Instruction Level Parallelism and Superscalar Processors

William Stallings
Computer Organization
and Architecture
8th Edition

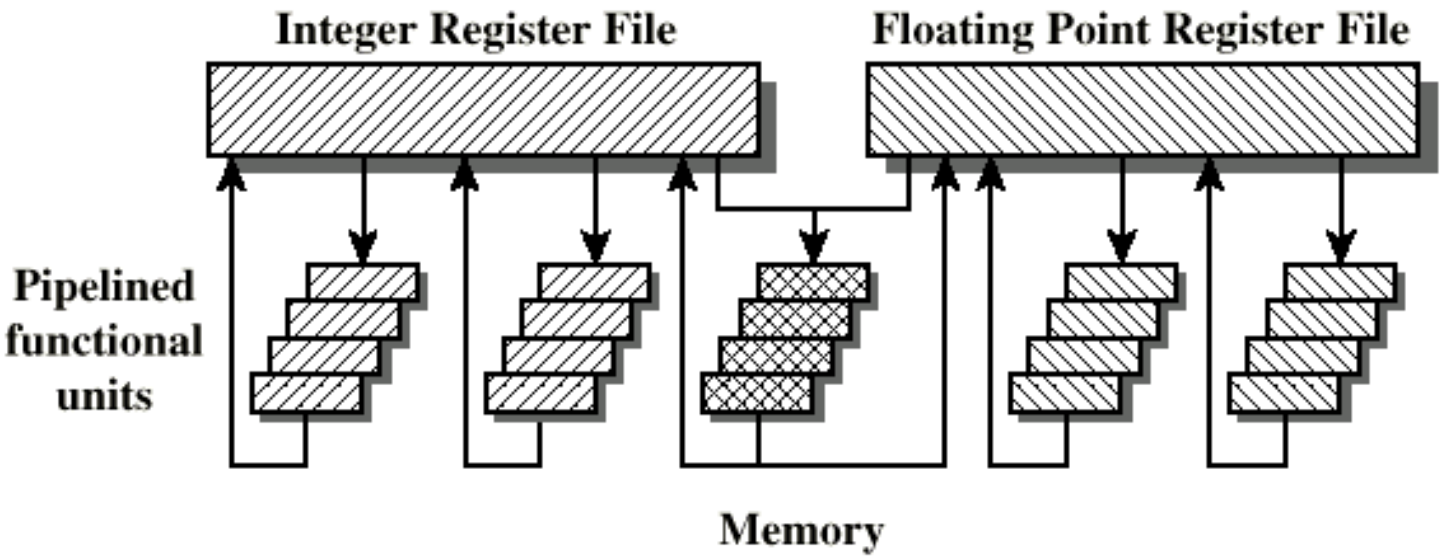
What is Superscalar?

- Superscalar adalah sebuah arsitektur prosesor yang mampu mengeksekusi lebih dari satu instruksi dalam satu siklus clock.
  - memiliki beberapa execution units yang dapat mengeksekusi instruksi secara paralel.
  - mampu mengambil (fetch) dan mendekode beberapa instruksi secara bersamaan.
- Intruksi umum (arithmetic, load/store, conditional branch) dapat dimulai dan diexecute secara independen
- Dapat di terapkan pada RISC & CISC
- Pada prakteknya hanya diterapkan pada RISC

Why Superscalar?

- Sebagian besar operasi adalah operasi scalar
  - Operasi scalar: operasi yang dilakukan pada nilai tunggal atau sepasang nilai tunggal
- Meningkatkan operasi ini dapat meningkatkan secara keseluruhan

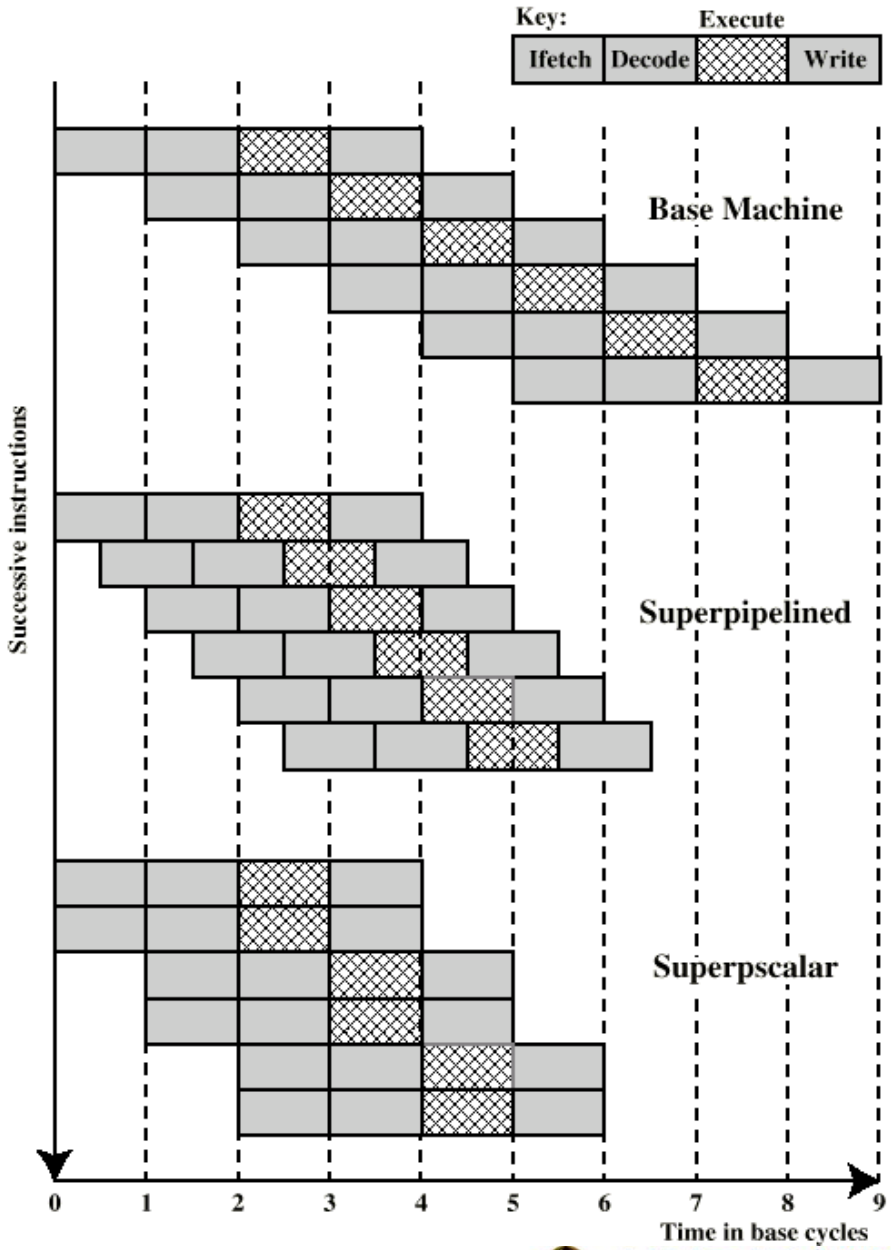
General Superscalar Organization



Superpipelined

- Beberapa tahapan pipeline membutuhkan lebih kecil dari setengah clock cycle.
- Double kecepatan internal clock memperoleh dua tugas per external clock cycle
- Superscalar mengijikan parallel fetch execute

Superscalar v Superpipeline



Limitations *Superscalar*

- Instruction level parallelism
- Compiler based optimisation
- Hardware techniques
- Dibatasi oleh:
  - True data dependency
  - Procedural dependency
  - Resource conflicts
  - Output dependency
  - Antidependency

True Data Dependency

- ADD r1, r2 (r1 := r1+r2;)
- MOVE r3,r1 (r3 := r1;)
- Dapat melakukan fetch and decode intruksi kedua secara parallel dengan pertama.
- Tidak dapat meng-execute intruksi kedua sampai intruksi pertama selesai.

