TEKNIK PIPELINE & SUPERSCALAR

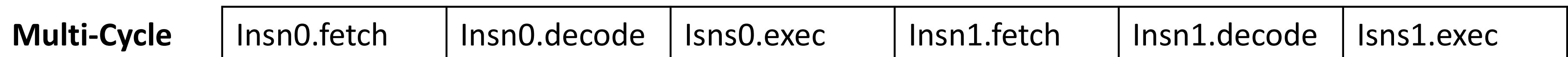
Team Dosen
Telkom University
2016

UNIVERSITAS Telkom

Sebelum Pipeline

Single-Cycle

Multi-Cycle



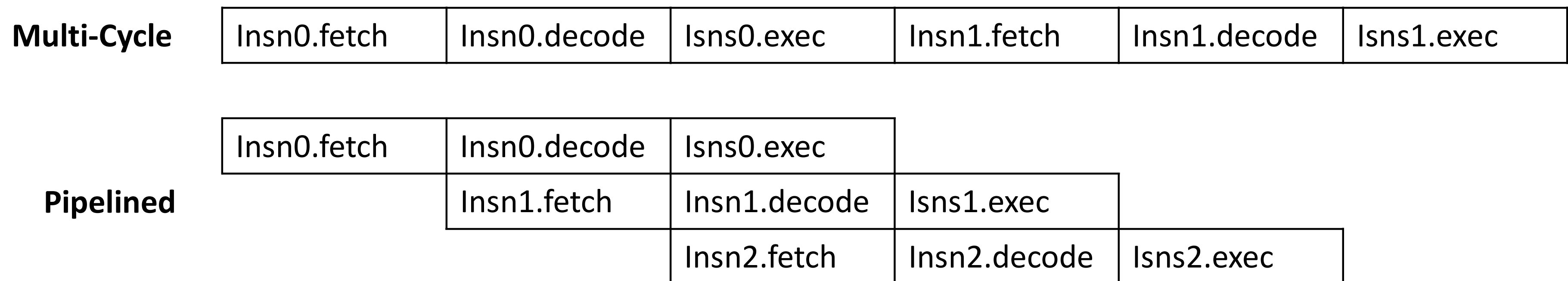
- Single-cycle control : hardwired
  - Low CPI
  - Long clock period
- Multi-cycle control : micro-programmed
  - Short clock period
  - High CPI

Filosofi Pipeline

- Performansi adalah fungsi dari
  - CPI: cycles per instruction
  - Siklus clock
  - Jumlah intsruksi
- Mengurangi salah satu atau semua dari ke 3 faktor akan menikatkan performansi
- Langkah pertama adalah menerapkan konsep pipeline proses eksekusi instruksi
  - Overlap computations
- Hasil ?
  - Mengurangi siklus clock
  - Mengurangi waktu efektif CPU dibandingkan dengan siklus clock asal

UNIVERSITAS Telkom

Setelah Pipeline

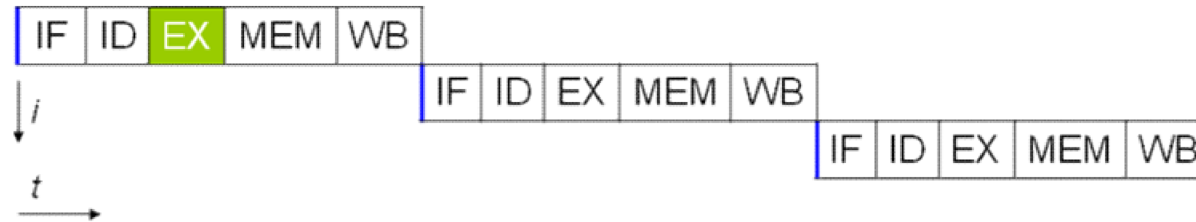


- Diawali dengan multi-cycle
- Ketika isnsn0 bergerak dari stage 1 (fetch) ke stage 2 (decode)
  - Isnsn1 memulai stage 1
- Setiap instruksi melewati semua stage, tetapi semua instruksi masuk dan keluar dengan waktu yang lebih cepat.

UNIVERSITAS Telkom

Pengenalan Pipeline

Kita pecah kerja prosesor kedalam 5 tahapan:



- Instruction fetch (IF)
- Instruction Decode (ID)
- Execution (EX)
- Memory Read/Write (MEM)
- Result Writeback (WB)

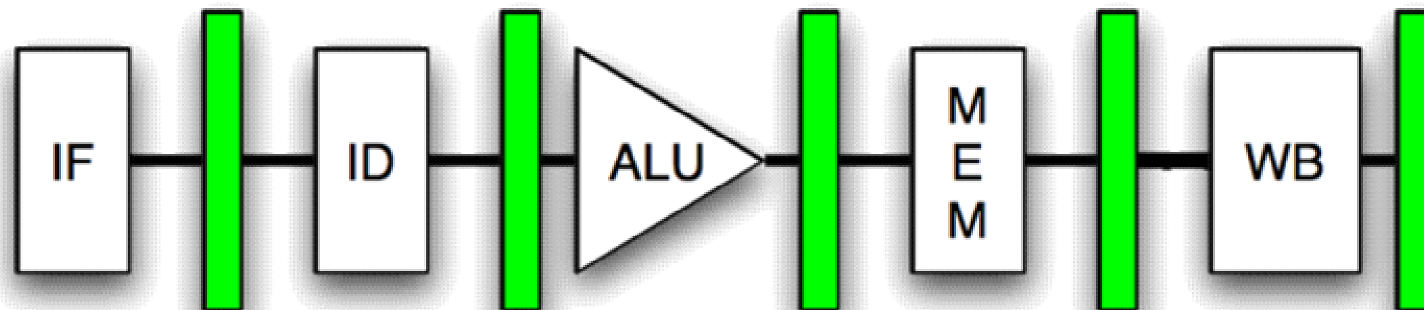
UNIVERSITAS Telkom

Hardware Pipeline

IF	ID	EX	MEM	WB				
	IF	ID	EX	MEM	WB			
		IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	
				IF	ID	EX	MEM	WB

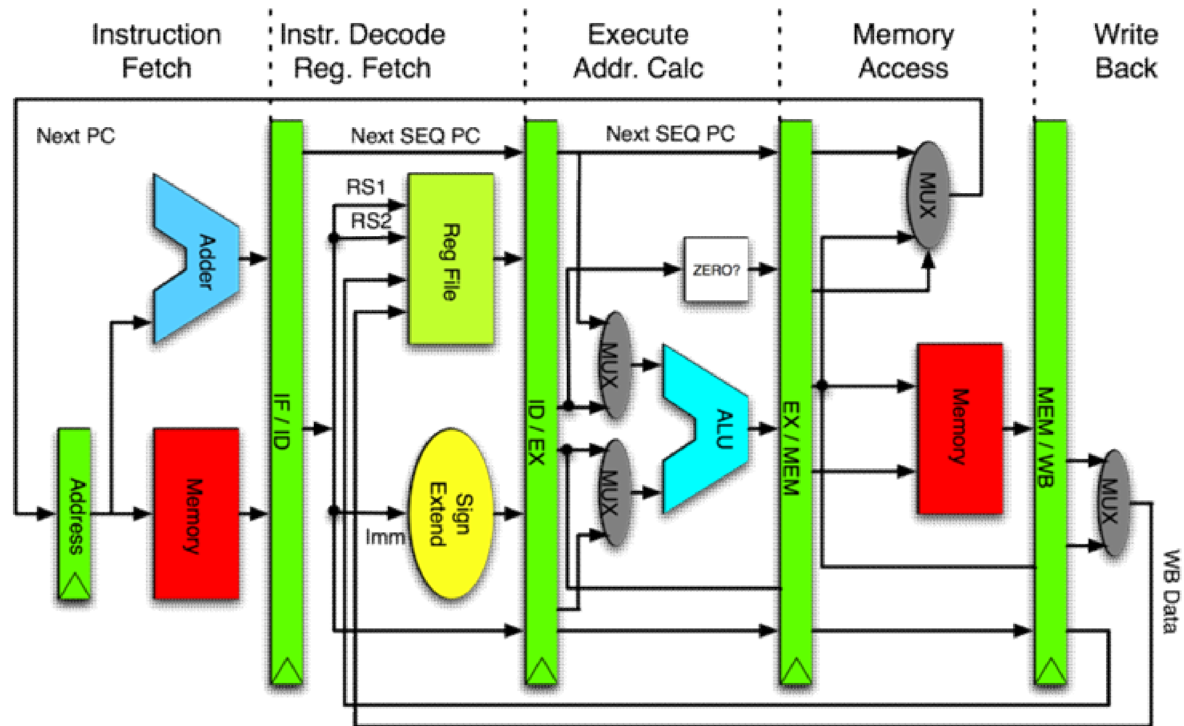
i ↓, *t* →

- Untuk melakukan pipeline diperlukan register penyimpan hasil antar siklus dan juga perlu hardware redundan dari CPU siklus tunggal



