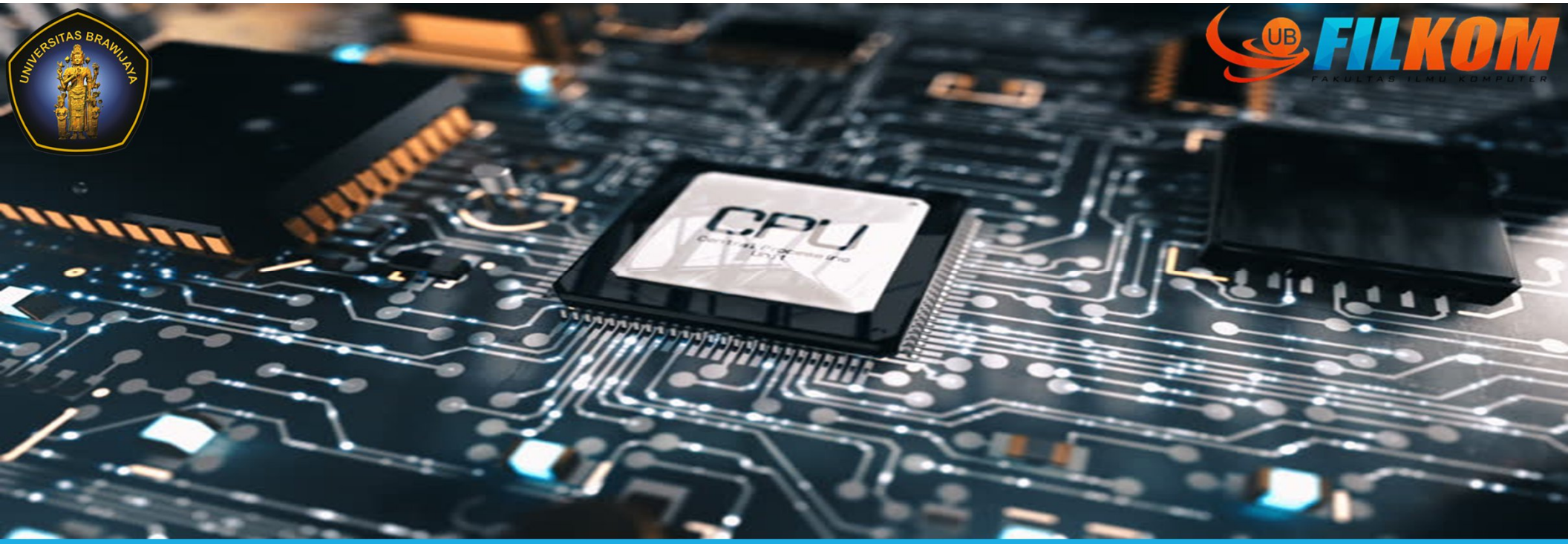


Arsitektur dan Organisasi Komputer

COM 60011

Topik #13 – Instruction Level Parallelism (ILP) Dan Superscalar

Tujuan

CPMK : Mahasiswa mampu menjabarkan arsitektur dan organisasi dari prosesor (CPU) pada suatu komputer.

Sub CPMK : Mahasiswa mampu menjelaskan tentang teknologi pipeline serta pengembangannya termasuk superpipeline dan superscalar yang digunakan pada komputer.

Materi Terkait:

- **Desain isu**
- **Superscalar**
- **Superpipeline**
- **Limitasi pada x86 dan ARM**

Instruction Level Parallelism (ILP)
Dan Superscalar

Pendahuluan

Processor komputer dibagi menjadi tiga kategori utama, yaitu scalar, superscalar dan vector.

Processor scalar merupakan tipe processor yang paling umum dikenali dan digunakan oleh mayoritas pengguna komputer.

Processor ini menerima satu perintah dalam satu waktu dan mengeksekusinya secara berurutan atau sesuai dengan prioritasnya.

Mengapa Prosesor Superscalar 1

Processor superscalar merupakan sebuah tipe processor yang dapat mengeksekusi beberapa set instruksi sekaligus dalam waktu yang sama.

Tipe processor ini dapat terdiri dari beberapa sub-unit yang bertugas untuk mengontrol tipe fungsi-fungsi dasar tertentu.

Sementara processor lain juga memiliki unit ini, processor superscalar dapat mengirimkan informasi secara langsung pada unit-unit tersebut untuk diproses sementara processor utama melakukan pekerjaan lainnya.

Processor superscalar merupakan titik tengah dari tiga tipe processor utama.

Superscalar ini mampu menjlankan Instruction Level Parallelism dengan satu prosesor.Superscalar dapat diaplikasikan di RISC dan CISC, tapi pada umumnya RISC.

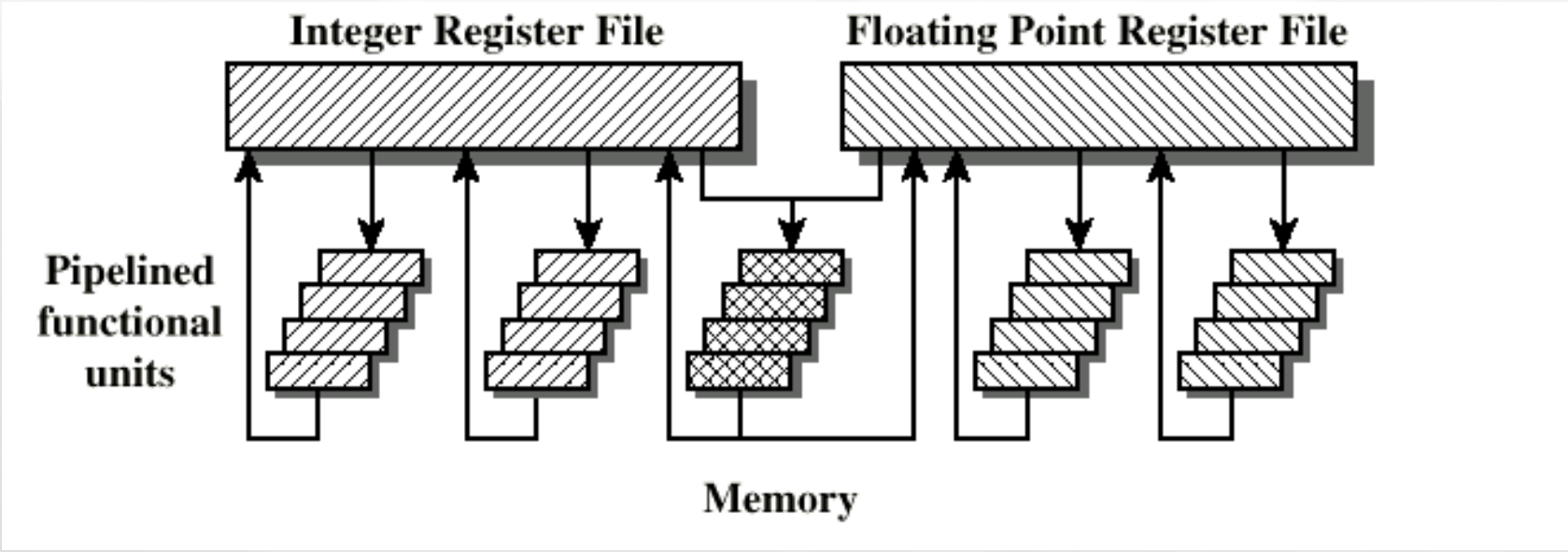
Mengapa Prosesor Superscalar 2

Superscalar (superskalar) adalah arsitektur prosessor yang memungkinkan eksekusi yang bersamaan (parallel) dari instruksi yang banyak pada tahap pipeline yang sama sebaik tahap pipeline yang lain.

Standar pipeline yang digunakan adalah untuk pengolahan bilangan matematika integer (bilangan bulat, bilangan yang tidak memiliki pecahan), kebanyakan CPU juga memiliki kemampuan untuk pengolahan untuk data floating point (bilangan berkoma).