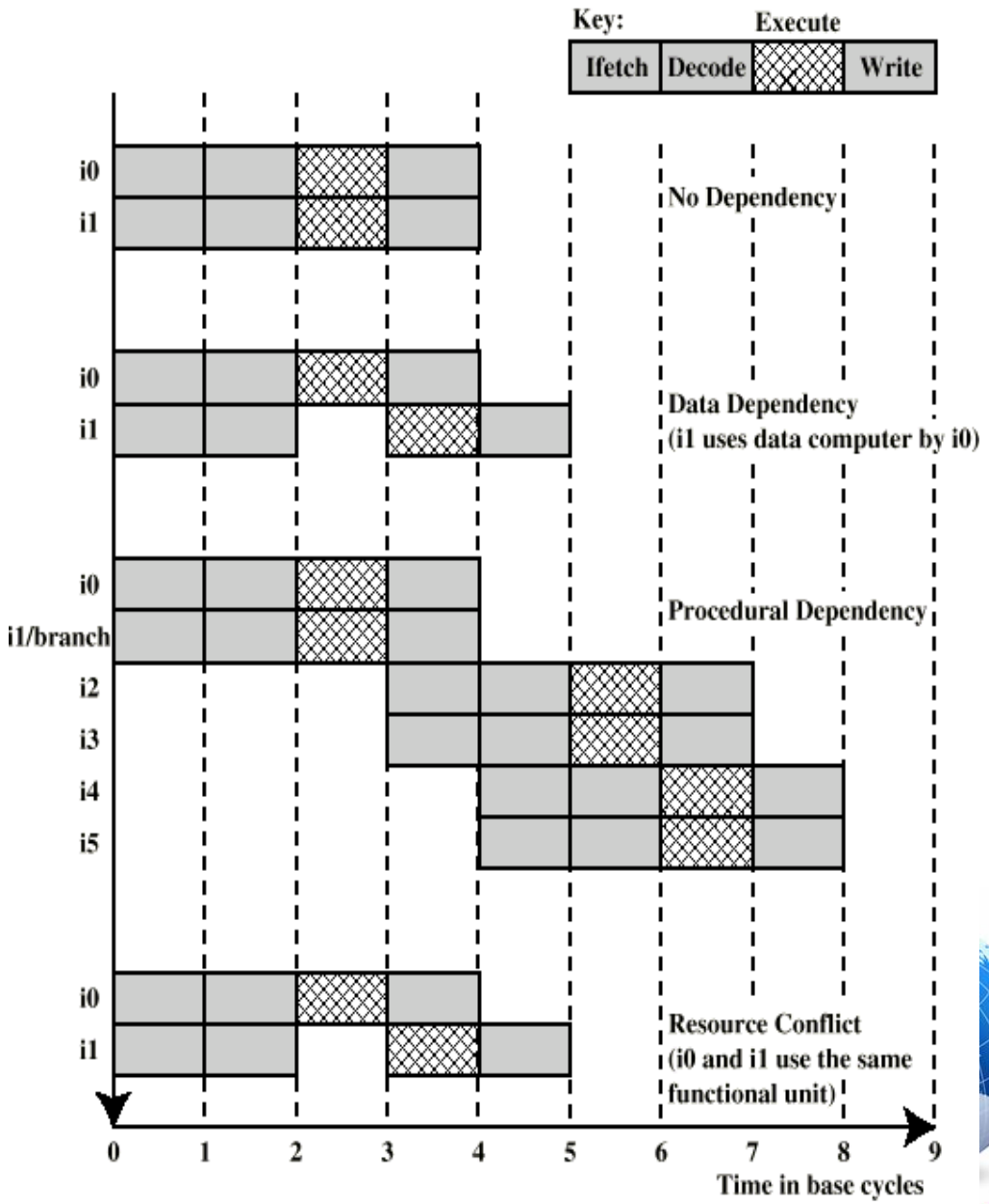
Procedural Dependency

- Tidak dapat execute intruksi setelah sebuah cabang secara parallel dengan intruksi sebelum sebuah cabang
- Selain itu, jika Panjang intruksi tidak tetap, intruksi harus di kodekan untuk mengetahui berapa banyak fetch yang dibutuhkan
- Hal ini mencegah pengambilan secara simultan.

Resource Conflict

- Dua atau lebih intruksi memerlukan akses ke sumber daya yang sama pada waktu bersamaan
  - e.g. two arithmetic instructions
- Can duplicate resources
  - e.g. have two arithmetic units

Effect of Dependencies



Design Issues- *Superscalar*

- Instruction level parallelism
  - Instruksi dalam suatu urutan bersifat independen
  - Eksekusi bisa tumpang tindih
  - Diatur oleh ketergantungan data dan prosedural
- Machine Parallelism
  - Kemampuan untuk memanfaatkan paralelisme tingkat instruksi
  - Diatur oleh jumlah dari parallel pipelines

Instruction Issue Policy

- Urutan pengambilan instruksi
- Urutan di mana instruksi dijalankan
- Urutan instruksi yang mengubah register dan memori

In-Order Issue
Out-of-Order Completion

- Output dependency
  - R3:= R3 + R5; (I1)
  - R4:= R3 + 1; (I2)
  - R3:= R5 + 1; (I3)
  - I2 bergantung pada hasil I1 - data dependency
  - Jika I3 selesai sebelum I1, hasil dari I1 akan salah - output (read-write) dependency.

Antidependency

- Antidependency: sebuah instruksi menulis ke register sebelum instruksi sebelumnya selesai membacanya.
- Write-write dependency
  - R3:=R3 + R5; (I1)
  - R4:=R3 + 1; (I2)
  - R3:=R5 + 1; (I3)
  - R7:=R3 + R4; (I4)
  - I3 tidak dapat diselesaikan sebelum I2 dimulai karena I2 memerlukan nilai di R3 dan I3 mengubah R3

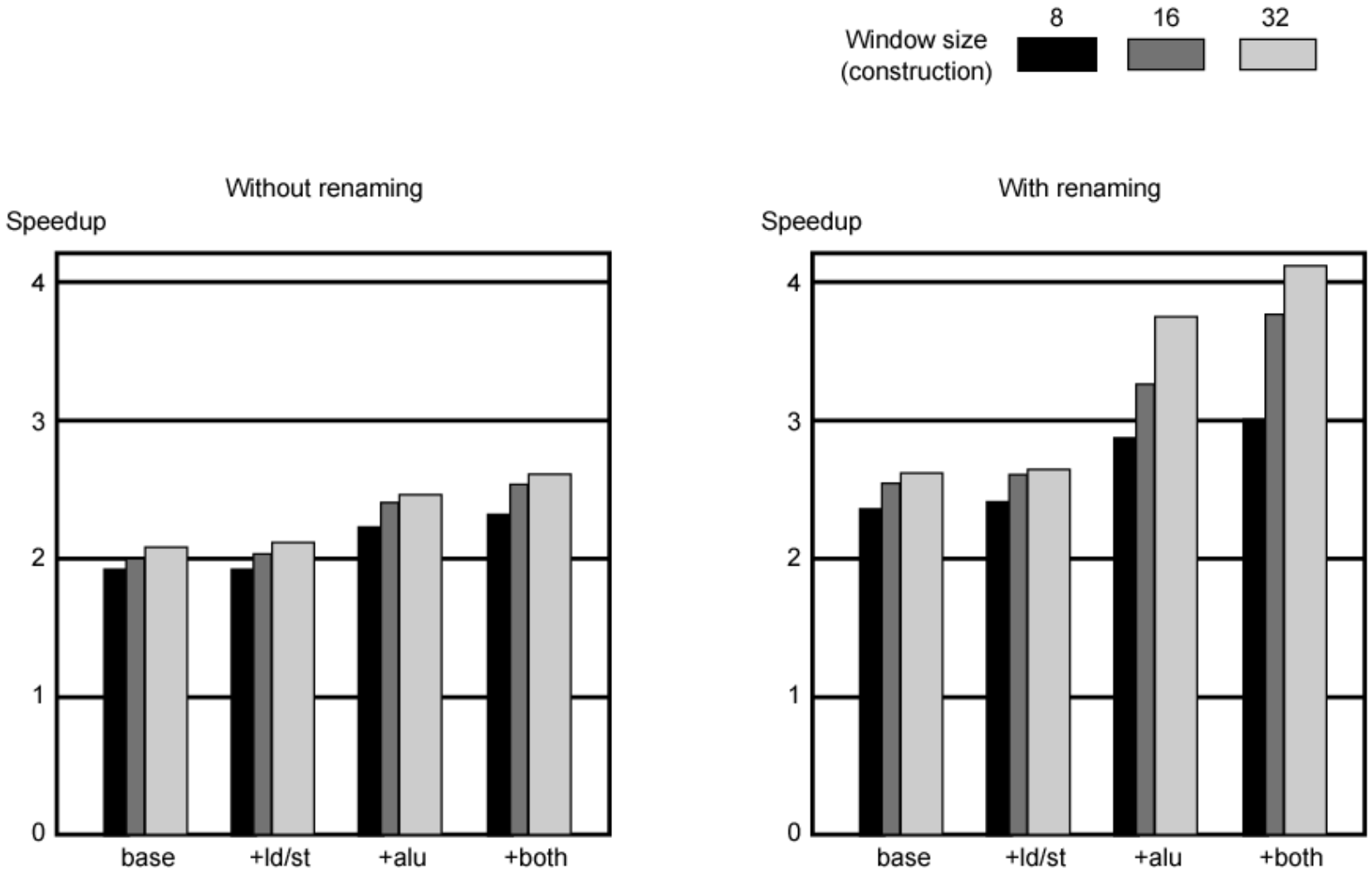
Solusi-Register Renaming

- Register Renaming adalah teknik yang digunakan untuk mengatasi ketergantungan antar-instruksi dan konflik akses register dalam pipelining eksekusi instruksi.
- Output dan antidependensi terjadi karena isi register mungkin tidak mencerminkan urutan yang benar dari program
- Dapat mengakibatkan terhentinya pipeline
- Register dialokasikan secara dinamis
  - yaitu register tidak diberi nama secara spesifik

