21164 Instruction Unit Pipeline

Fetch & issue

S0: instruction fetch
branch prediction bits read

S1: opcode decode
target address calculation
if predict taken, redirect the fetch
instruction TLB check

S2: instruction slotting: decide which of the next 4 instructions can be issued

- intra-cycle structural hazard check
- intra-cycle data hazard check
- steer issuable instructions to appropriate functional unit

S3: instruction dispatch

- inter-cycle load-use data hazard check
- inter-cycle structural hazard check
- register read

21164 Integer Pipeline

Execute (2 pipelines)

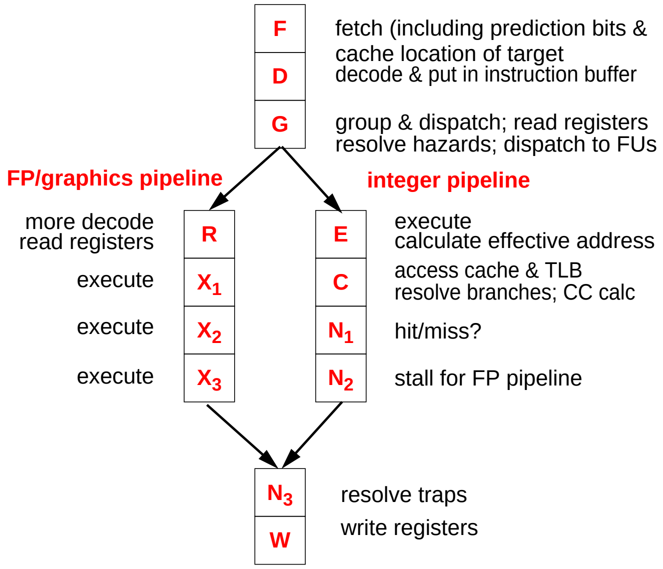
S4: integer execution
effective address calculation

S5: conditional move & branch execution
data cache access

S6: register write

UltraSPARC 1

Integrated integer & (partial) floating point pipeline



F — fetch (including prediction bits & cache location of target

D — decode & put in instruction buffer

G — group & dispatch; read registers resolve hazards; dispatch to FUs

FP/graphics pipeline

R — more decode read registers

X_1 — execute

X_2 — execute

X_3 — execute

integer pipeline

E — execute calculate effective address

C — access cache & TLB resolve branches; CC calc

N_1 — hit/miss?

N_2 — stall for FP pipeline

N_3 — resolve traps

W — write registers

Pentium Pro

10 stage fetch & decode pipeline (minimum)

BTB access (1 stage)

instruction fetch & align for decoding (2.5 stages)

decode & uop generation (2.5 stages)

register renaming & instruction issue to reservation stations (4 stages minimum)

4 stage integer pipeline (minimum)

execute, resolve branch (1 stage)

write registers (3 stages minimum)

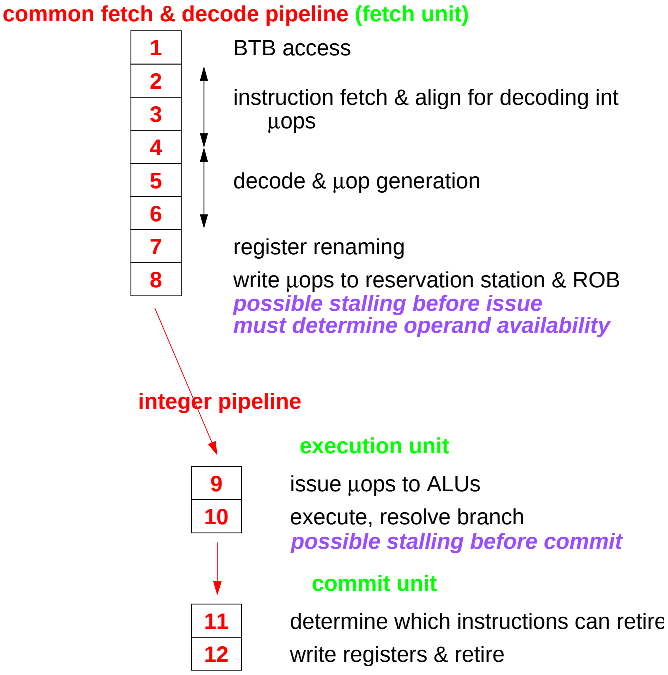
5 stage load pipeline (minimum)