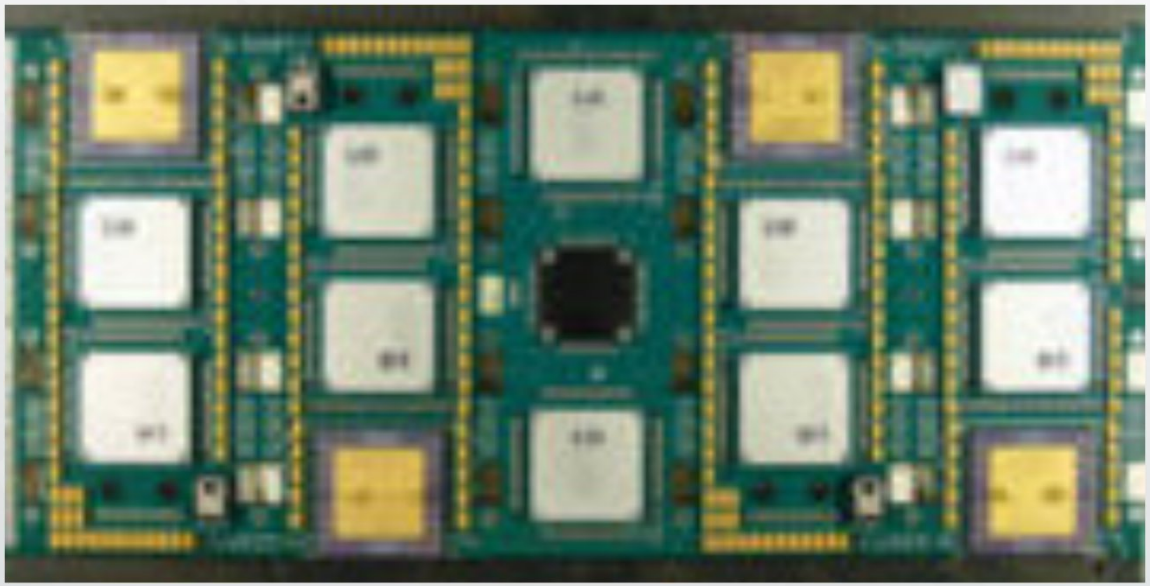
Pipeline yang mengolah integer dapat juga digunakan untuk mengolah data bertipe floating point ini, namun untuk aplikasi tertentu, terutama untuk aplikasi keperluan ilmiah CPU yang memiliki kemampuan pengolahan floating point dapat meningkatkan kecepatan prosesnya secara dramatis.

Organisasi Superscalar secara umum



Contoh Processor Superscalar

Processor board of a CRAY T3e parallel computer with four superscalar Alpha processors



Superpipeline

Teknologi pipeline yang digunakan pada komputer bertujuan untuk meningkatkan kinerja dari komputer.

Secara sederhana, pipeline adalah suatu cara yang digunakan untuk melakukan sejumlah kerja secara bersamaan tetapi dalam tahap yang berbeda yang dialirkan secara kontiniu pada unit pemrosesan

Teknik pipeline ini dapat diterapkan pada berbagai tingkatan dalam sistem komputer.

Bisa pada level yang tinggi, misalnya program aplikasi, sampai pada tingkat yang rendah, seperti pada instruksi yang dijalankan oleh microprocessor.

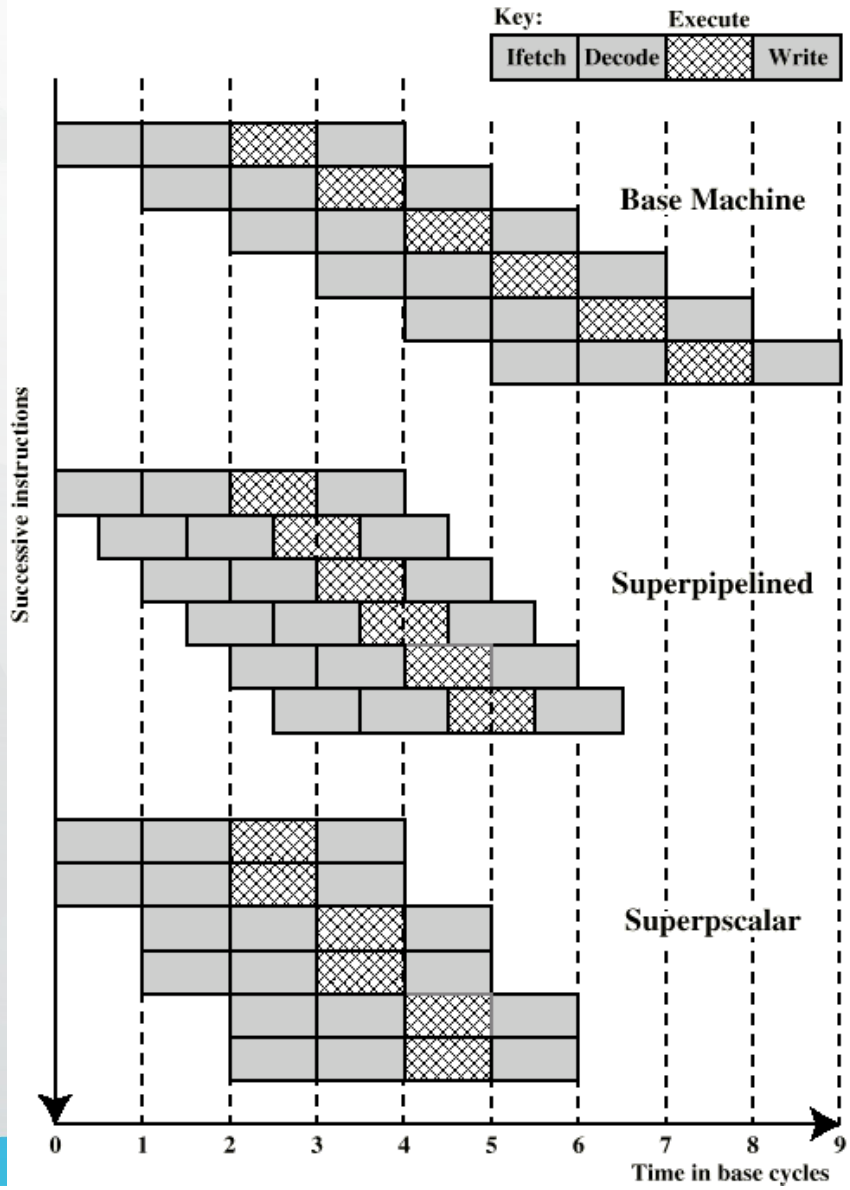
Teknik Superpipeline

Teknik pipeline yang diterapkan pada microprocessor, dapat dikatakan sebuah arsitektur khusus.Ada perbedaan khusus antara model microprocessor yang tidak menggunakan arsitektur pipeline dengan microprocessor yang menerapkan teknik ini.

Pada microprocessor yang tidak menggunakan pipeline, satu instruksi dilakukan sampai selesai, baru instruksi berikutnya dapat dilaksanakan.

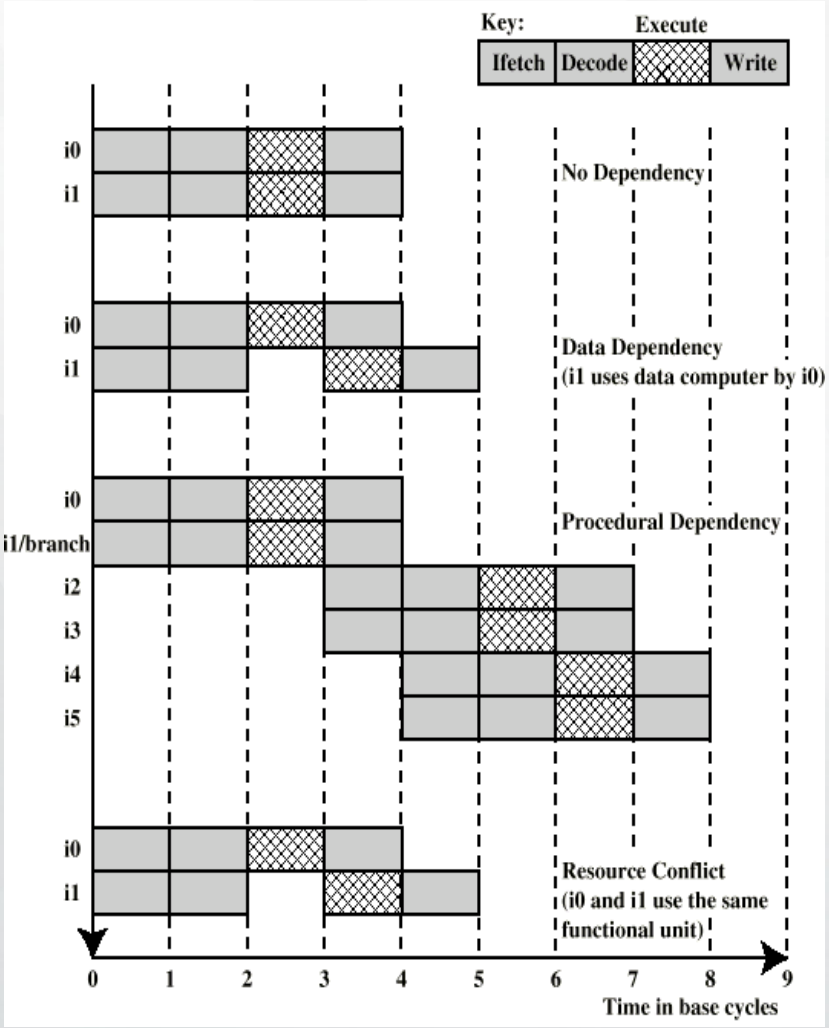
Sedangkan dalam microprocessor yang menggunakan teknik pipeline, ketika satu instruksi sedangkan diproses, maka instruksi yang berikutnya juga dapat diproses dalam waktu yang bersamaan.