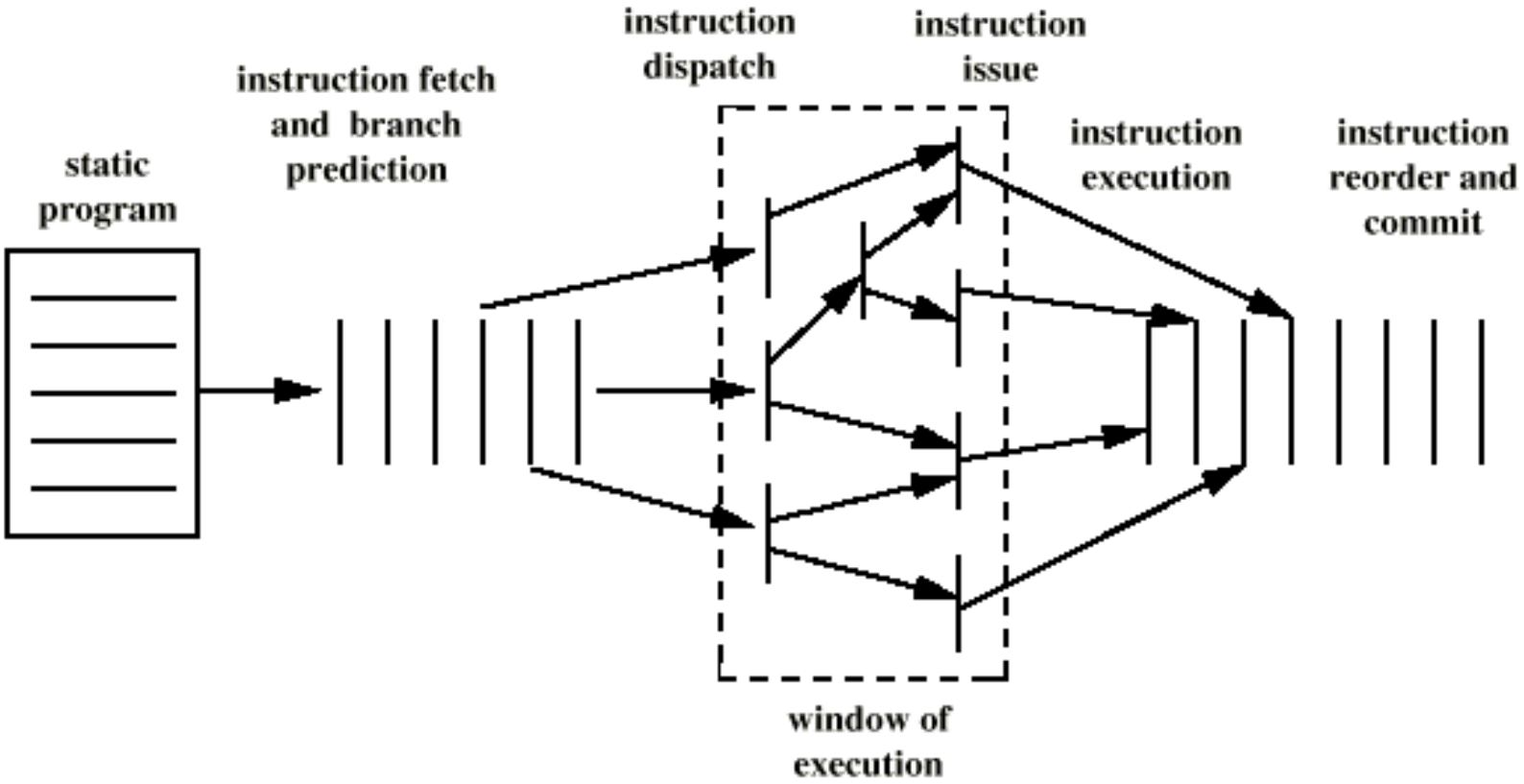
Register Renaming example

- R3b:=R3a + R5a (I1)
- R4b:=R3b + 1 (I2)
- R3c:=R5a + 1 (I3)
- R7b:=R3c + R4b (I4)
- Tanpa subskrip mengacu pada register logis dalam instruksi
- Dengan subskrip, register perangkat keras dialokasikan
- Note R3a R3b R3c

Speedups of Machine Organizations Without Procedural Dependencies



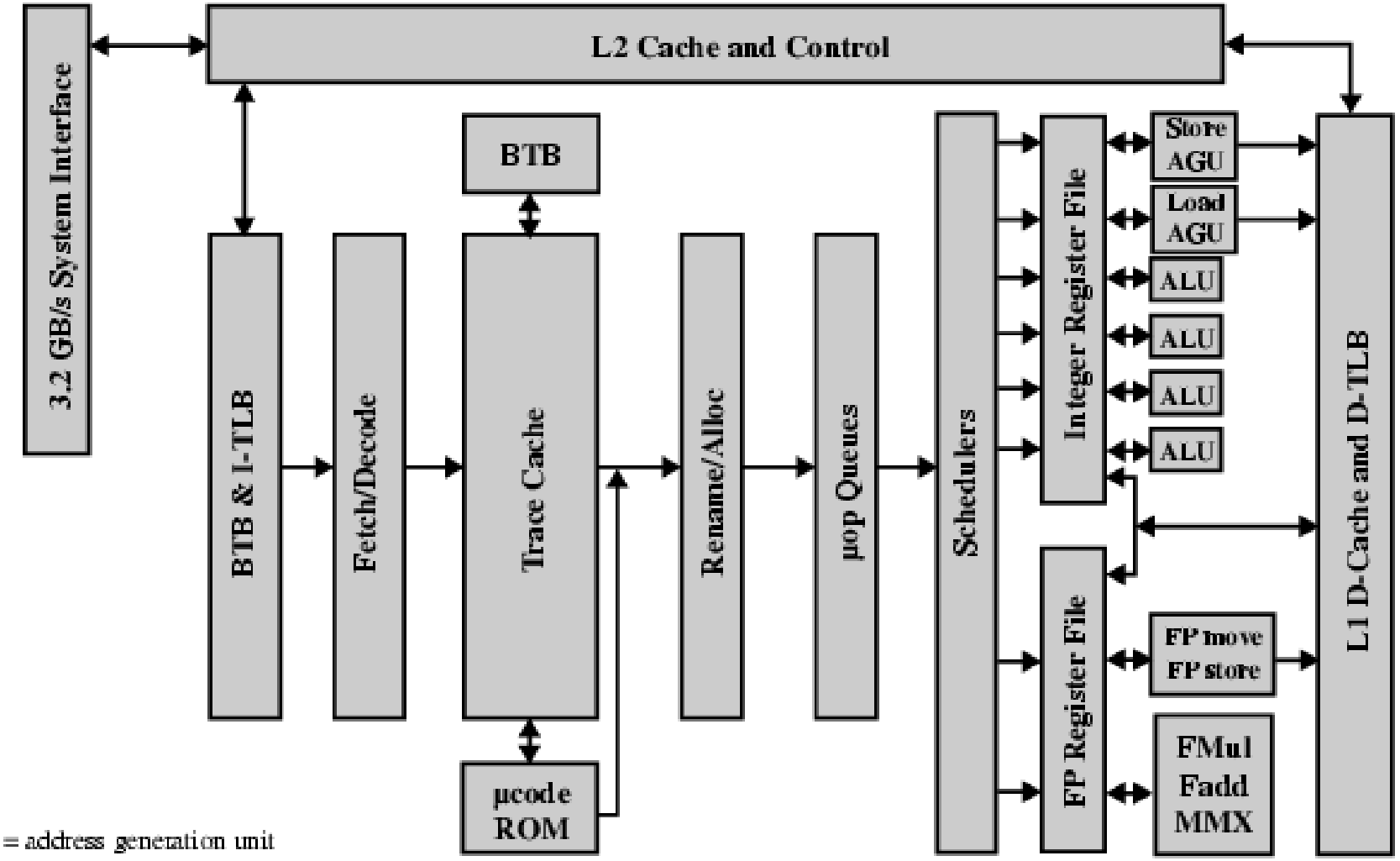
Superscalar Execution



Perkembangan Pentium 4

- 80486 - CISC
- Pentium – beberapa komponen superscalar
  - Dua unit eksekusi bilangan bulat
- Pentium Pro – Superscalar penuh
- Model berikutnya menyempurnakan superscalar