address calculation & to memory reorder buffer (1 stage minimum)

integrated L1 & L2 data cache access

pipelined FP add & multiply

Intel P6 (Pentium Pro)



common fetch & decode pipeline (fetch unit)

1 BTB access

2–4 instruction fetch & align for decoding int µops

4–6 decode & µop generation

7 register renaming

8 write µops to reservation station & ROB

possible stalling before issue

must determine operand availability

integer pipeline

execution unit

9 issue µops to ALUs

10 execute, resolve branch

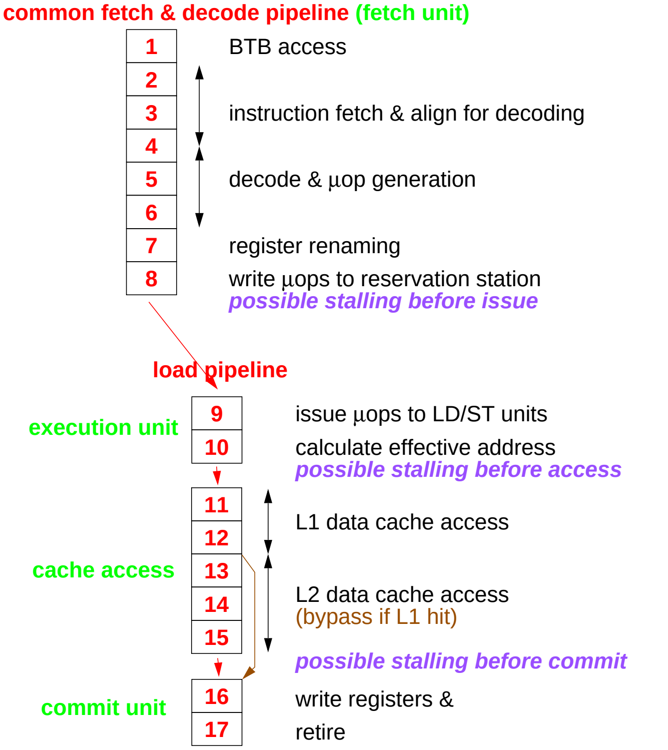
possible stalling before commit

commit unit

11 determine which instructions can retire

12 write registers & retire

Intel P6



common fetch & decode pipeline (fetch unit)

1 BTB access

2–4 instruction fetch & align for decoding

4–6 decode & µop generation

7 register renaming

8 write µops to reservation station

possible stalling before issue

load pipeline

execution unit

9 issue µops to LD/ST units

10 calculate effective address

possible stalling before access

cache access

11–12 L1 data cache access

13–15 L2 data cache access

(bypass if L1 hit)

possible stalling before commit

commit unit

16 write registers &

17 retire

Intel P6

Some bandwidth constraints: maximum for one cycle

- 16 bytes fetched
- 3 instructions decoded
- 5 instructions issued
- 1 load & 1 store access to the L1 cache
- 1 cache result returned
- 3 instructions to the reorder buffer
- 3 instructions committed

if

- good instruction mix
- right instruction order
- operands available
- functional units available
- load & store to different cache banks
- all previous instructions already committed

Comparison

Different pipeline structures

- partial decoding before placed in I cache (not X: too hard)
- good dynamic branch prediction
- slot/group/issue stage
- degree of pipelining

  A, S, X: most superpipelined

  M: old style
- number of pipelines

  M: integer, memory access, floating point all decoupled

  S: integrated integer & floating point

  X: integrated L1 & L2 data cache access