Tetapi, instruksi yang diproses secara bersamaan ini, ada dalam tahap proses yang berbeda.



Superscalar vs Superpipeline

Limitations



- True data dependency
- Procedural dependency
- Resource conflicts
- Output dependency
- Antidependency

Design Issues

Instruction level parallelism

- Instruksi dalam urutan bersifat independen
- Eksekusi bisa tumpang tindih
- Diatur oleh ketergantungan data dan prosedural

Machine Parallelism

- Kemampuan untuk memanfaatkan paralelisme tingkat instruksi
- Diatur oleh jumlah pipelines yang paralel

In-Order Issue In-Order Completion (Diagram)

Decode		Execute			Write		Cycle
I1	I2						1
I3	I4	I1	I2				2
I3	I4	I1					3
	I4			I3	I1	I2	4
I5	I6			I4			5
	I6		I5		I3	I4	6
			I6				7
					I5	I6	8

Out-of-Order Issue
Out-of-Order Completion

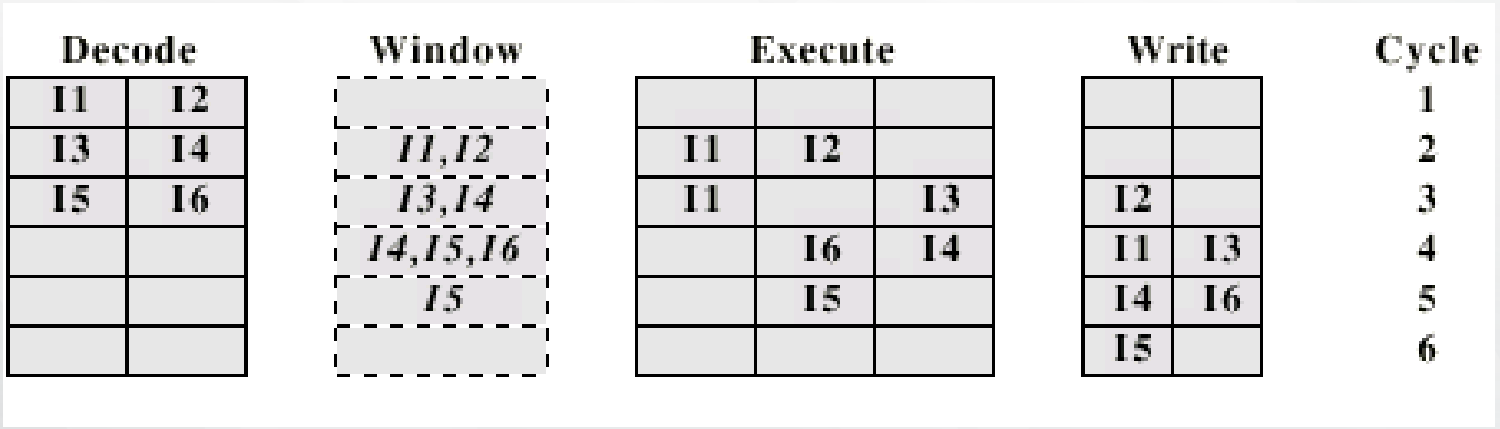
Decouple decode pipeline from execution pipeline

Can continue to fetch and decode until this pipeline is full

When a functional unit becomes available an instruction can be executed

Since instructions have been decoded, processor can look ahead

Out-of-Order Issue Out-of-Order Completion (Diagram)



Decode		Window	Execute			Write		Cycle
I1	I2							1
I3	I4	*I1,I2*	I1	I2				2
I5	I6	*I3,I4*	I1		I3	I2		3
		I4,I5,I6		I6	I4	I1	I3	4
		I5		I5		I4	I6	5
						I5		6

Reorder Buffer

Temporary storage for results

Commit to register file in program order

Register Renaming

- **Output and antidependencies occur because register contents may not reflect the correct ordering from the program**
- **May result in a pipeline stall**
- **Registers allocated dynamically**
  - i.e. registers are not specifically named

Register Renaming example

R3b:=R3a + R5a (I1)

R4b:=R3b + 1 (I2)

R3c:=R5a + 1 (I3)

R7b:=R3c + R4b (I4)

Without subscript refers to logical register in instruction

With subscript is hardware register allocated

Note R3a R3b R3c

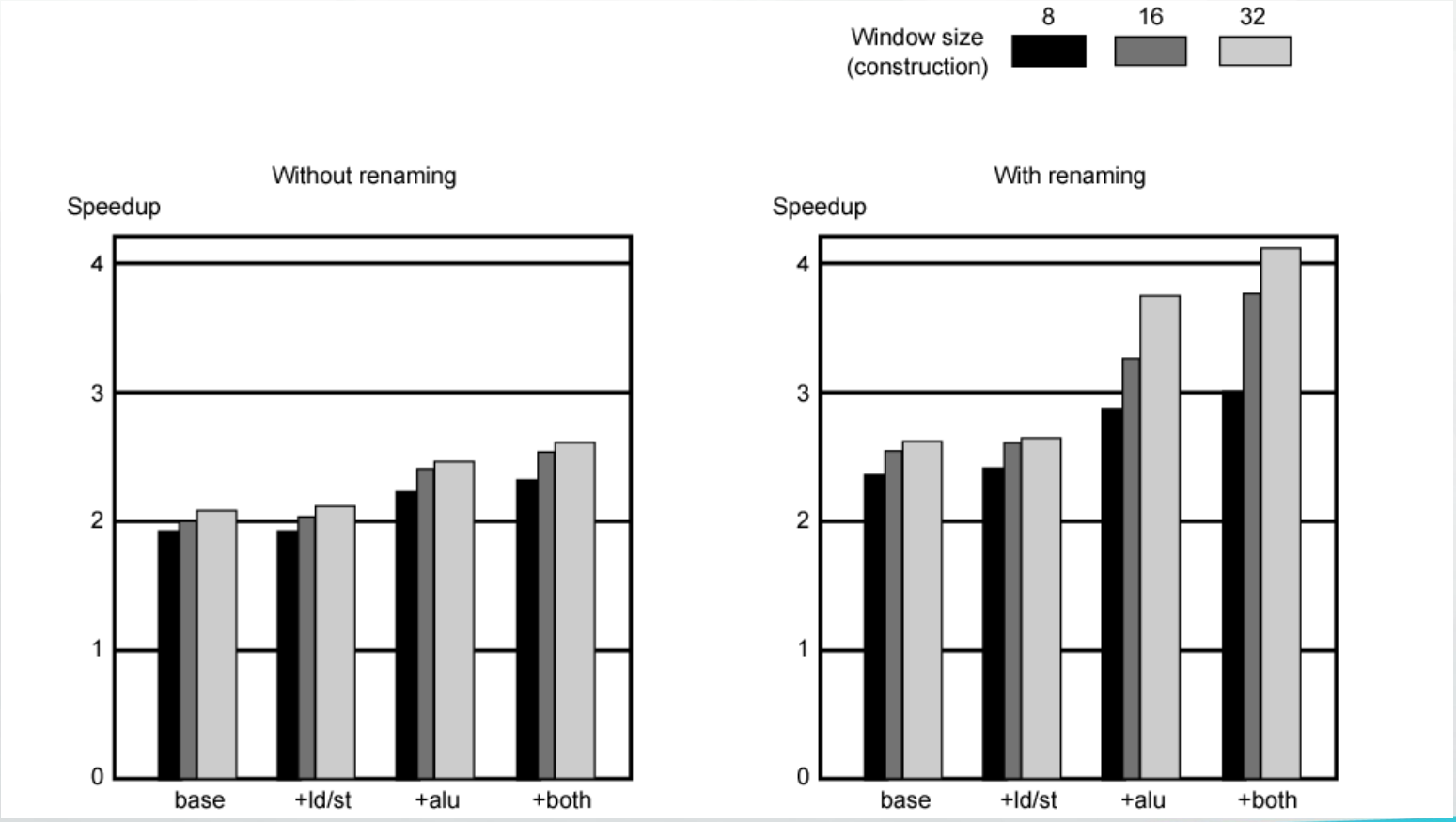
Alternative: Scoreboarding

- Bookkeeping technique
- Allow instruction execution whenever not dependent on previous instructions and no structural hazards