Pentium 4 Block Diagram

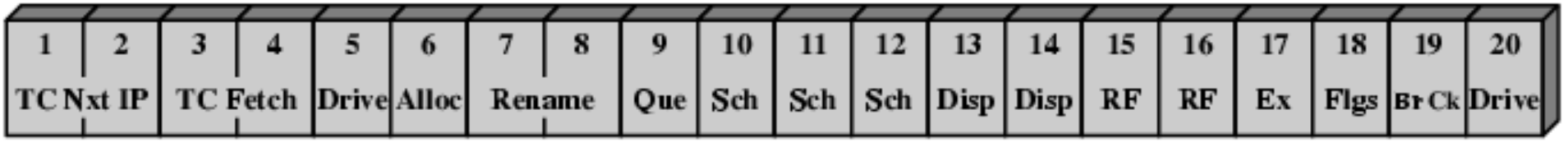


AGU = address generation unit
BTB = branch target buffer
D-TLB = data translation lookaside buffer
I-TLB = instruction translation lookaside buffer

Pentium 4 Operation

- Ambil instruksi dari memori dalam urutan program statis
- Menerjemahkan instruksi menjadi satu atau lebih instruksi RISC dengan panjang tetap (operasi mikro)
- Execute operasi mikro pada pipa superscalar
  - micro-ops dapat executed secara tidak urut
- Komit hasil operasi mikro ke register yang diatur dalam urutan aliran program asliCangkang CISC bagian luar dengan inti RISC bagian dalamPipa inti RISC bagian dalam setidaknya 20 tahap
  - Beberapa operasi mikro memerlukan beberapa tahapan eksekusi
    - Longer pipeline
  - c.f. five stage pipeline on x86 up to Pentium

Pentium 4 Pipeline

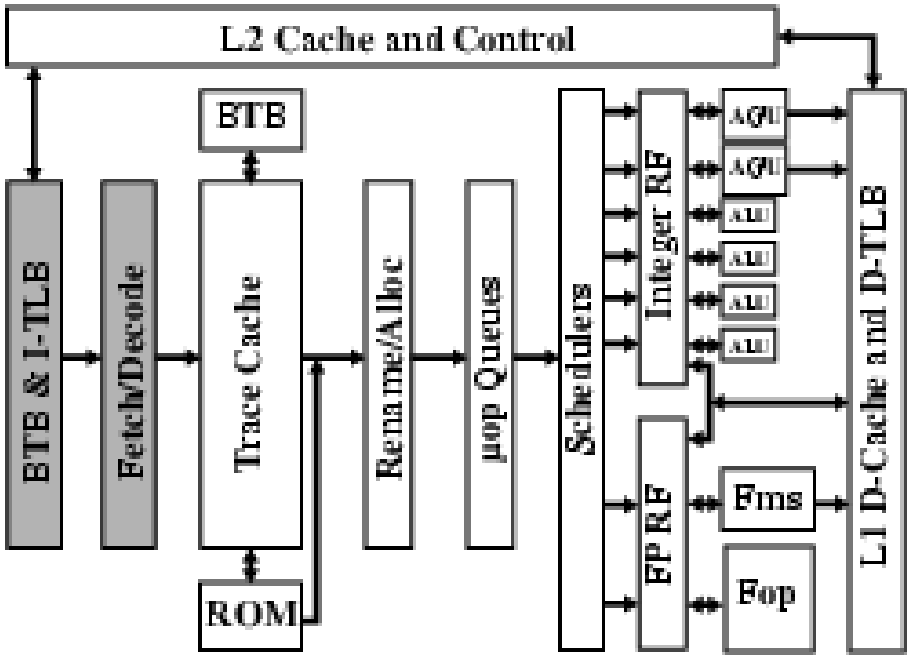


TC Next IP = trace cache next instruction pointer
TC Fetch = trace cache fetch
Alloc = allocate

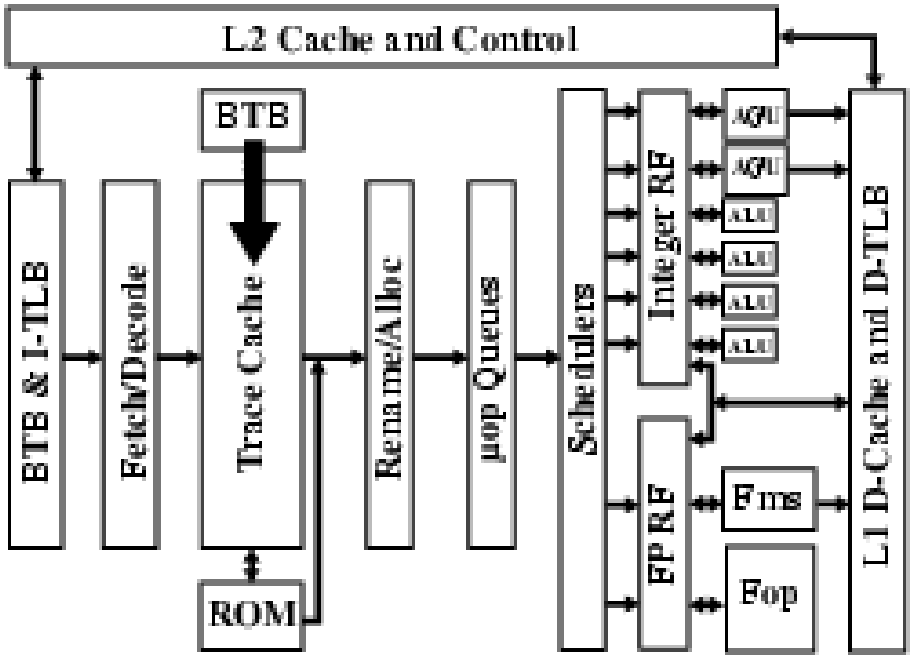
Rename = register renaming
Que = micro-op queuing
Sch = micro-op scheduling
Disp = Dispatch

RF = register file
Ex = execute
Flgs = flags
Br Ck = branch check

Pentium 4 Pipeline Operation (1)