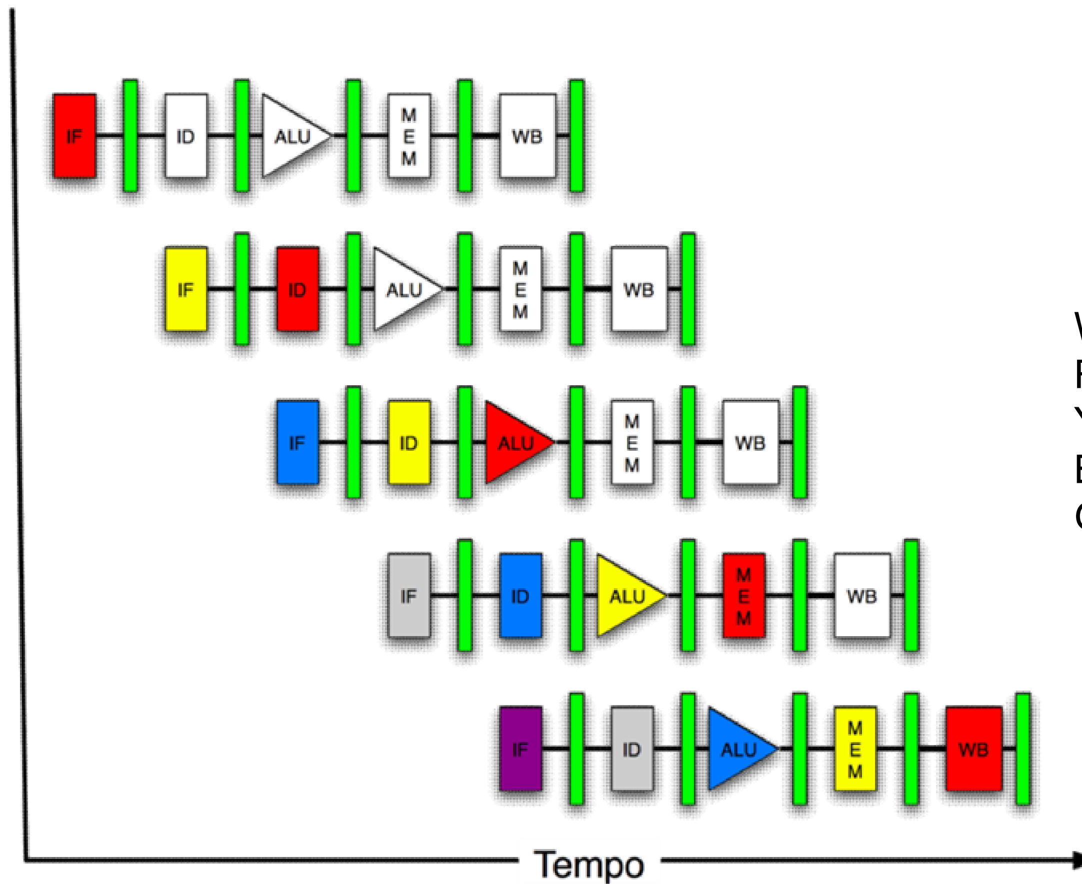
UNIVERSITAS
Telkom

Rancangan Prosesor Pipeline Umum



UNIVERSITAS Telkom

Pipeline Menjalankan 5 instruksi

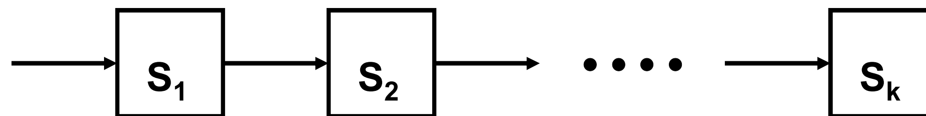


WHITE – NOP
RED – 2ND INSTR.
YELLOW – 3RD INSTR.
BLUE – 4TH INSTR.
GREY – 5TH INSTR.

UNIVERSITAS Telkom

Linear Pipeline Processor

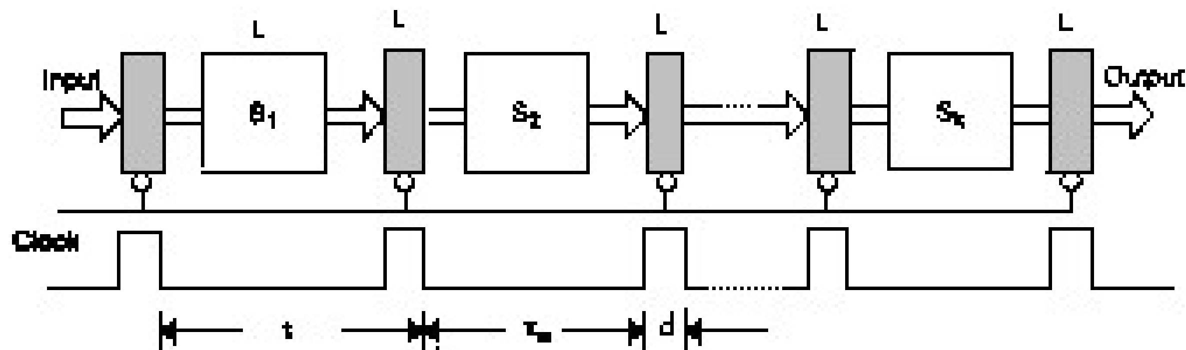
- Pipeline linier memproses urutan subtask dengan preseden linier.
- Data mengalir dari tahap S1 ke tahap akhir Sk
- Kendali aliran data: sinkron atau asinkron



UNIVERSITAS Telkom

Pipeline Sinkron

- Semua transfer terjadi serempak
- Satu task atau operasi memasuki pipeline per siklus
- Tabel reservasi prosesor : diagonal



UNIVERSITAS Telkom

Utilisasi Tempat dan Waktu Pipeline

Stage				
S3			T1	T2
S2		T1	T2	T3
S1	T1	T2	T3	T4
	1	2	3	4

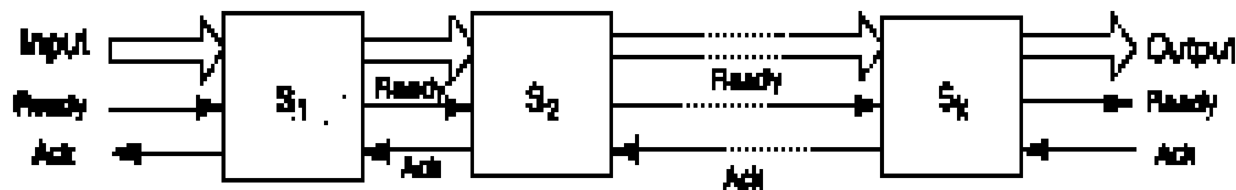
Time (Siklus) →

Pipeline penuh setelah 4 siklus

UNIVERSITAS Telkom

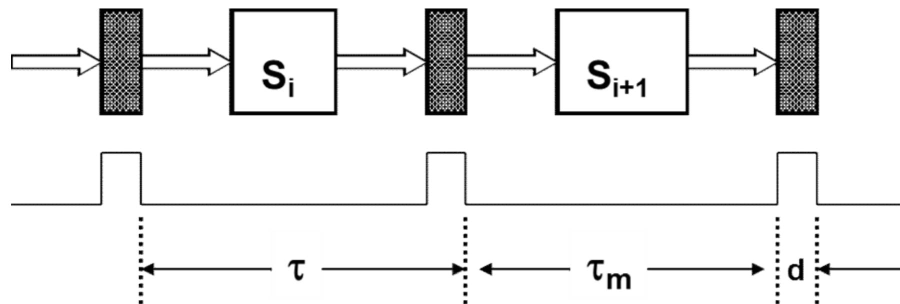
Pipeline Asinkron

- Transfer baru dilakukan jika prosesor berikutnya siap
- Ada protokol handshaking antar prosesor
- Umum digunakan di sistem multiprosesor dengan message-passing