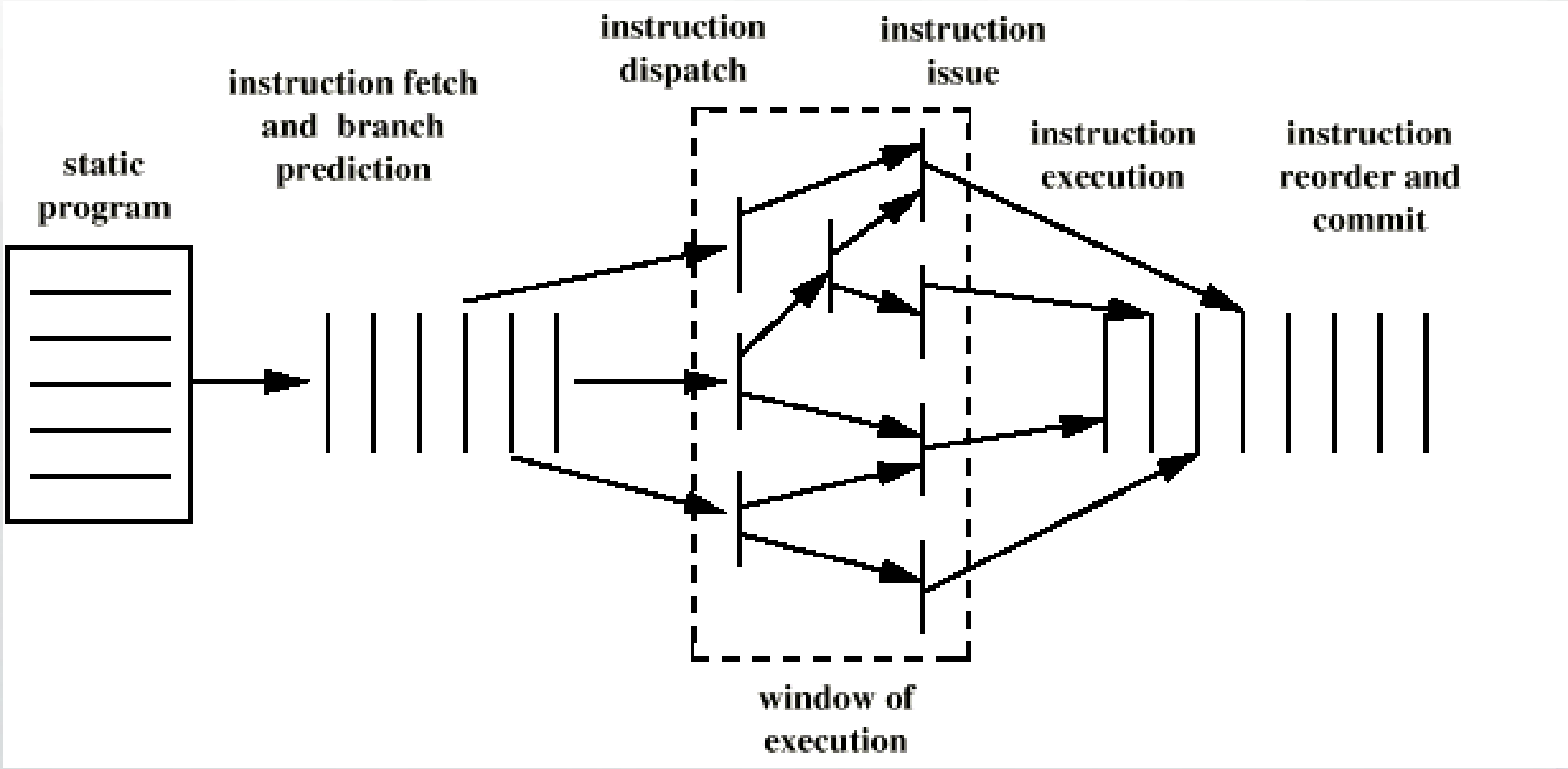
Machine Parallelism

- Duplication of Resources
- Out of order issue
- Renaming
- Not worth duplication functions without register renaming
- Need instruction window large enough (more than 8)

Speedups of Machine Organizations Without Procedural Dependencies



Superscalar Execution



Implementasi superscalar

Proses fetch dari beberapa instruksi secara bersamaan.

Logika untuk menentukan ketergantungan sebenarnya yang meliputi nilai register.

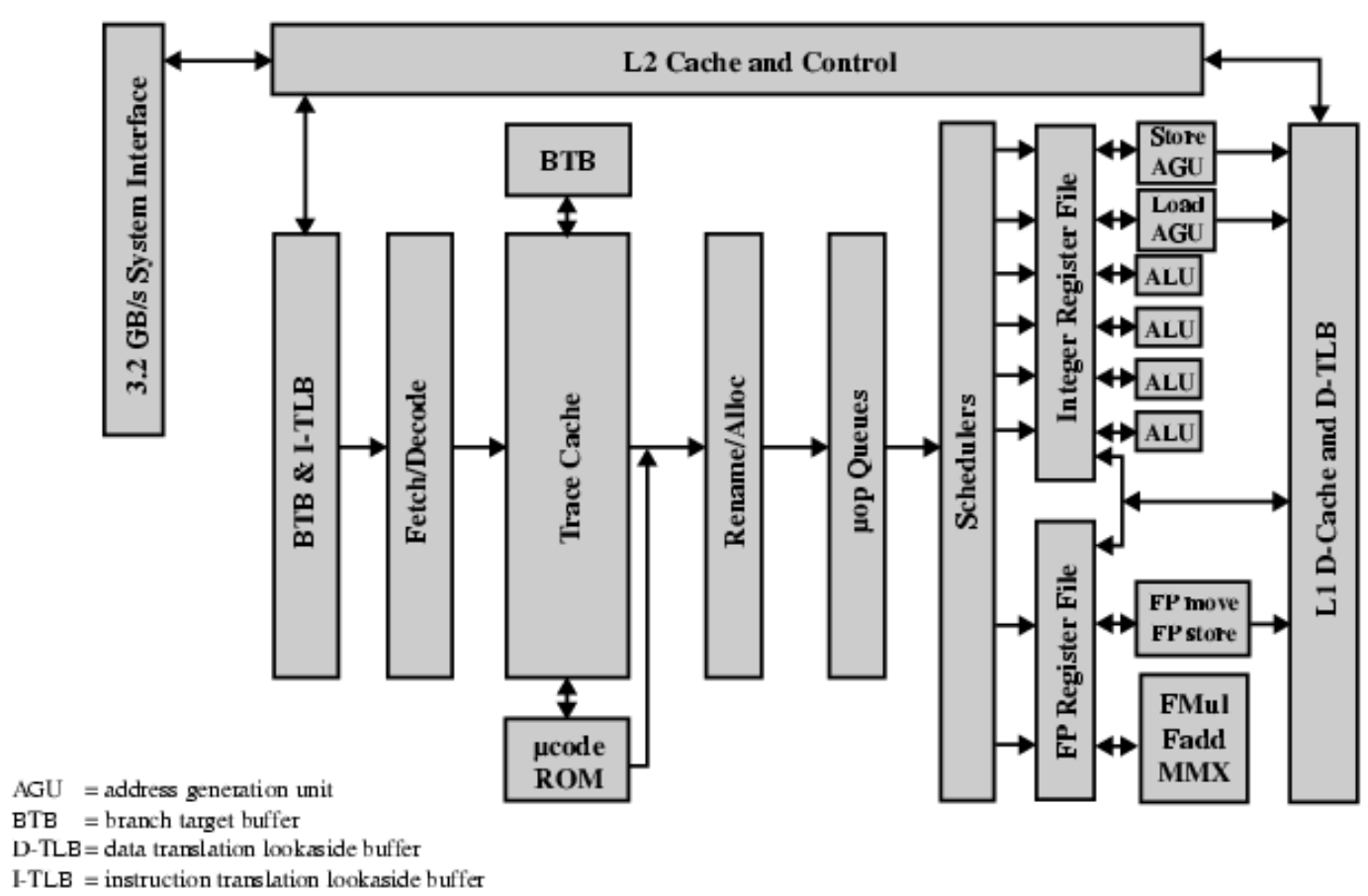
Mekanisme untuk mengkomunikasikan nilai tersebut. Mekanisme untuk menginisialisasi instruksi paralel.