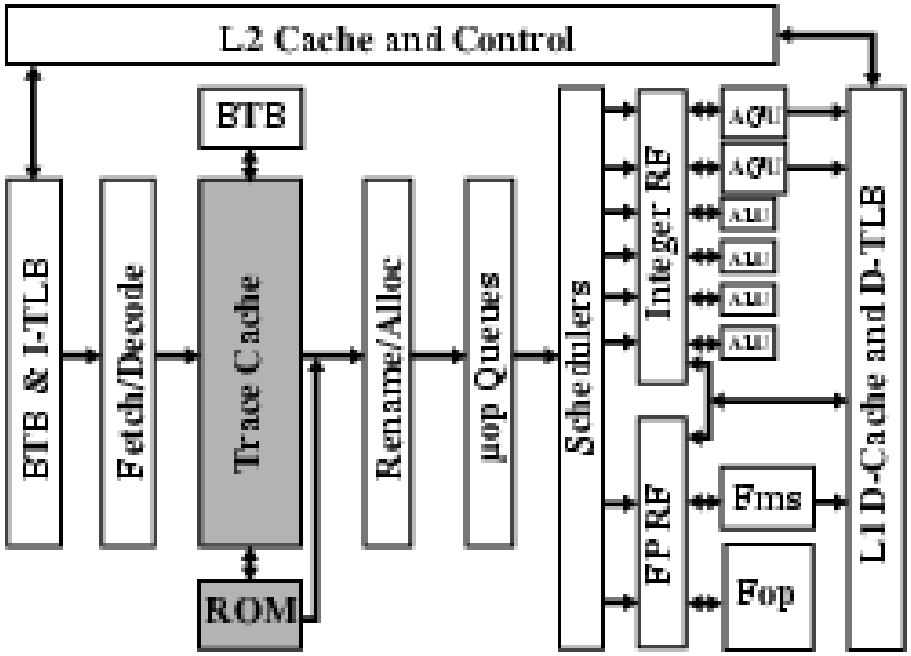


(a) Generation of micro-ops

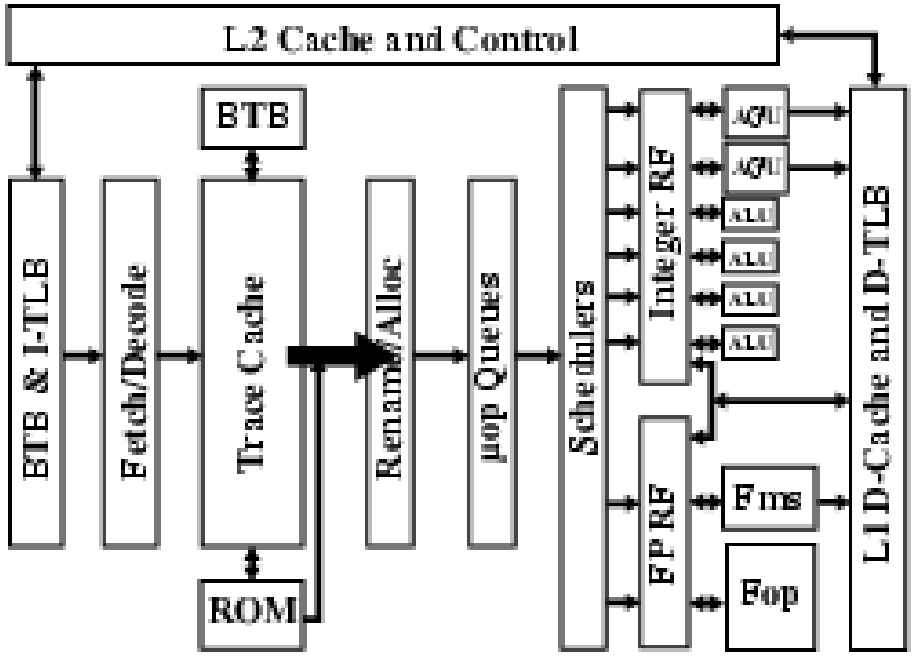


(b) Trace cache next instruction pointer

Pentium 4 Pipeline Operation (2)

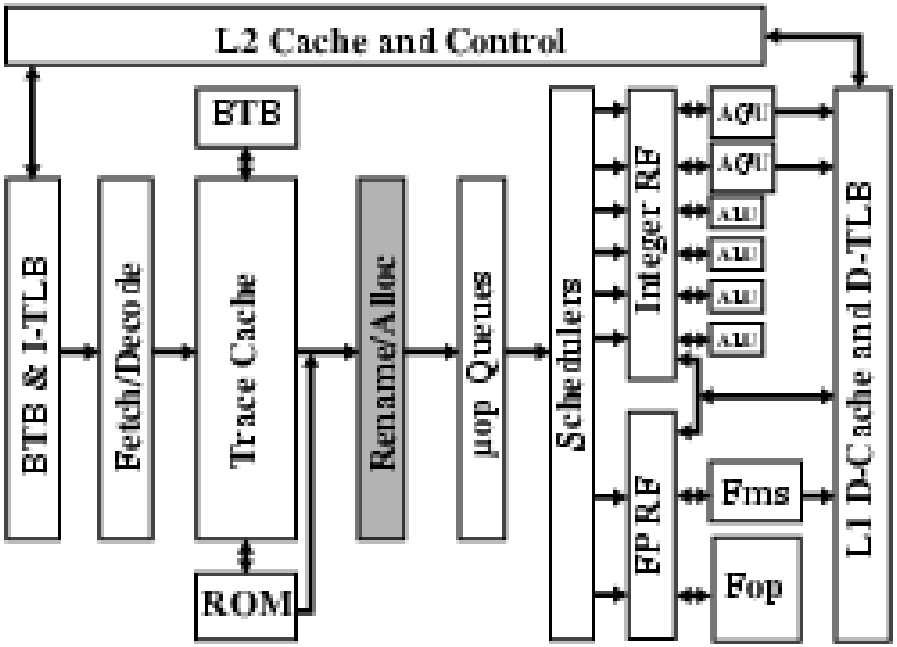


(c) Trace cache fetch

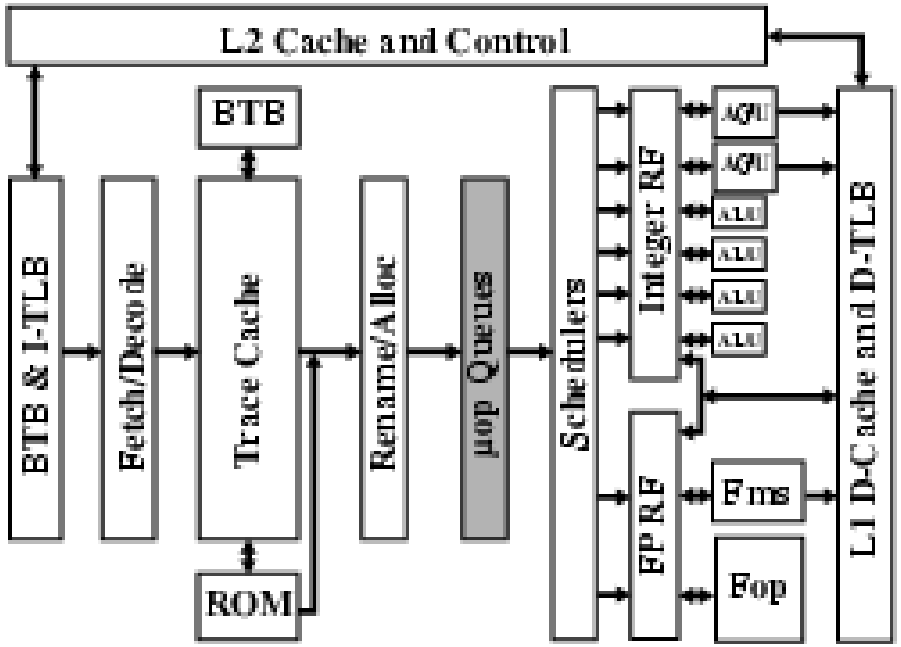


(d) Drive

Pentium 4 Pipeline Operation (3)