Universitas Telkom

Pipeline Clock and Timing



- Siklus clock pipeline : τ
- Delay Latch : d

$$\tau = max\ \{\tau_m\} + d$$

- Pipeline frequency : f

$$f = \frac{1}{\tau}$$

UNIVERSITAS Telkom

Speedup dan Efisiensi

- *Pipeline k-tahap* memproses *n* task dalam
- k + (n-1) siklus clock :
- *k* siklus untuk task pertama dan *n-1* siklus untuk sisa *n-1* task

- Waktu total untuk memproses *n* task

$$T_k = [k + (n-1)]\tau$$

- Prosesor tanpa pipeline

$$T_1 = nk\tau$$

- Faktor Speedup

$$S_k = \frac{T_1}{T_k} = \frac{nk\tau}{[k + (n-1)]\tau} = \frac{nk}{k + (n-1)}$$

UNIVERSITAS Telkom

Efisiensi dan Throughput

- Efisiensi dari k-tahap pipeline :

$$E_k = \frac{S_k}{k} = \frac{n}{k + (n-1)}$$

- Pipeline throughput (jumlah task per satuan waktu) :
- Catatan : ekivalen dgn IPC

$$H_k = \frac{n}{[k + (n-1)]\tau} = \frac{nf}{k + (n-1)}$$

UNIVERSITAS Telkom

Contoh Performansi Pipeline

- Task mempunyai 4 subtask dgn waktu : t1=60, t2=50, t3=90, and t4=80 ns; Delay = 10
  - $\tau = max\ \{\tau_m\} + d$
  - $\tau = 90\ ns + 10\ ns$
  - $\tau = 100\ ns$
- Waktu siklus pipeline = 100 ns

- Eksekusi non-pipeline
  - $\tau = \tau_1 + \tau_2 + \tau_3 + \tau_4$
  - $\tau = 60\ ns + 50\ ns + 90\ ns + 80\ ns$
  - $\tau = 280\ ns$
- Waktu siklus non-pipeline = 280 ns

- Speedup untuk kasus diatas : 280/100 = 2.8 !!

UNIVERSITAS Telkom

Contoh Performansi Pipeline

- Waktu pipeline utk 1000 task :
  - $T_k = [1000 + (4 - 1)]100\ ns$
  - $T_k = 1003\ x\ 100\ ns$
  - $T_k = 100300\ ns$
- Waktu skuensial : 1000*280ns
  - $T_1 = nk\tau$
  - $T_1 = 1000\ x\ 280\ ns$
- Throughput= 1000/1003
- Apa yg salah?
- Bagaimana cara memperbaiki performansi ?

UNIVERSITAS Telkom

Peningkatan Performansi

- Tidak membuat dua kali lebih cepat:
  - Fetch biasanya lebih sebentar dari eksekusi
  - Jika ekseksi melibatkan akses memori, tingkat fetch harus menunggu
  - Setiap jump atau pencabangan berarti instruksi yang di-prefetch bukan instruksi yang diperlukan
- Tambah tingkat untuk meningkatkan performansi

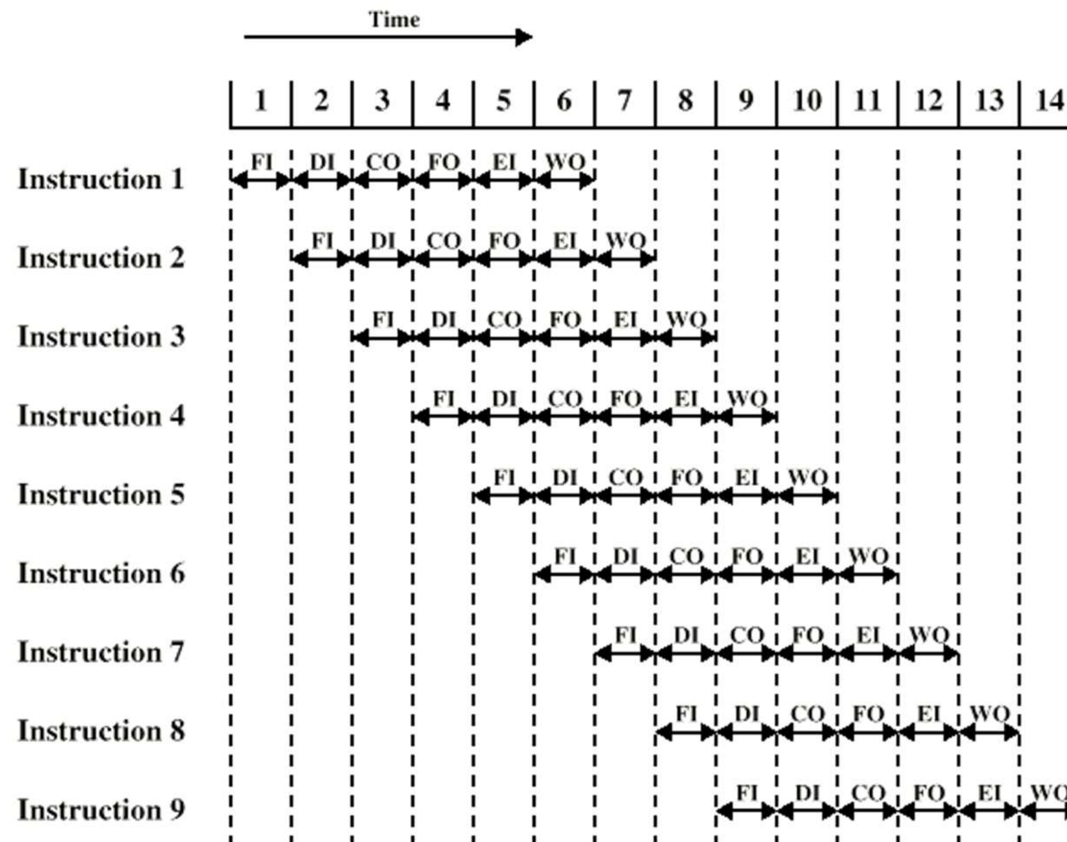
UNIVERSITAS Telkom

Tingkat-tingkat Pengolahan Instruksi:

- Fetch instruction (FI)
- Decode instruction (DI)
- Calculate operands (CO)
- Fetch operands (FO)
- Execute instructions (EI)
- Write operand (WO)