Tersedianya sumber untuk eksekusi paralel dari beberapa instruksi.

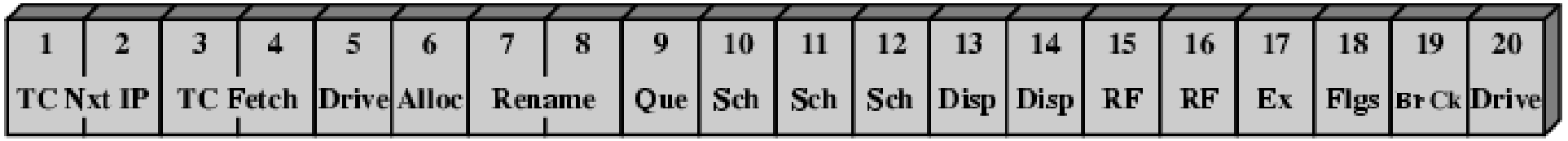
Mekanisme processing instruksi dengan urutan yang sesuai.

Superscalar dapat digunakan untuk berbagai keperluan dan dapat diimplementasikan pada perangkat prosessor seperti Pentium 4.

Pentium 4

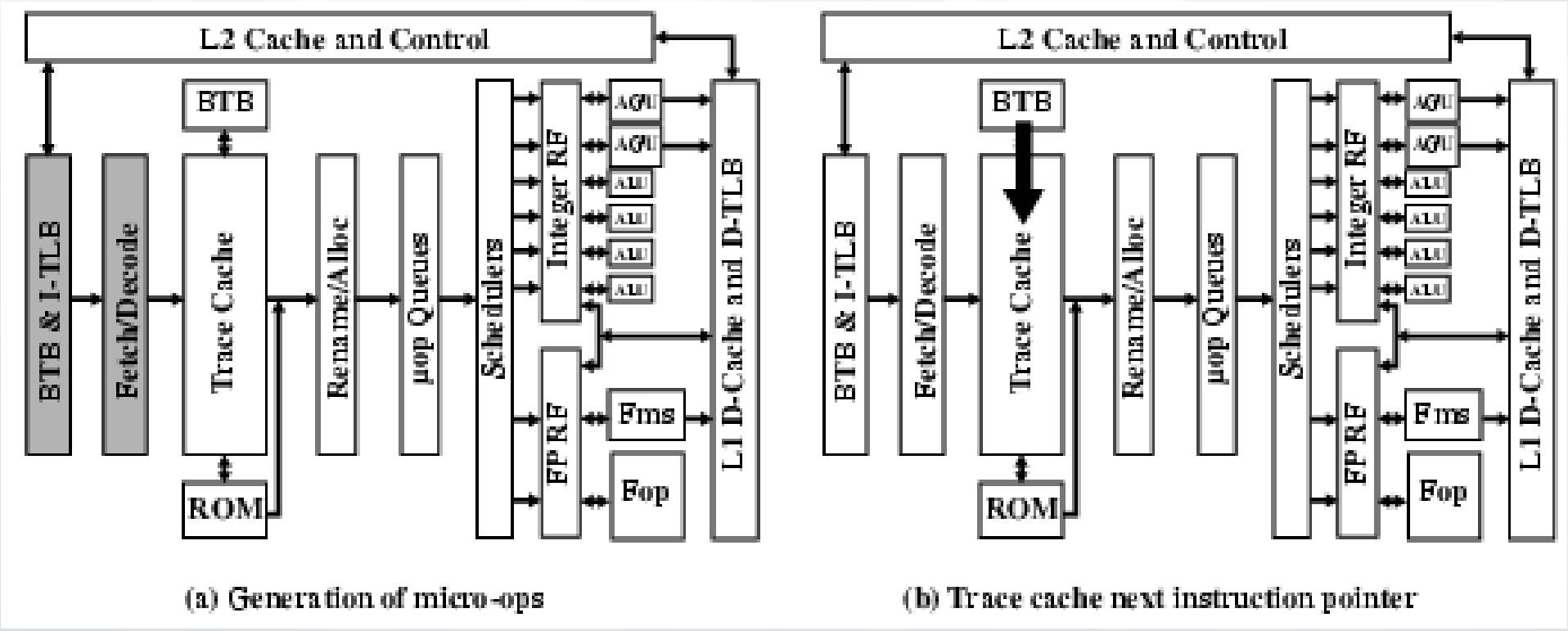

L2 Cache and Control
3.2 GB/s System Interface
BTB
BTB & I-TLB
Fetch/Decode
Trace Cache
µcode ROM
Rename/Alloc
µop Queues
Schedulers
Integer Register File
FP Register File
Store AGU
Load AGU
ALU
ALU
ALU
ALU
FP move
FP store
FMul
Fadd
MMX
L1 D-Cache and D-TLB
AGU = address generation unit
BTB = branch target buffer
D-TLB = data translation lookaside buffer
I-TLB = instruction translation lookaside buffer

Pentium 4 Pipeline

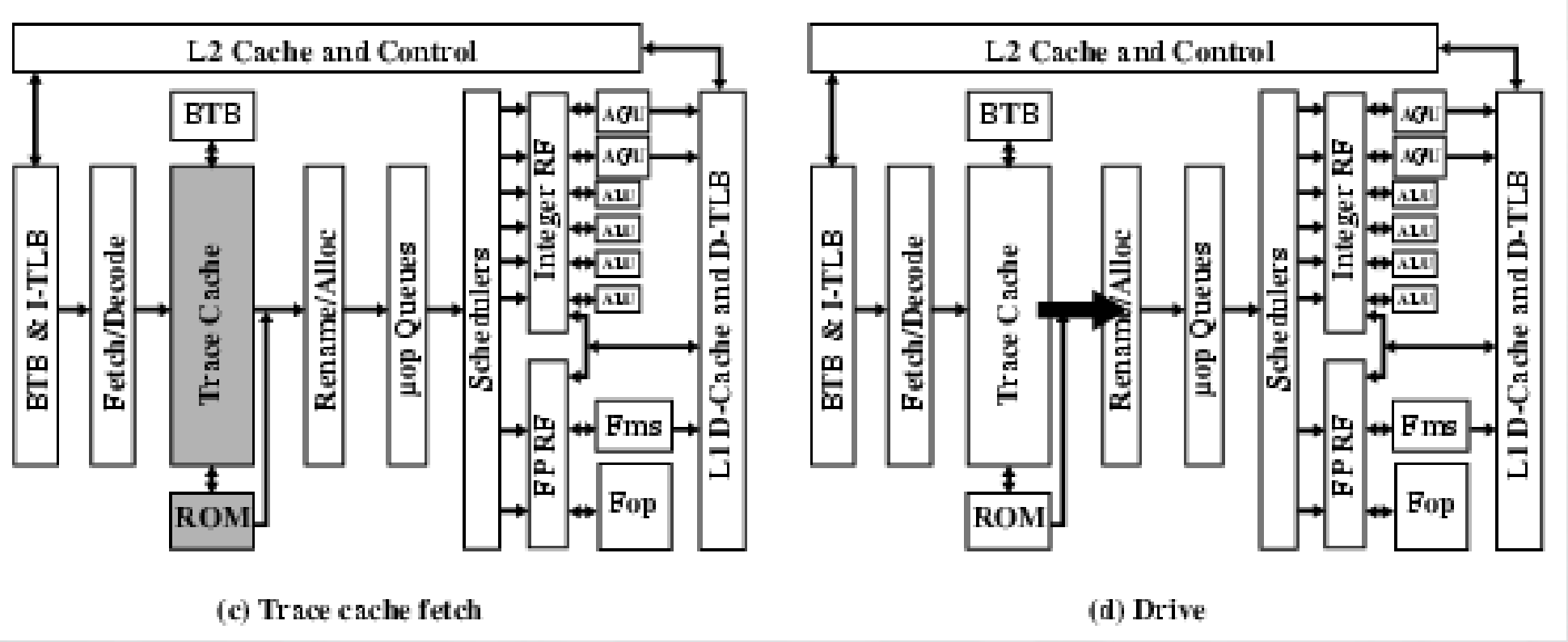


TC Next IP = trace cache next instruction pointer
TC Fetch = trace cache fetch
Alloc = allocate
Rename = register renaming
Que = micro-op queuing
Sch = micro-op scheduling
Disp = Dispatch
RF = register file
Ex = execute
Flgs = flags
Br Ck = branch check