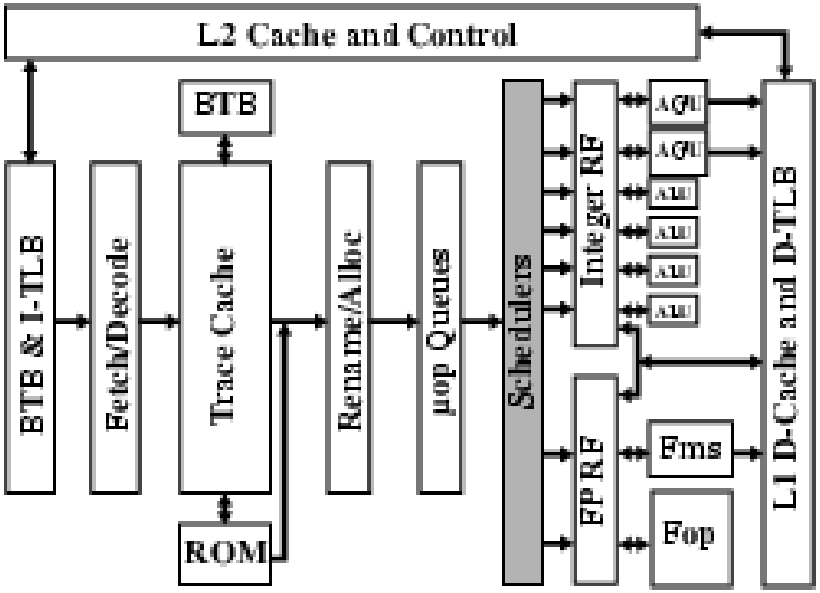


(e) Allocate; Register renaming

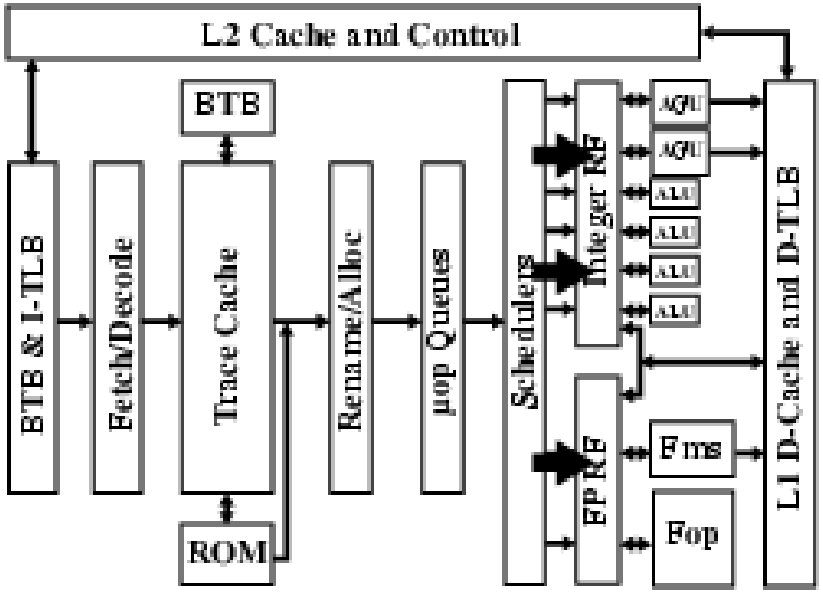


(f) Micro-op queuing

Pentium 4 Pipeline Operation (4)

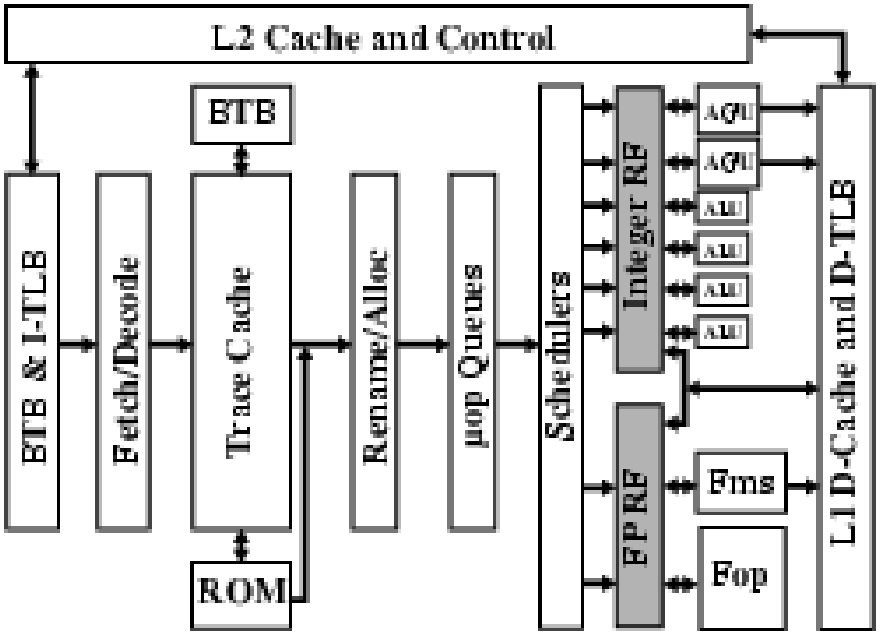


(g) Micro-op scheduling

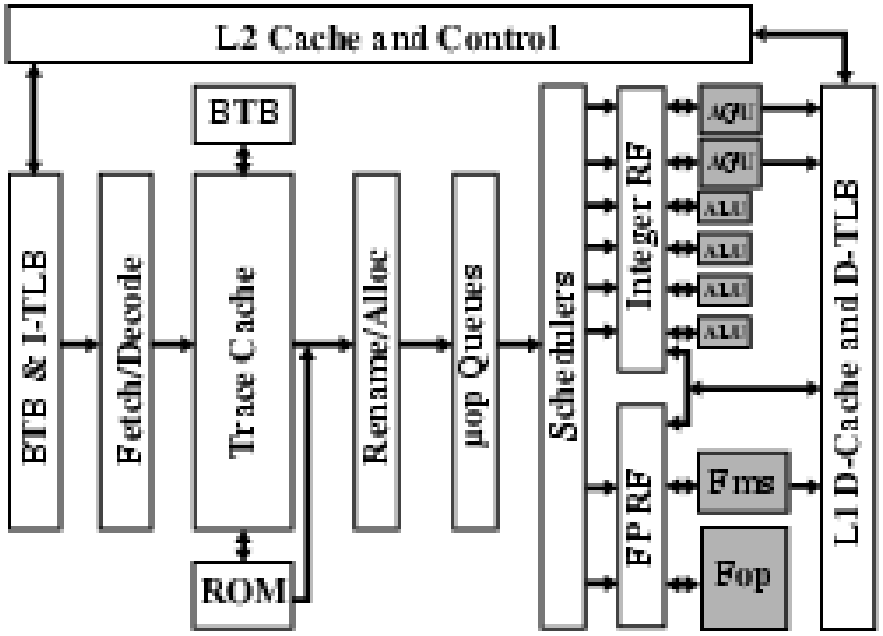


(h) Dispatch

Pentium 4 Pipeline Operation (5)

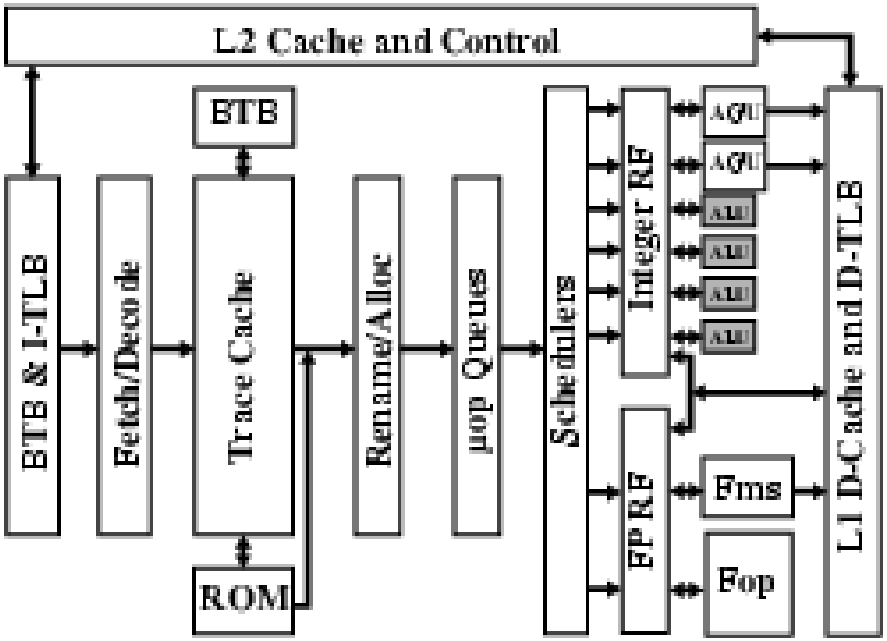


(i) Register file

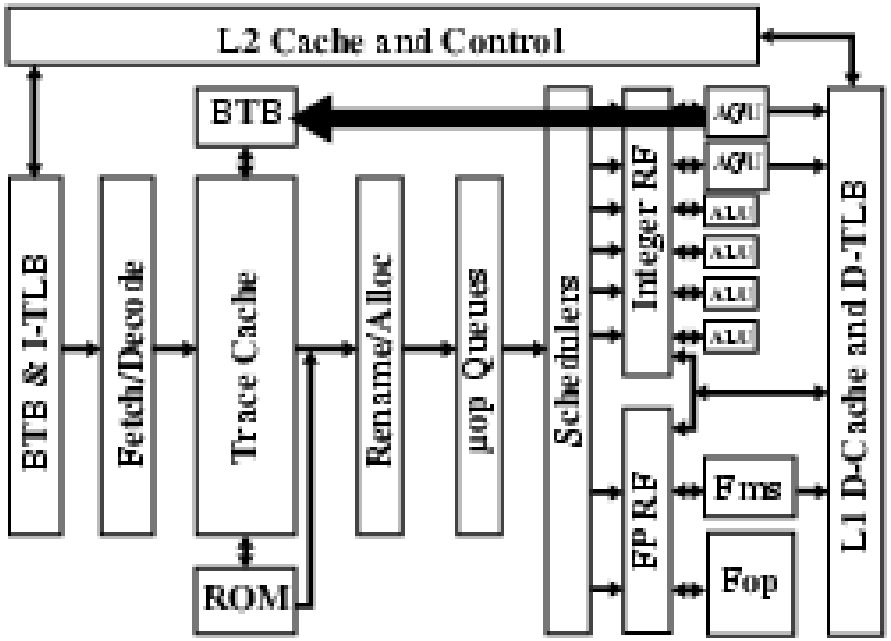


(j) Execute; flags

Pentium 4 Pipeline Operation (6)