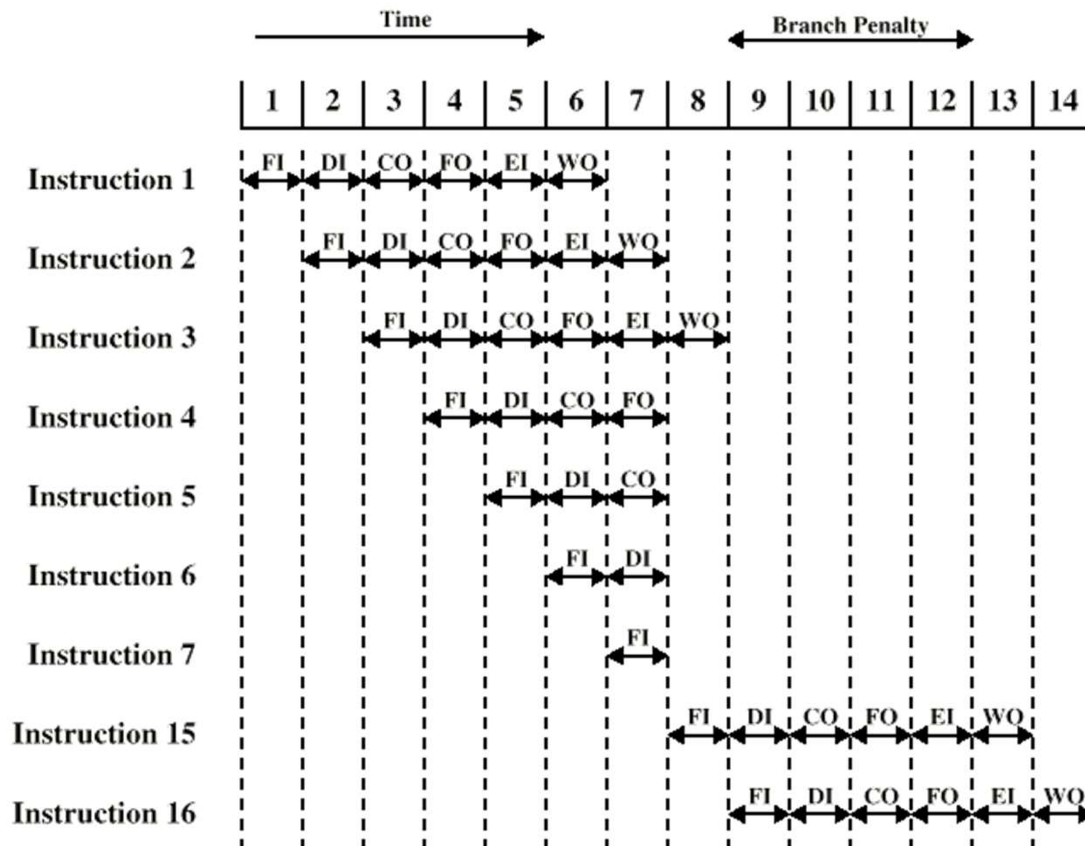
UNIVERSITAS Telkom

Diagram Waktu Pipeline



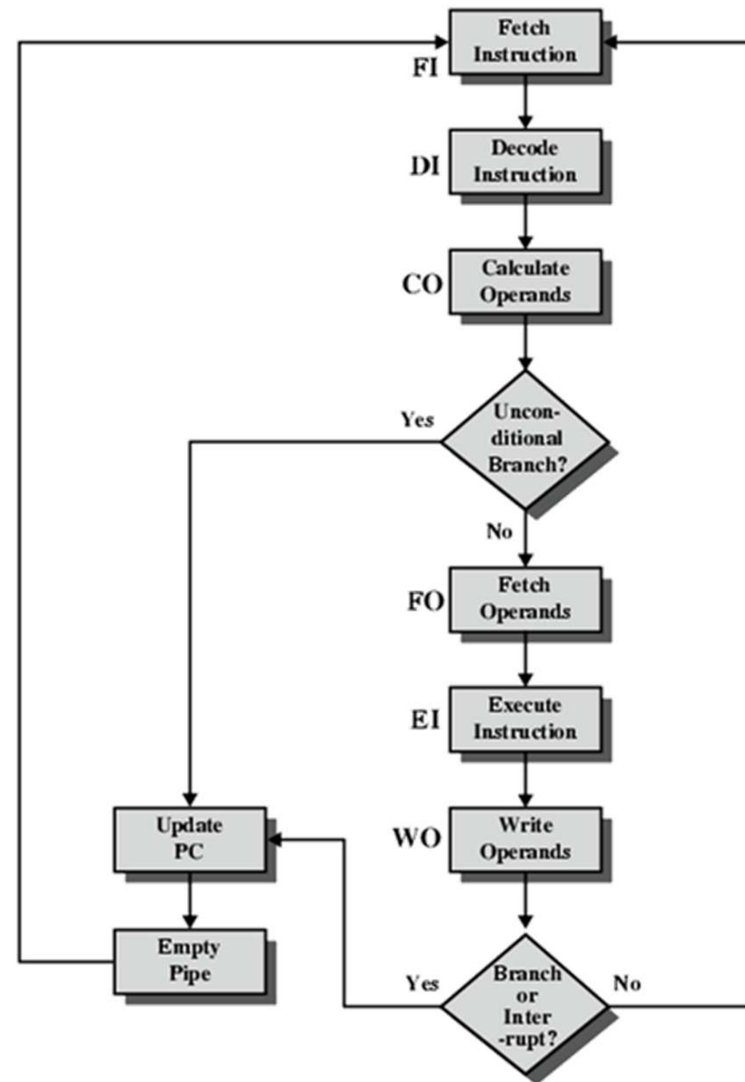
UNIVERSITAS Telkom

Pencabangan di Pipeline



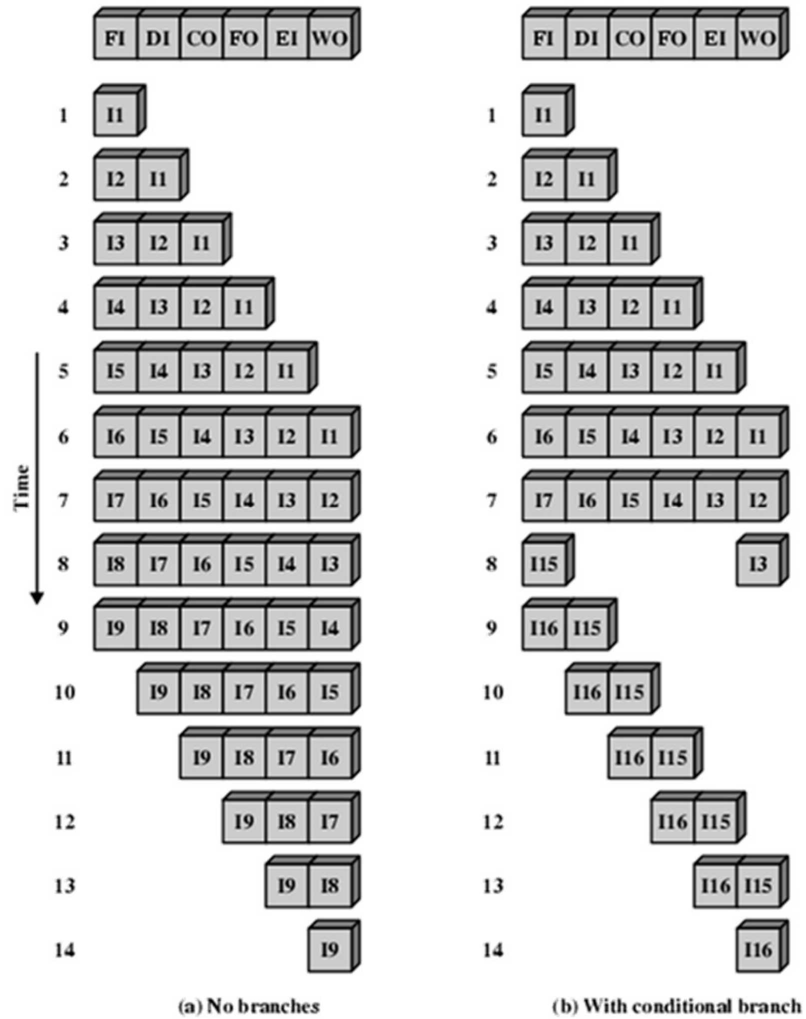
UNIVERSITAS Telkom

Enam Tingkat Pipeline Instruksi



UNIVERSITAS Telkom

Alternatif Kondisi Pipeline



Penanganan Pencabangan

- Memperbanyak aliran (Multiple Streams)
- Prefetch Target Pencabangan
- Loop buffer
- Prediksi pencabangan
- Penundaan Pencabangan

UNIVERSITAS Telkom

Memperbanyak Aliran

- Ada 2 pipeline
- Prefetch setiap pencabangan ke pipeline terpisah
- Gunakan pipeline yang sesuai

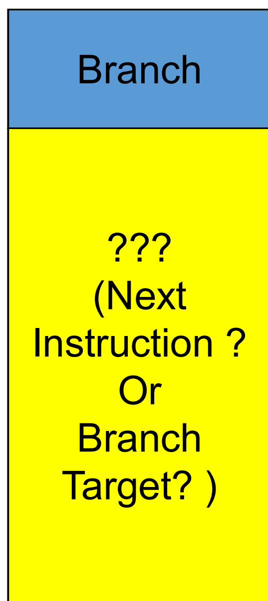
- Pencabangan yang lebih banyak akan memerlukan lebih banyak pipeline

