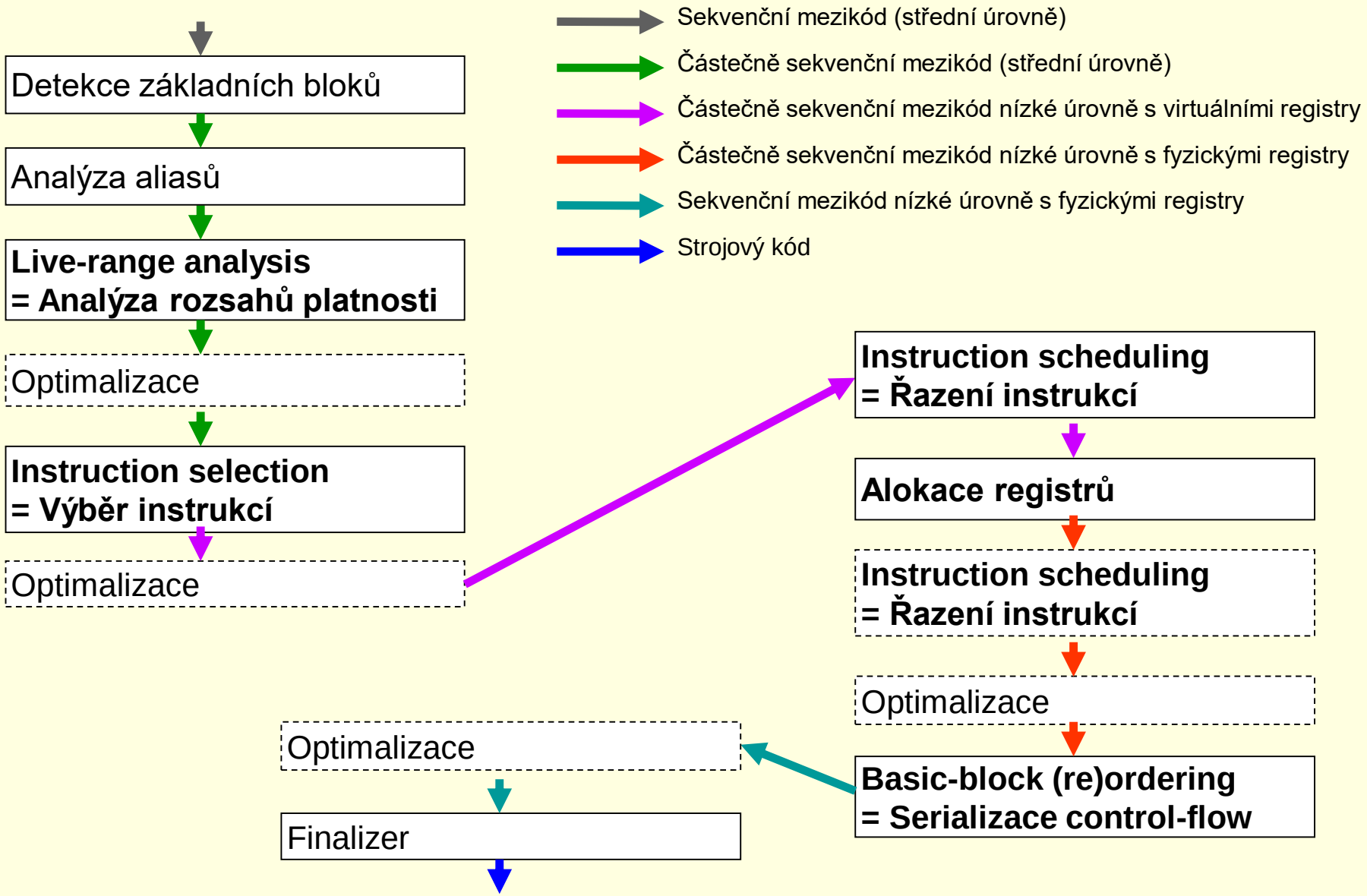
Architektura back-endu

- Částečně sekvenční mezikód bez schedulingu



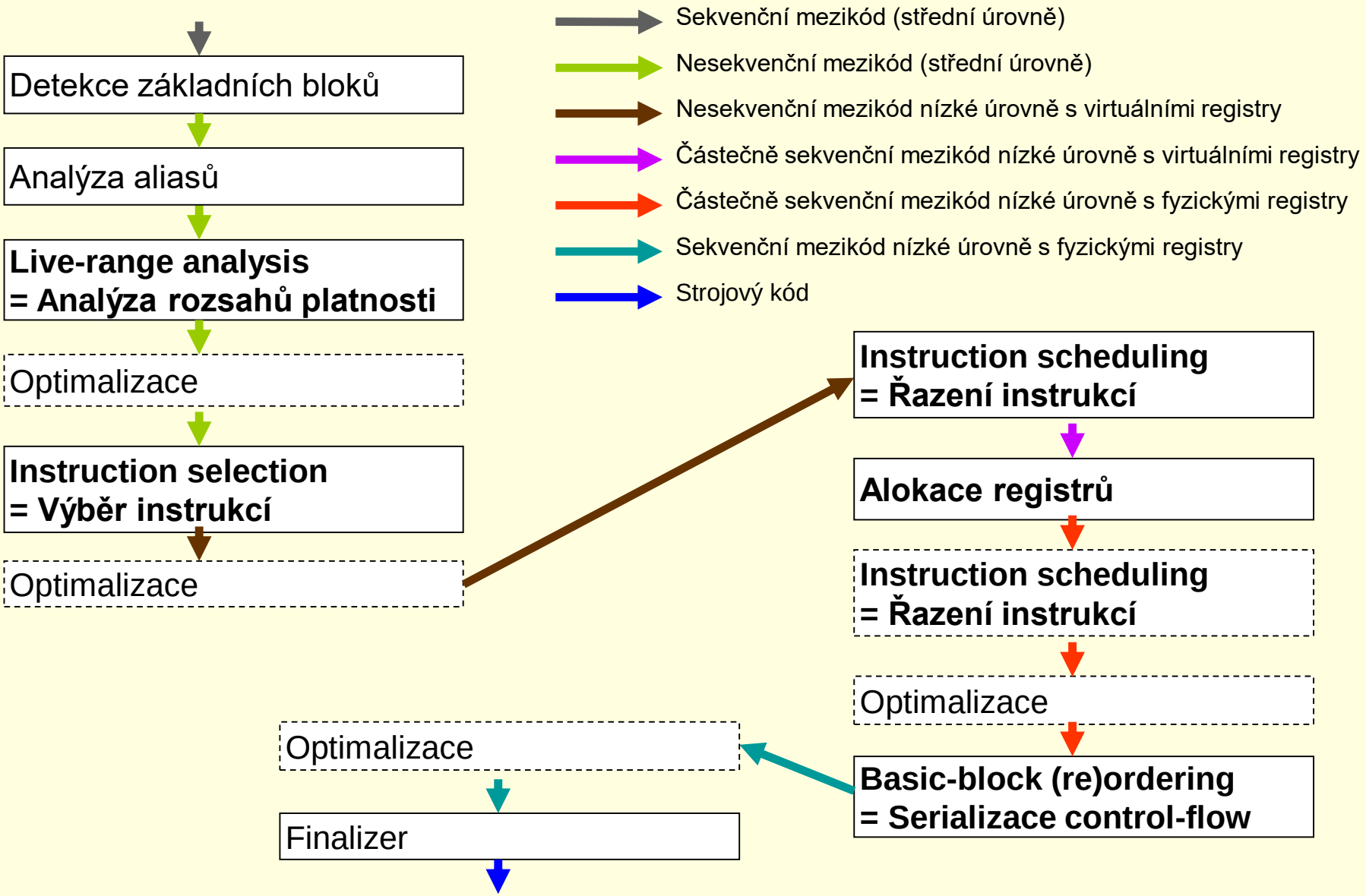
Architektura back-endu

- Částečně sekvenční mezikód se schedulingem



Architektura back-endu

- Nesekvenční mezikód



Architektura back-endu

- Instruction selection
  - Výběr strojových instrukcí
    - 1:n – přímočaré řešení
    - m:n – stromové/grafové gramatiky apod.
  - Vliv na kvalitu kódu poklesl
    - RISC, load-store kód apod.
- Instruction scheduling
  - Řazení instrukcí pro lepší využití ILP (instruction-level parallelism)
    - NP-úplná úloha
  - Lokální v BB
    - Speciální řešení smyček (software pipelining)
    - Částečně globální varianty (trace scheduling)
  - Zrychluje kód o 30-150%
- Register allocation
  - Přidělování fyzických registrů
    - NP-úplná úloha
    - Standardní řešení: Barvení grafu