(k) Branch check

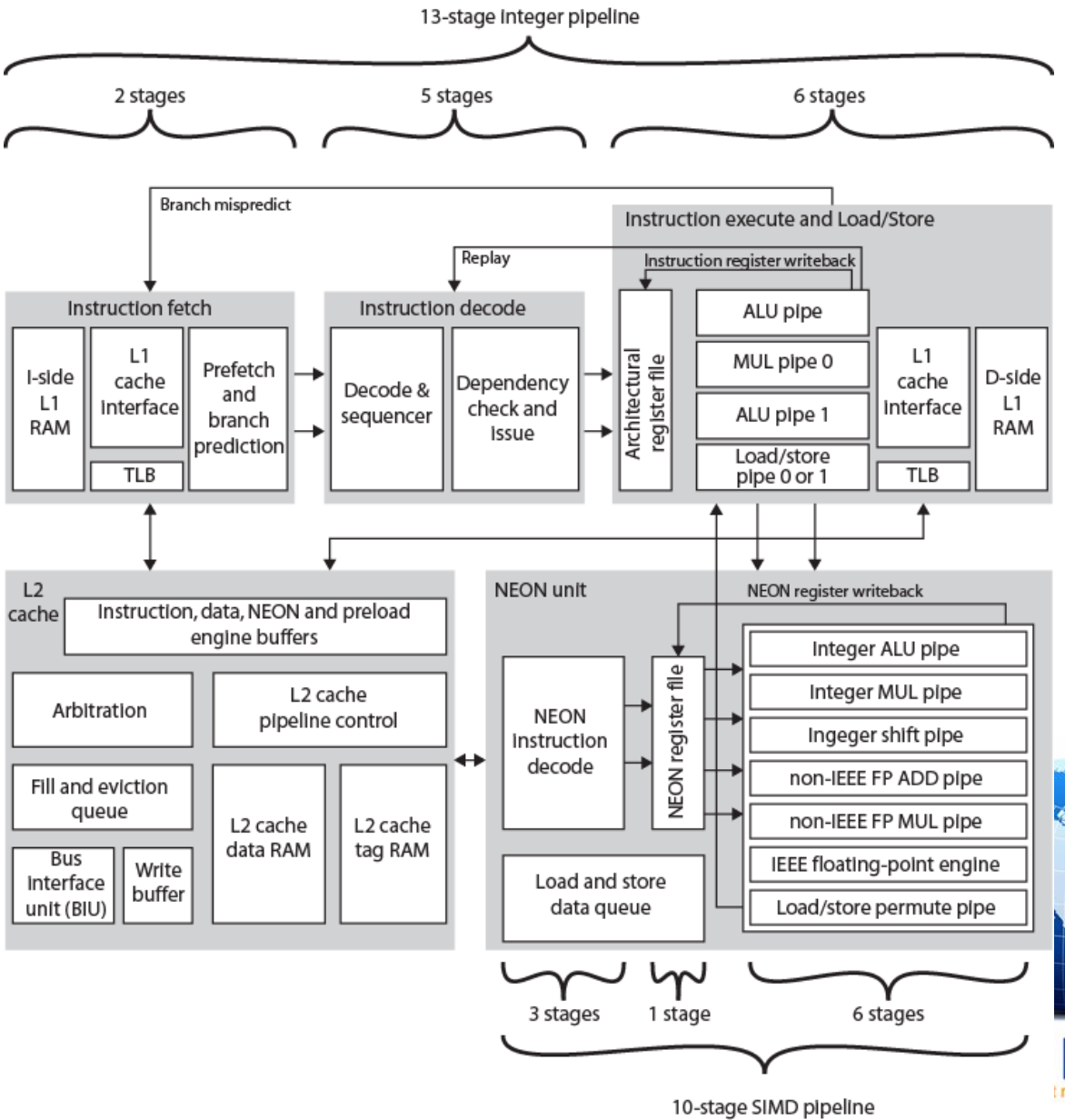


(l) Branch check result

ARM CORTEX-A8

- ARM mengacu pada Cortex-A8 sebagai application processors
- Embedded processor yang menjalankan complex operating system
  - Wireless, consumer and imaging applications
  - Mobile phones, set-top boxes, gaming consoles automotive navigation/entertainment systems
- Three functional units
- Dual, in-order-issue, 13-stage pipeline
  - Pertahankan daya yang dibutuhkan seminimal mungkin
  - Masalah yang tidak berurutan memerlukan logika ekstra yang memakan daya ekstra
- Separate SIMD (single-instruction-multiple-data) unit
  - 10-stage pipeline

ARM Cortex-A8 Block Diagram



Instruction Fetch Unit

- Predicts instruction stream
- Fetches instructions from the L1 instruction cache
  - Up to four instructions per cycle
- Into buffer for decode pipeline
- Fetch unit includes L1 instruction cache
- Speculative instruction fetches
- Branch or exceptional instruction cause pipeline flush
- Stages:
- F0 address generation unit generates virtual address
  - Normally next sequentially
  - Can also be branch target address
- F1 Used to fetch instructions from L1 instruction cache
  - In parallel fetch address used to access branch prediction arrays
- F3 Instruction data are placed in instruction queue
  - If branch prediction, new target address sent to address generation unit
- Two-level global history branch predictor
  - Branch Target Buffer (BTB) and Global History Buffer (GHB)
- Return stack to predict subroutine return addresses
- Can fetch and queue up to 12 instructions
- Issues instructions two at a time

Instruction Decode Unit

- Decodes and sequences all instructions
- Dual pipeline structure, *pipe0* and *pipe1*
  - Two instructions can progress at a time
  - Pipe0 contains older instruction in program order
  - If instruction in pipe0 cannot issue, pipe1 will not issue
- Instructions progress in order
- Results written back to register file at end of execution pipeline
  - Prevents WAR hazards
  - Keeps tracking of WAW hazards and recovery from flush conditions straightforward
  - Main concern of decode pipeline is prevention of RAW hazards

Instruction Processing Stages

- D0 Thumb instructions decompressed and preliminary decode is performed
- D1 Instruction decode is completed
- D2 Write instruction to and read instructions from pending/replay queue
- D3 Contains the instruction scheduling logic
  - Scoreboard predicts register availability using static scheduling
  - Hazard checking
- D4 Final decode for control signals for integer execute load/store units

Integer Execution Unit

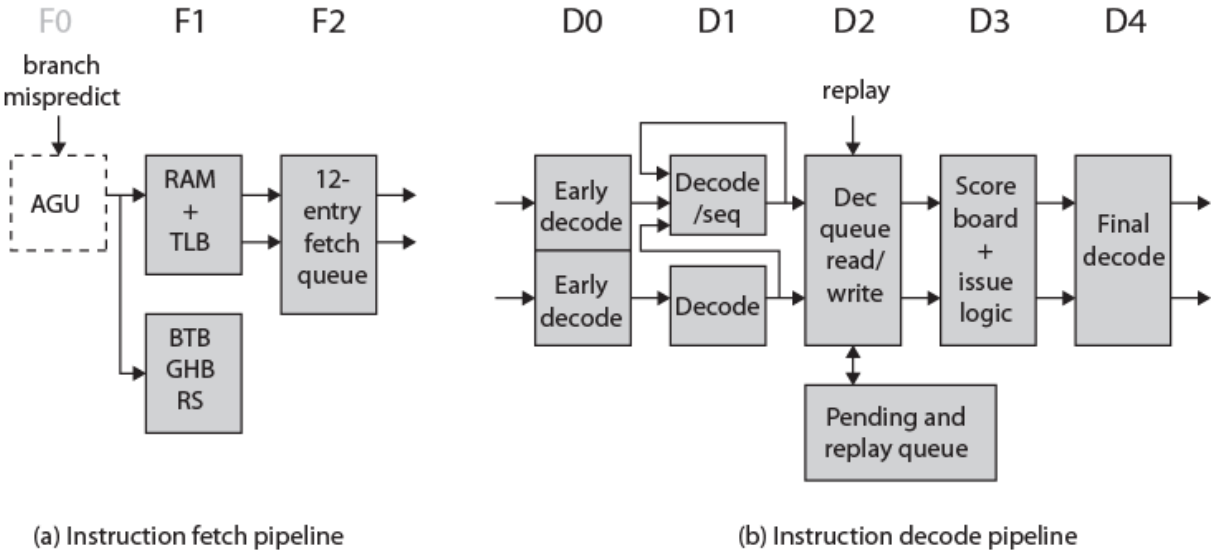
- Two symmetric (ALU) pipelines, an address generator for load and store instructions, and multiply pipeline
- Pipeline stages:
- E0 Access register file
  - Up to six registers for two instructions
- E1 Barrel shifter if needed.
- E2 ALU function
- E3 If needed, completes saturation arithmetic
- E4 Change in control flow prioritized and processed
- E5 Results written back to register file