UNIVERSITAS Telkom

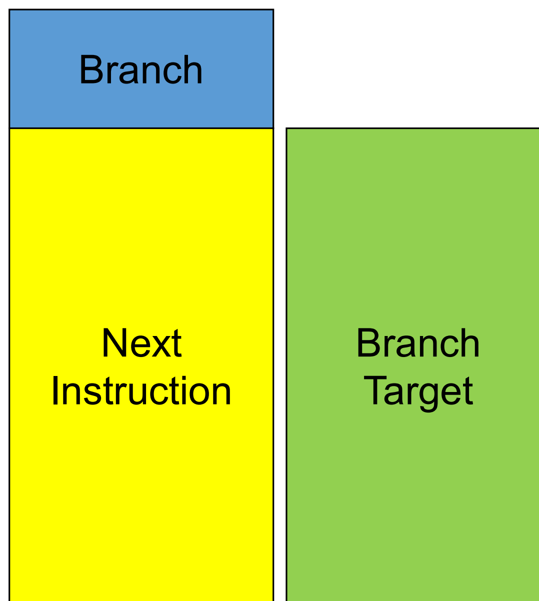
Prefetch Target Pencabangan

- Target dari pencabangan di-prefetch sebagai tambahan dari instruksi pencabangan berikutnya
- Simpan target sampai pencabangan di eksekusi
- Digunakan IBM 360/91

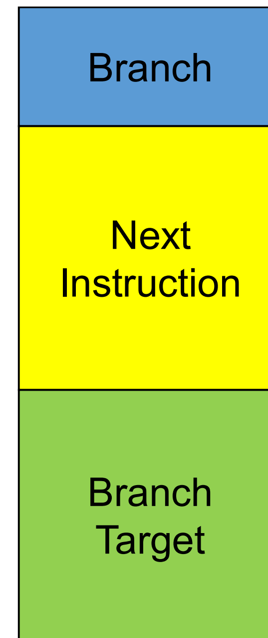
UNIVERSITAS Telkom



Single Pipeline



Double Pipeline



Branch Target
Prefect

UNIVERSITAS
Telkom

Loop Buffer

- Penyimpanan sangat cepat
- Dilakukan oleh tingkat fetch dari pipeline
- Periksa buffer sebelum fetch dari memori
- Sangat baik untuk loop kecil atau jump
- Misal : cache
- Digunakan CRAY-1

UNIVERSITAS
Telkom

Prediksi Pencabangan

- Prediksi Tak Terjadi
  - Asumsikan jump tidak akan terjadi
  - 68020 & VAX 11/780
- Prediksi Terjadi
  - Asumsikan jump akan terjadi
- *Prediksi oleh Opcode*
- Switch Terjadi/Tak Terjadi
  - Berdasarkan sejarah sebelumnya
  - Tabel sejarah pencabangan
    - Menyimpan alamat instruksi pencabangan, sejarah, alamat tujuan instruksi di sebuah tabel di cache

UNIVERSITAS Telkom

Penundaan Pencabangan

- Jangan lakukan jump sampai benar-benar tak bisa dihindari
- Susun kembali instruksi

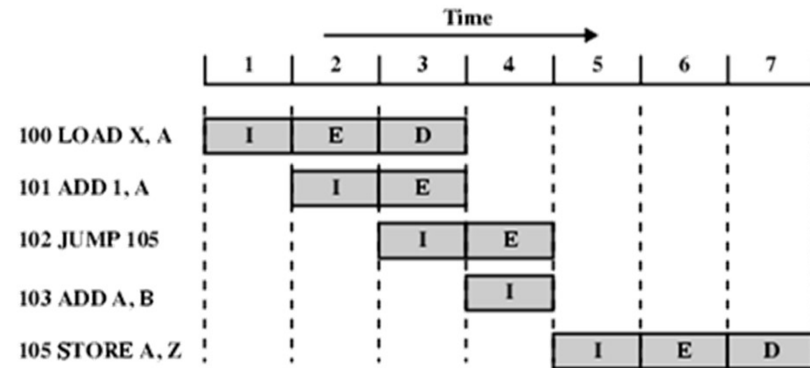
UNIVERSITAS
Telkom

Pencabangan Normal dan Tertunda

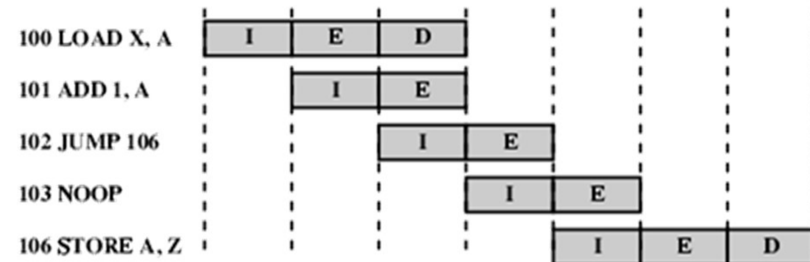
Alamat	Normal	Tertunda	Teroptimasi
100	LOAD X,A	LOAD X,A	LOAD X,A
101	ADD 1,A	ADD 1,A	JUMP 105
102	JUMP 105	JUMP 106	ADD 1,A
103	ADD A,B	NOOP	ADD A,B
104	SUB C,B	ADD A,B	SUB C,B
105	STORE A,Z	SUB C,B	STORE A,Z
106		STORE A,Z	

UNIVERSITAS Telkom

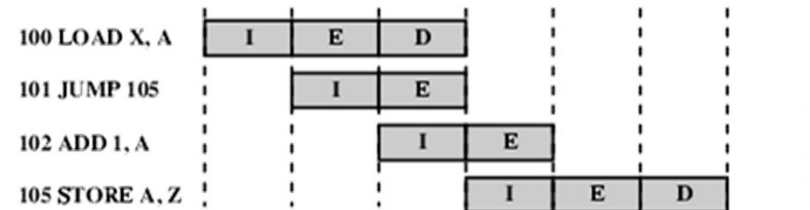
Penggunaan Pencabangan Tertunda



(a) Traditional Pipeline



(b) RISC Pipeline with Inserted NOOP



(c) Reversed Instructions

UNIVERSITAS Telkom

Pendahuluan Superscalar

- Instruksi (arithmetic, load/store, conditional branch) dapat dimulai dan di eksekusi secara terpisah
- Bisa diterapkan untuk RISC & CISC
- Kebanyakan diterapkan di RISC

UNIVERSITAS
Telkom

Kenapa Superscalar?

- Kebanyakan operasi yang dilakukan adalah skalar
- Meningkatkan operasi ini berarti meningkatkan performa secara keseluruhan