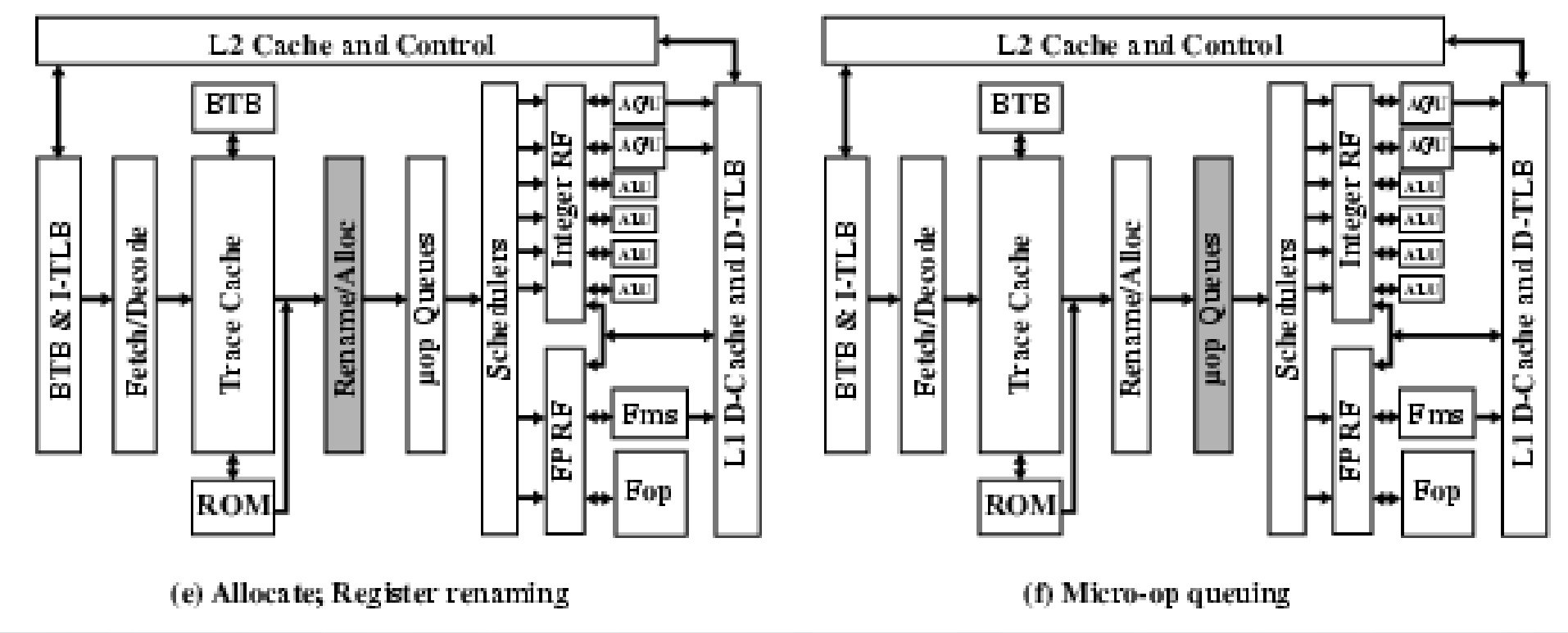
Pentium 4 Pipeline Operation (1)



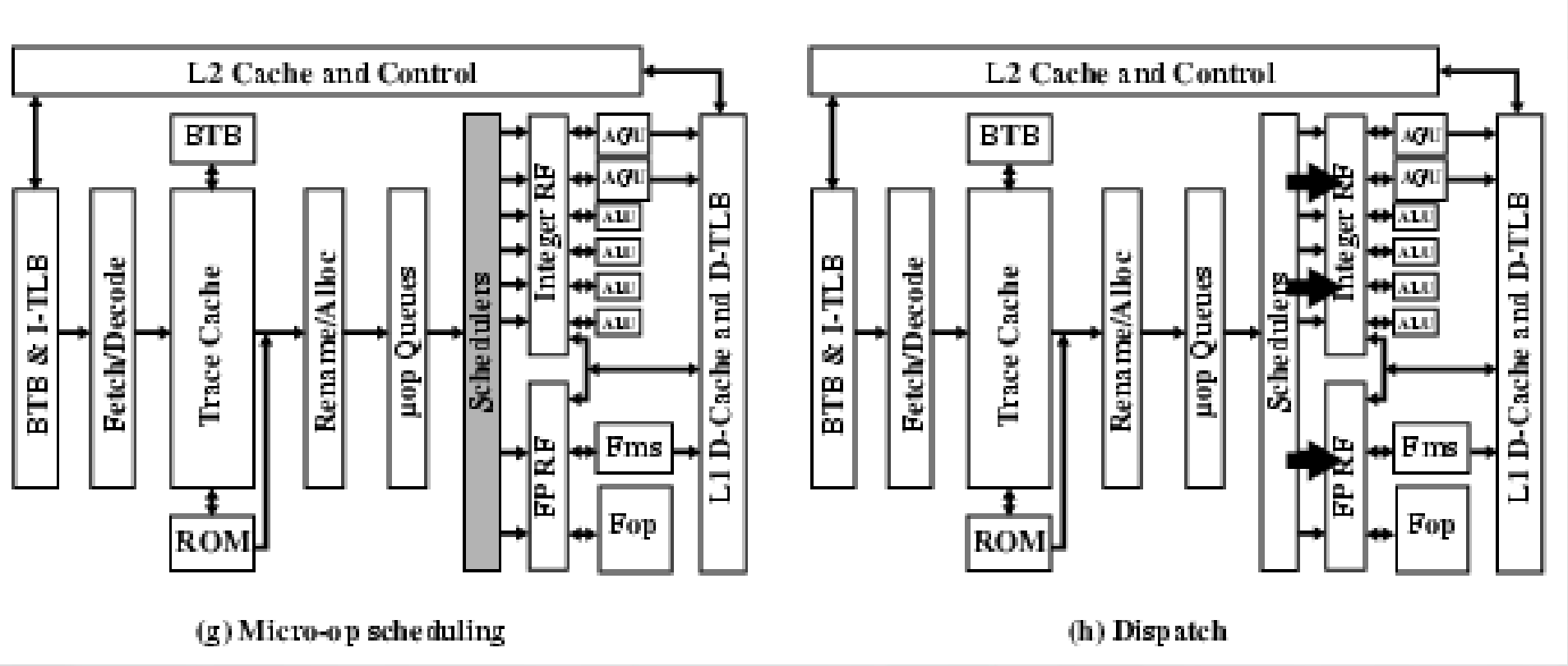
Pentium 4 Pipeline Operation (2)



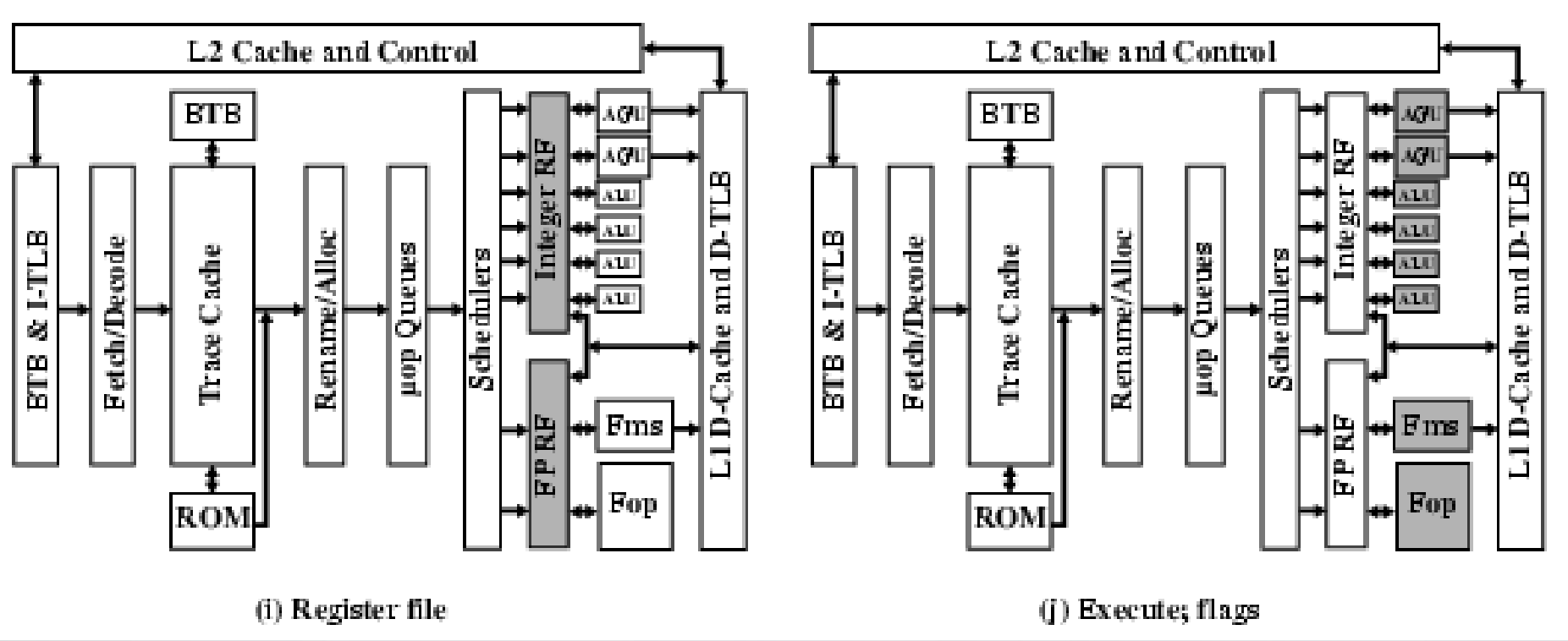
Pentium 4 Pipeline Operation (3)