- Multiply unit instructions routed to pipe0
  - Performed in stages E1 through E3
  - Multiply accumulate operation in E4

Load/store pipeline

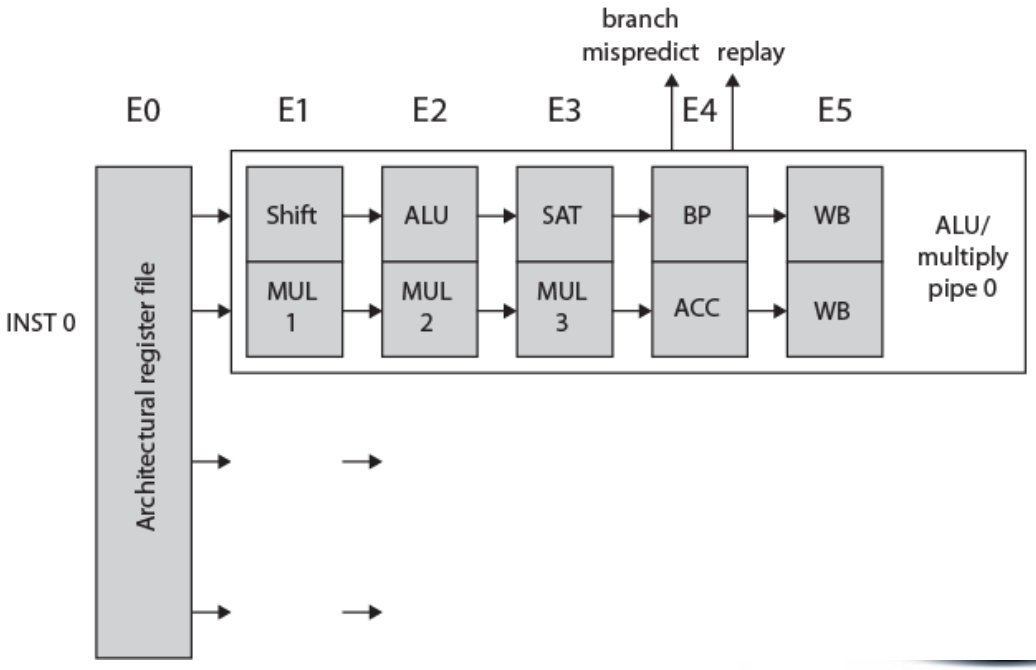
- Parallel to integer pipeline
- E1 Memory address generated from base and index register
- E2 address applied to cache arrays
- E3 load, data returned and formatted
- E3 store, data are formatted and ready to be written to cache
- E4 Updates L2 cache, if required
- E5 Results are written to register file

ARM Cortex-A8 Integer Pipeline



(a) Instruction fetch pipeline

(b) Instruction decode pipeline

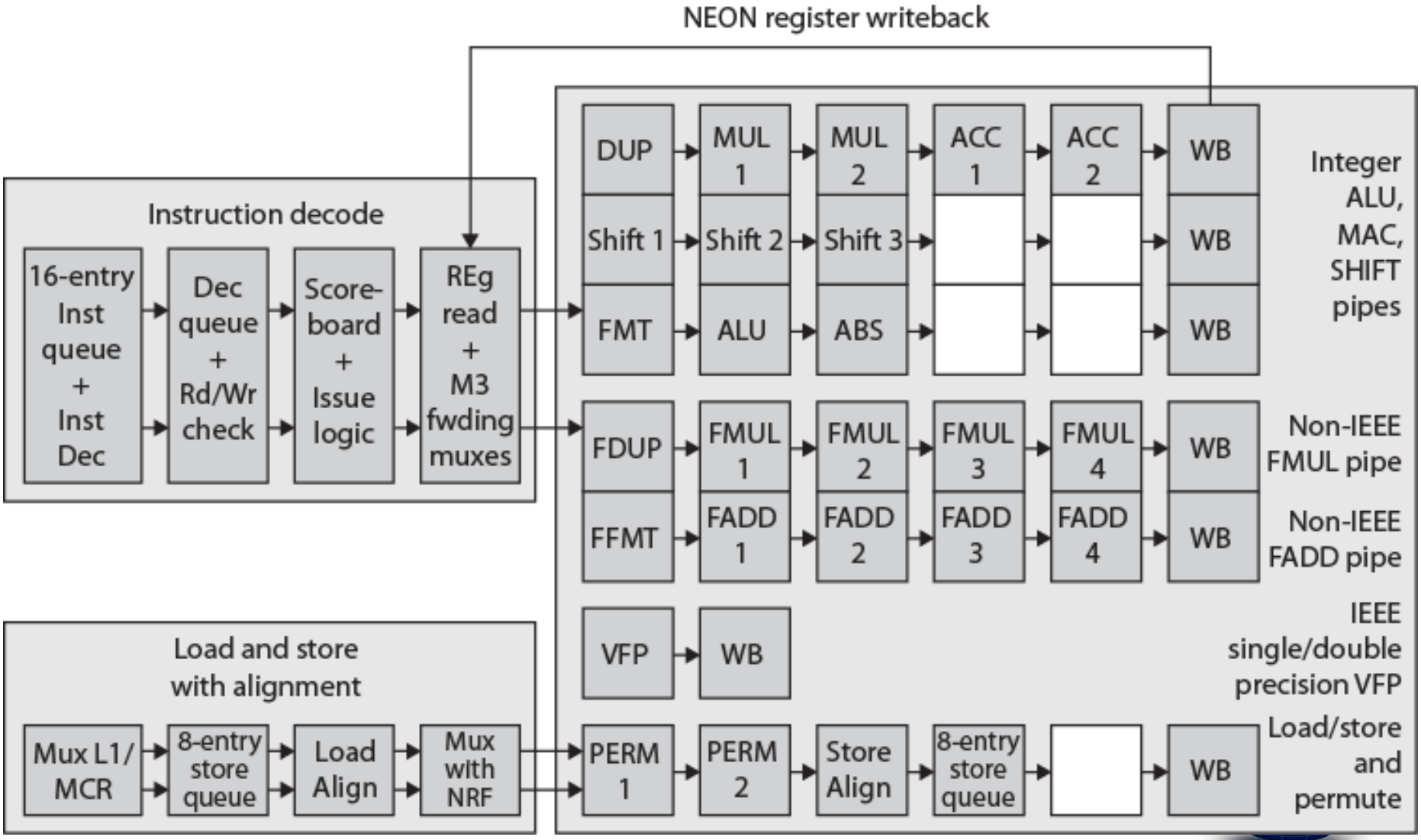


SIMD and Floating-Point Pipeline

- SIMD and floating-point instructions pass through integer pipeline
- Processed in separate 10-stage pipeline
  - NEON unit
  - Handles packed SIMD instructions
  - Provides two types of floating-point support

- If implemented, vector floating-point (VFP) coprocessor performs IEEE 754 floating-point operations
  - If not, separate multiply and add pipelines implement floating-point operations

ARM Cortex-A8 NEON & Floating Point Pipeline



Tugas

- Jelaskan perkembangan/Evolusi arsitektur processor Pentium dan ARM Cortex?
- Jelaskan perbedaan keduanya?
- Apa kelebihan dan kekurangan keduanya?

TERIMAKASIH

UNDIP | UNIVERSITAS DIPONEGORO
becomes an excellent research university