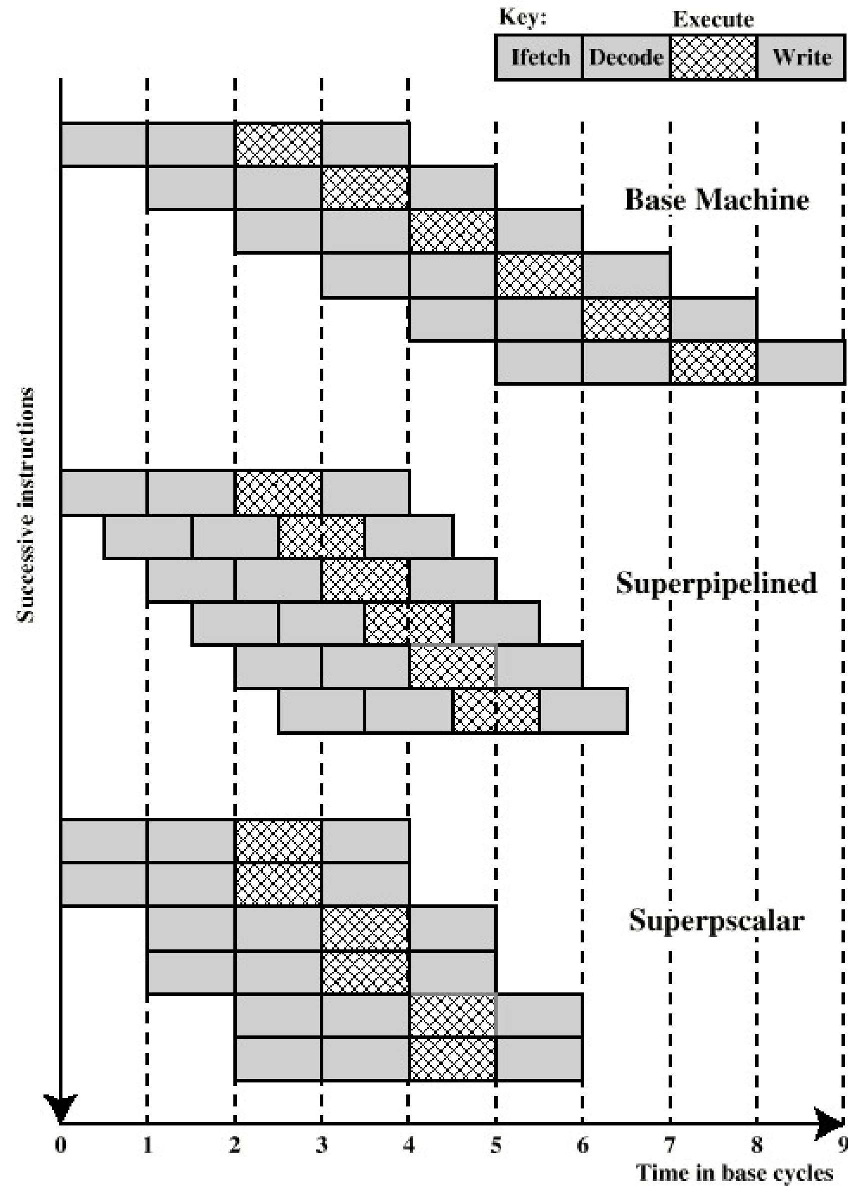
UNIVERSITAS Telkom

Superpipelined

- Banyak tingkat pipeline hanya memerlukan kurang dari ½ siklus clock
- Clock internal ganda menyebabkan bisa dilakukan dua task per siklus clock eksternal
- Superscalar membuat proses fetch dan eksekusi paralel

UNIVERSITAS
Telkom

Superscalar vs Superpipeline



UNIVERSITAS Telkom

Keterbatasan

- Level instruksi paralel
- Optimasi berbasis compiler
- Teknik-teknik hardware
- Dibatasi oleh:
  - Dependensi data
  - Dependensi prosedur
  - Perebutan resource
  - Dependensi output (write after write)
  - Antidependensi (Write after read)

UNIVERSITAS Telkom

Dependensi Data

- ADD r1, r2 (r1 := r1+r2;)
- MOVE r3,r1 (r3 := r1;)
- Dapat melakukan fetch dan decode instruksi kedua secara paralel dengan instruksi pertama
- **Tidak dapat** meng-eksekusi intruksi kedua sampai intruksi pertama selesai

UNIVERSITAS Telkom

Dependendensi Prosedur

- Tidak dapat meng-eksekusi intruksi setelah pencabangan secara paralel dengan instruksi sebellum pencabangan
- Juga, jika panjang instruksi berbeda, instruksi harus di-decode untuk tahu berapa fetch yang diperlukan
- Ini untuk mencegah fetch simultan

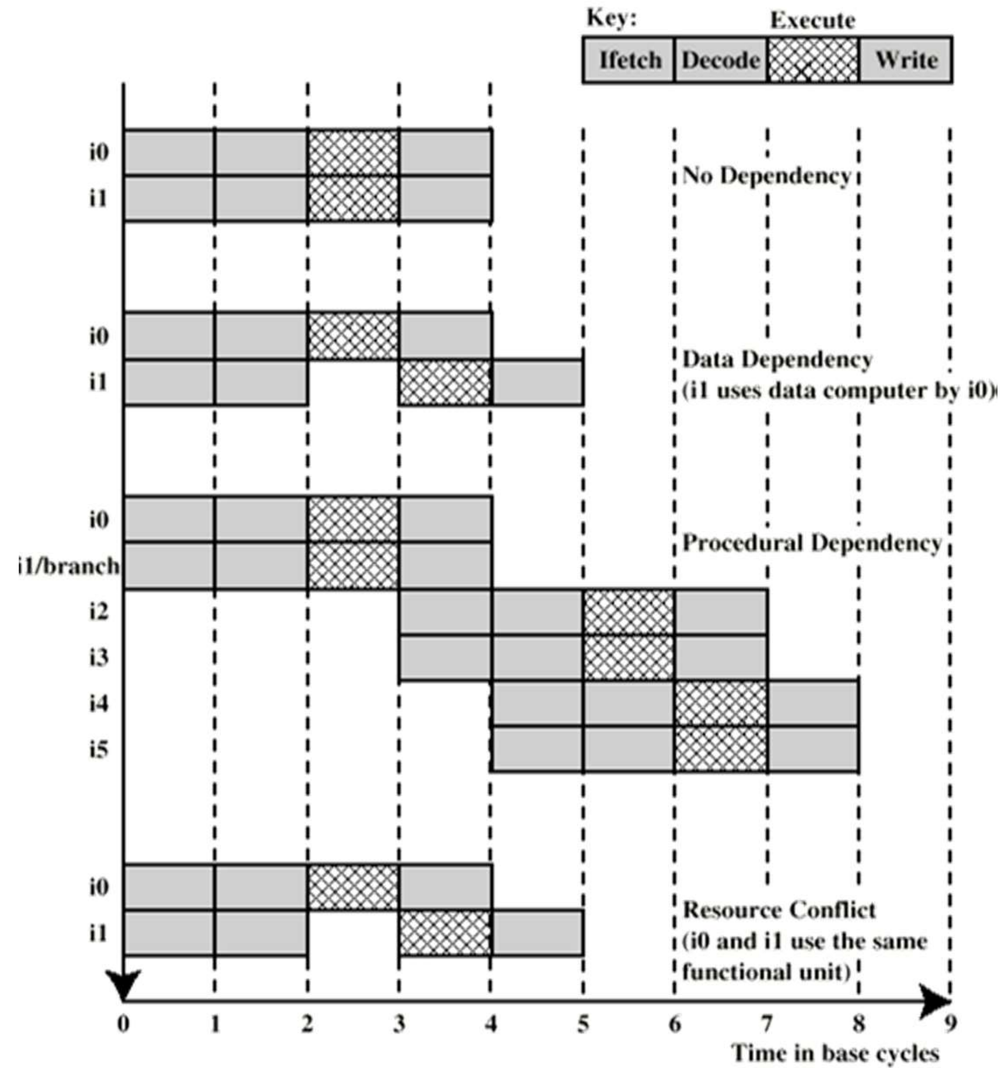
UNIVERSITAS Telkom

Perebutan Resource

- Dua atau lebih instruksi memerlukan akses resource yang sama pada satu saat
  - Misal: dua intruksi arithmetik
- Bisa dibuat resource ganda
  - Misal: ada dua unit arithmetik

