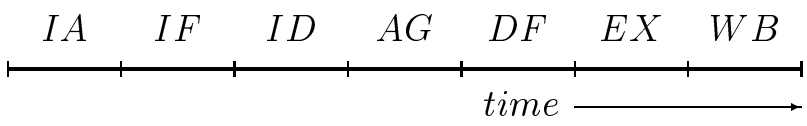
Pipelining

→ concerns executing several instructions simultaneously in as many consecutive pipeline stages of a processor in the same order as listed in the program text;
→ objective is increased instruction throughput while the execution time of individual instructions remains the same as in well mapped processors ⇝ ideally one instructions each enters and leaves the pipeline each cycle;
→ assumes total mutual independence of instructions in the pipeline ⇝ is not generally the case ⇝ throughput is less than one instruction per cycle;
→ instruction execution must be partitioned into as many phases of equal length in time corrsponding to pipeline stages;

typical phases are

IA ⇝ instruction addressing in memory
IF ⇝ instruction fetch (from cache)
ID ⇝ instruction decode
AG ⇝ address generation (for mem operands)
DF ⇝ data fetch (from cache)
EX ⇝ executing operation
WB ⇝ write back result (to register)



Organizing pipeline operation

static pipelining → all instructions move in order through all pipeline stages, even if certain stages can be bypassed by specific instructions ⇝ e.g. EX phase of LD | ST instructions, AG | DF phases of RR instructions ⇝ single pipeline

dynamic pipelining → permitt | support bypassing pipeline stages

type_1 → preserves in order execution wrt ID and WB phases (otherwise out-of-order execution) ⇝ writing results must be delayed until all preceding instructions have completed their WB phases ⇝ almost nothing is gaind wrt static pipelining ⇝ single pipeline;

type_2 → preserves order wrt ID phase, permits out-of-order execution and completion of EX and WB, provided there are no dependency violations among the instructions in the pipeline ⇝ parallel pipeline organization;

type_3 → instructions may be executed completely out-of-order through all phases | stages ⇝ out of a pool of executable instructions are selected for execution those which are not dependent on instructions in the pipeline ⇝ superscalar pipeline organization;