Pentium 4 Pipeline Operation (4)



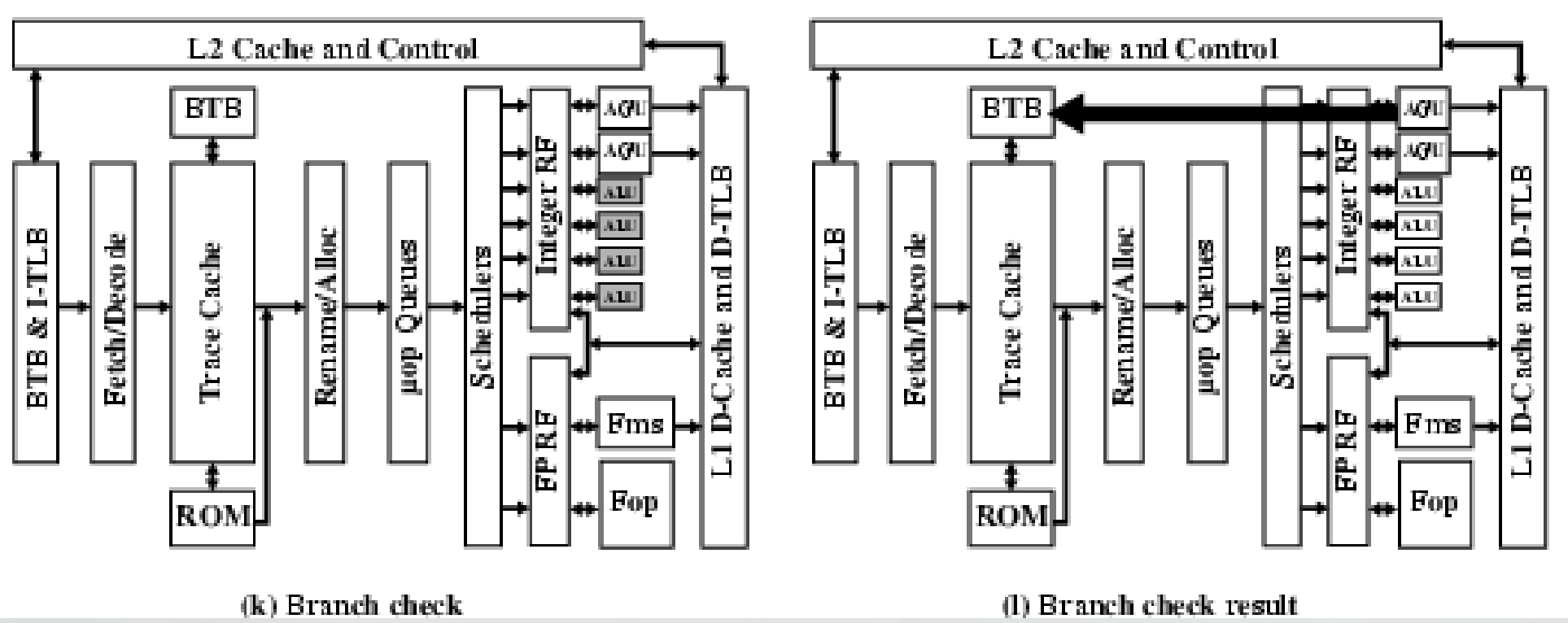
Pentium 4 Pipeline Operation (5)



(i) Register file

(j) Execute; flags

Pentium 4 Pipeline Operation (6)



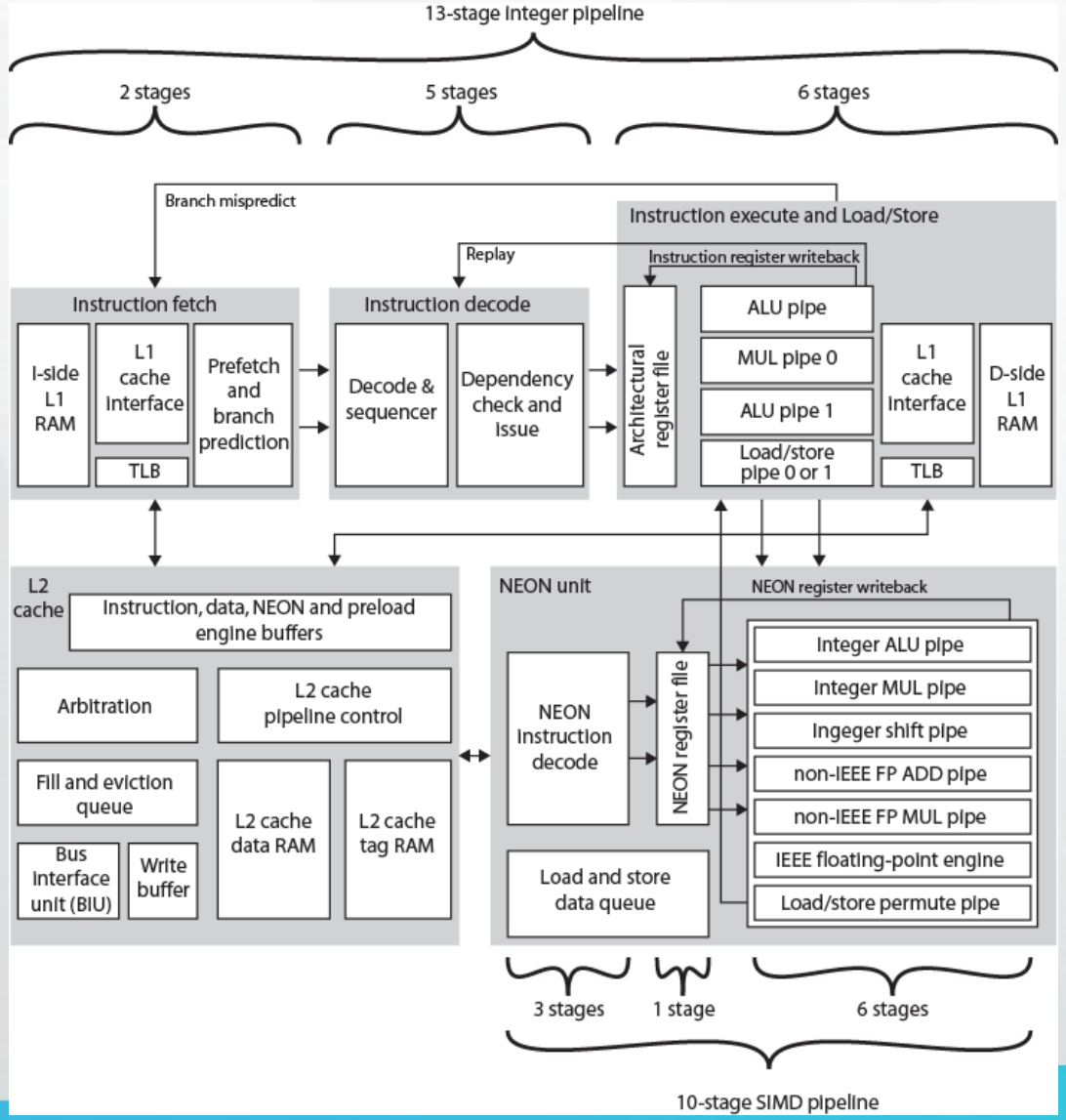
(k) Branch check

(l) Branch check result

ARM CORTEX-A8

- **ARM refers to Cortex-A8 as application processors**
- **Embedded processor running 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**
  - Keep power required to a minimum
  - Out-of-order issue needs extra logic consuming extra power
- **Figure shows the details of the main Cortex-A8 pipeline**
- **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