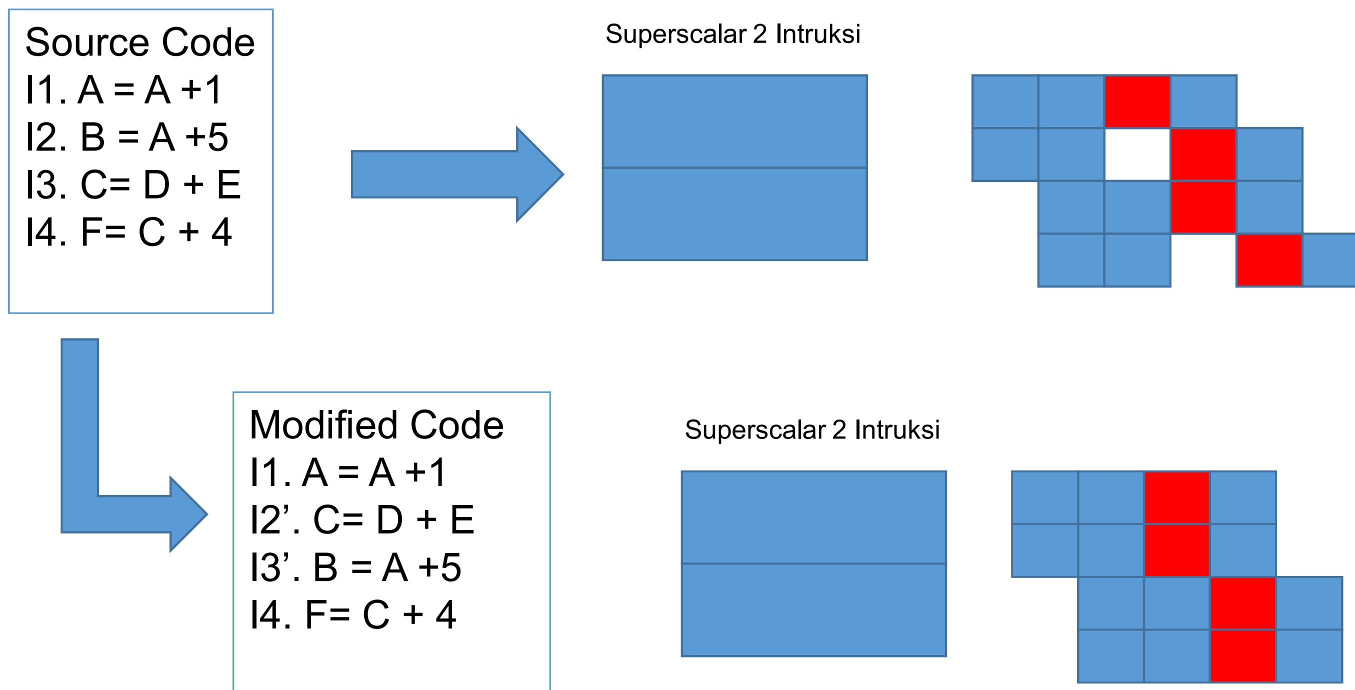
UNIVERSITAS Telkom

Pengaruh Dependensi



UNIVERSITAS Telkom

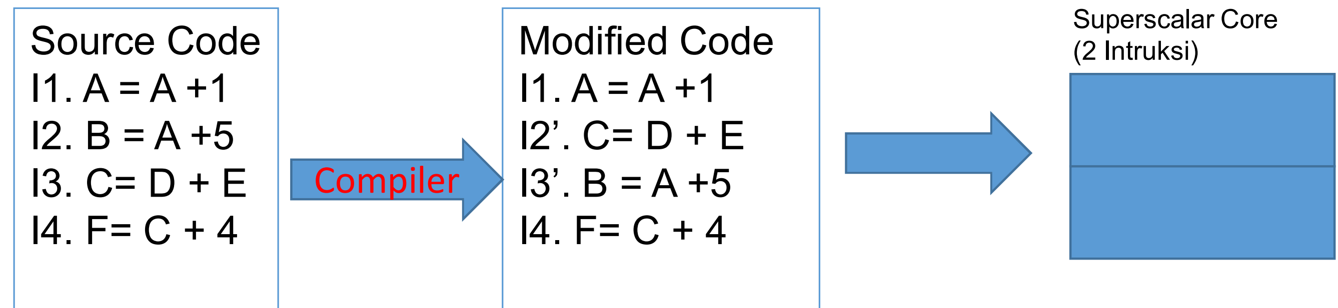
Modified Code



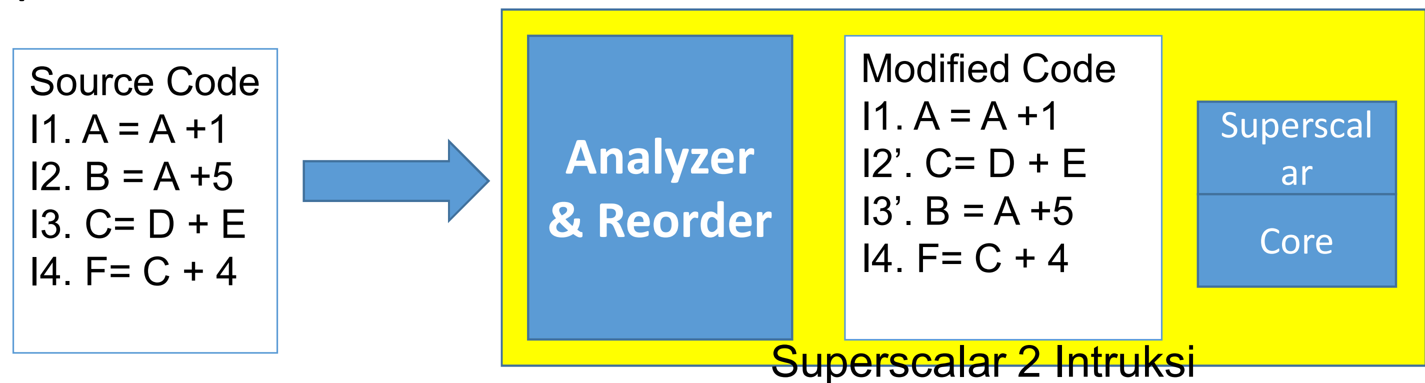
Siapa yang harus melakukan pengubahan code ??

ILP vs MLP

- Serahkan pengubahan pada compiler (tambahan tugas) → Instruction Level Parallelism



Serahkan pada prosesor → Machine Level Parallelism



Universitas Telkom

Dasar Perancangan

- Paralel Tingkat Instruksi
  - Instruksi yang berurutan harus independen
  - Eksekusi bisa saling overlap
  - Diatur oleh dependensi data dan prosedur
- Paralel di Mesin
  - Kemampuan untuk memanfaaatkan paralel tingkat instruksi
  - Diatur oleh jumlah pipeline paralel

UNIVERSITAS Telkom

Kebijakan Untuk Instruksi

- Atur instruksi mana yang di-fetch
- Atur instruksi mana yang di-eksekusi
- Atur instruksi mana yang mengubah register dan memori

UNIVERSITAS
Telkom

In-Order Issue In-Order Completion

- Instruksi disusun sesuai urutan terjadinya
- Tidak cukup efisien
- Terdapat fetch >1 instruksi

UNIVERSITAS Telkom

Contoh Soal

- Figure 14.4a gives an example of this policy. We assume a superscalar pipeline capable of fetching and decoding two instructions at a time, having three separate functional units (e.g., two integer arithmetic and one floating-point arithmetic), and having two instances of the write-back pipeline stage. The example assumes the fol- lowing constraints on a six-instruction code fragment:
- I1 requires two cycles to execute.
- I3 and I4 conflict for the same functional unit.
- I5 depends on the value produced by I4.
- I5 and I6 conflict for a functional unit.

UNIVERSITAS Telkom

In-order Issue In-order Completion

Decode		Execute		Write		Cycle
1	2					1
						2
						3
						4
						5
						6
						7
						8

UNIVERSITAS
Telkom

In-order Issue In-order Completion

Decode		Execute		Write		Cycle
1	2					1
3	4	1	2			2
						3
						4
						5
						6
						7
						8

UNIVERSITAS
Telkom

In-order Issue In-order Completion

Decode		Execute		Write		Cycle
1	2					1
3	4	1	2			2
	4	1				3
						4
						5
						6
						7
						8

UNIVERSITAS Telkom

In-order Issue In-order Completion

Decode		Execute		Write		Cycle
1	2					1
3	4	1	2			2
	4	1				3
5	6	3	4	1	2	4
						5
						6
						7
						8