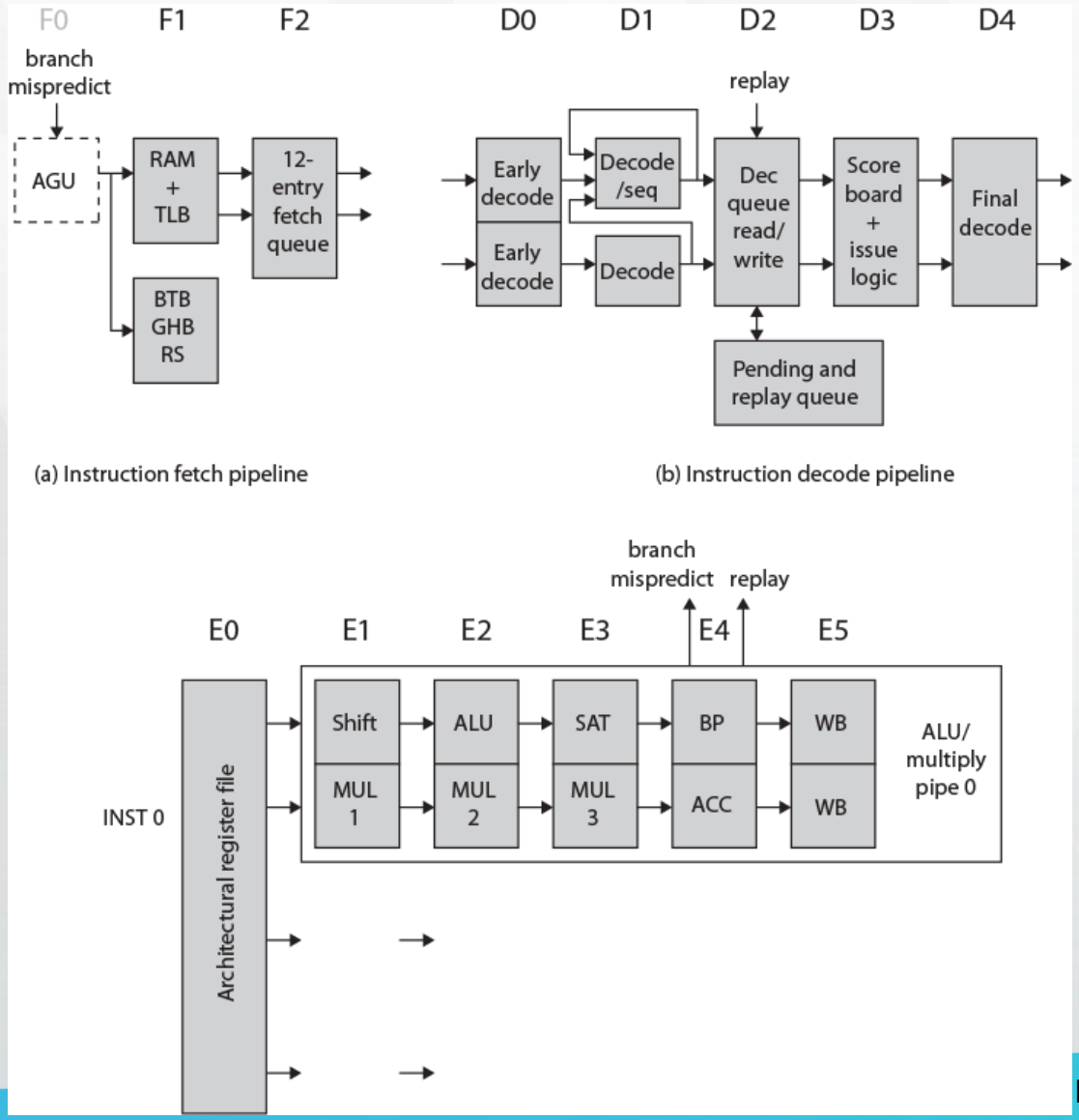
E3 load, data returned and formatted

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



SIMD and Floating-Point Pipeline

SIMD and floating-point instructions pass through integer pipeline

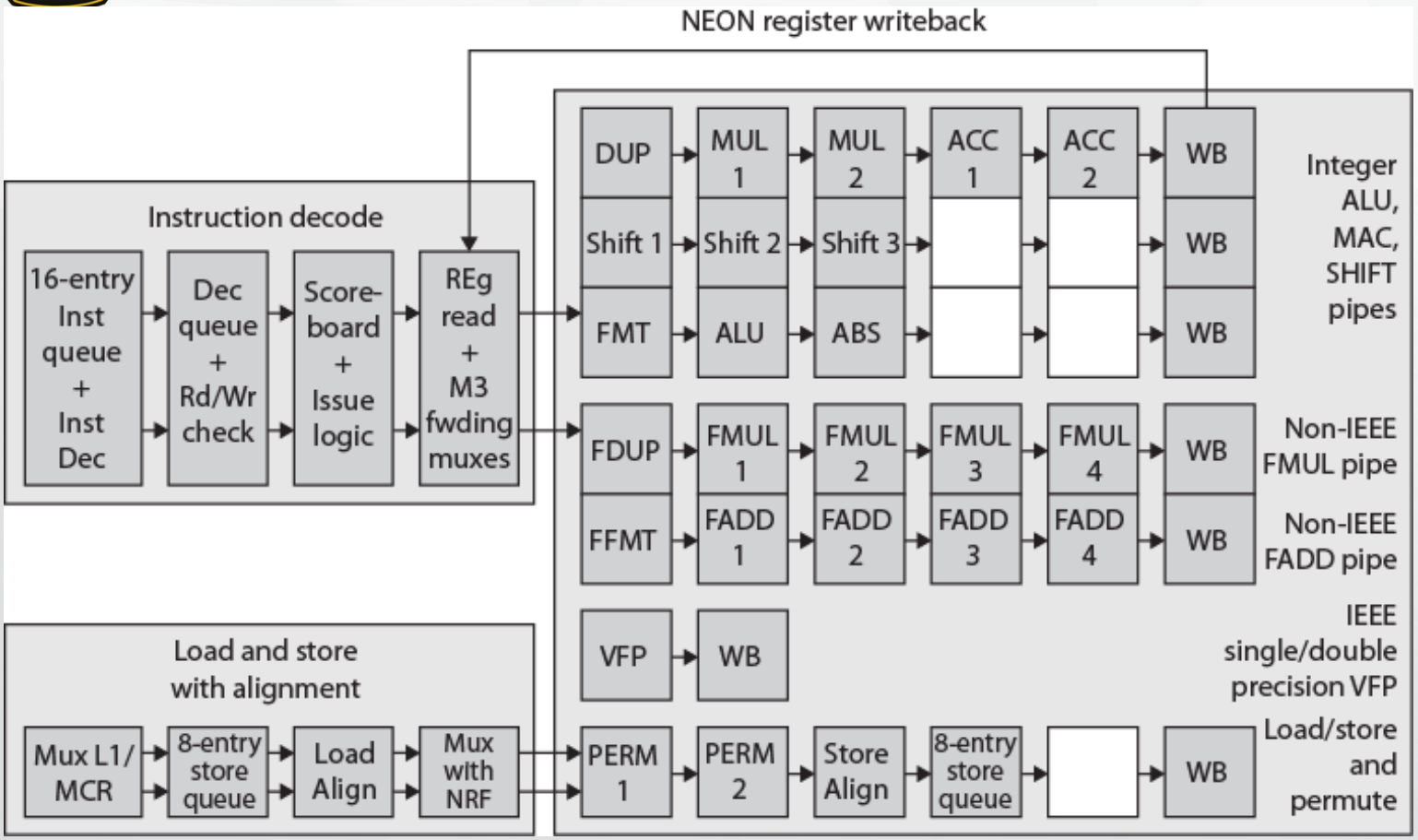
Processed in separate 10-stage pipeline

- 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



Terima Kasih

Pustaka : William Stallings, “Computer Organization and Architecture Designing for Performance Eighth Edition”, Prentice Hall, 2019