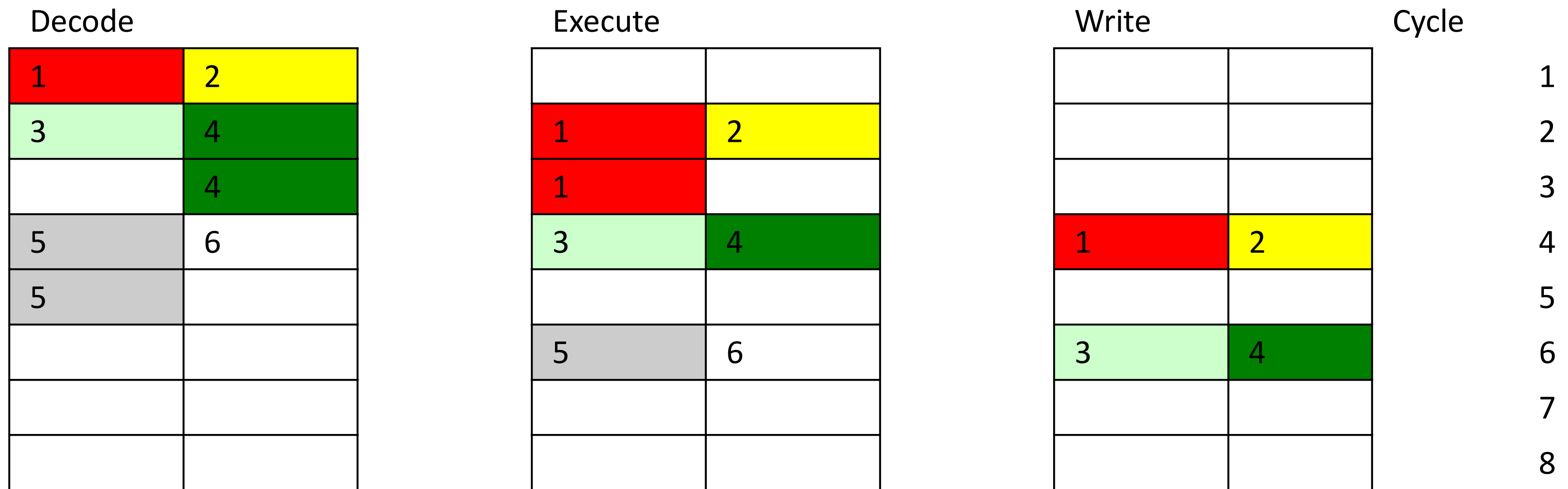
UNIVERSITAS Telkom

In-order Issue In-order Completion

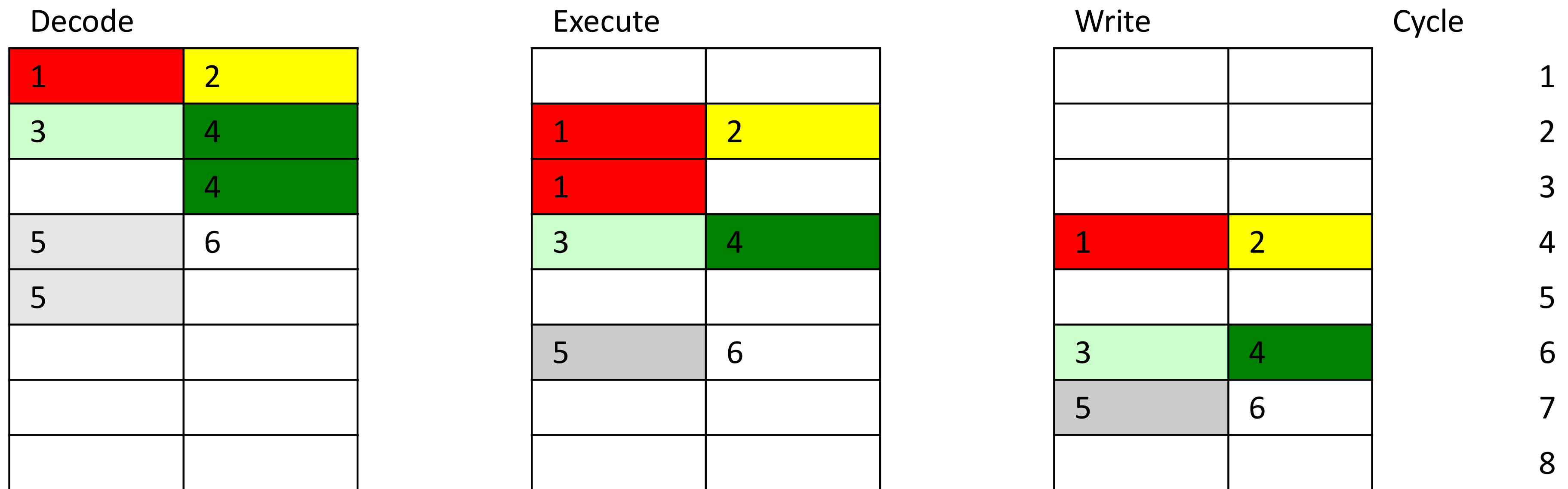
Decode		Execute		Write		Cycle
1	2					1
3	4	1	2			2
	4	1				3
5	6	3	4	1	2	4
5						5
						6
						7
						8

UNIVERSITAS Telkom

In-order Issue In-order Completion



In-order Issue In-order Completion



Cycle	Decode	Decode	Execute	Execute	Write	Write
1	1	2				
2	3	4	1	2		
3		4	1			
4	5	6	3	4	1	2
5	5					
6			5	6	3	4
7					5	6
8						

UNIVERSITAS
Telkom

In-Order Issue
Out-of-Order Completion

- Dependensi output
  - R3:= R3 + R5; (I1)
  - R4:= R3 + 1; (I2)
  - R3:= R5 + 1; (I3)
  - I2 tergantung hasil dari I1 – dependensi data
  - Jika I3 selesai sebelum I1, hasil dari I1 akan salah – dependensi output (read-write)

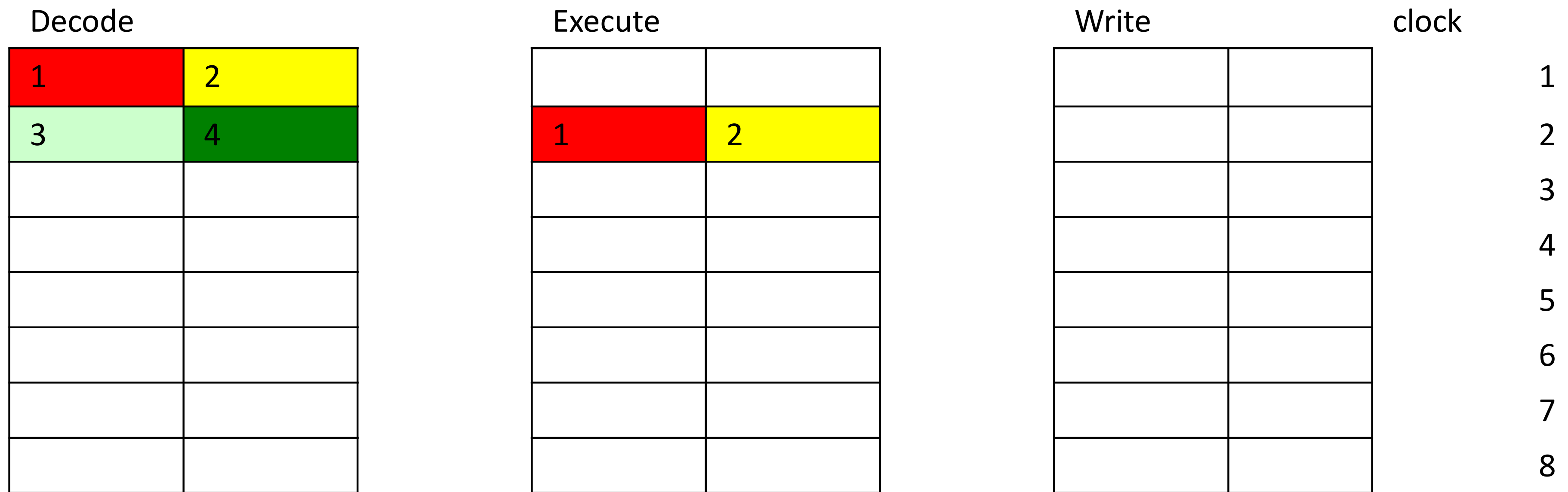
UNIVERSITAS
Telkom

In-order Issue Out-of-order Completion

Decode		Execute		Write		clock
1	2					1
						2
						3
						4
						5
						6
						7
						8

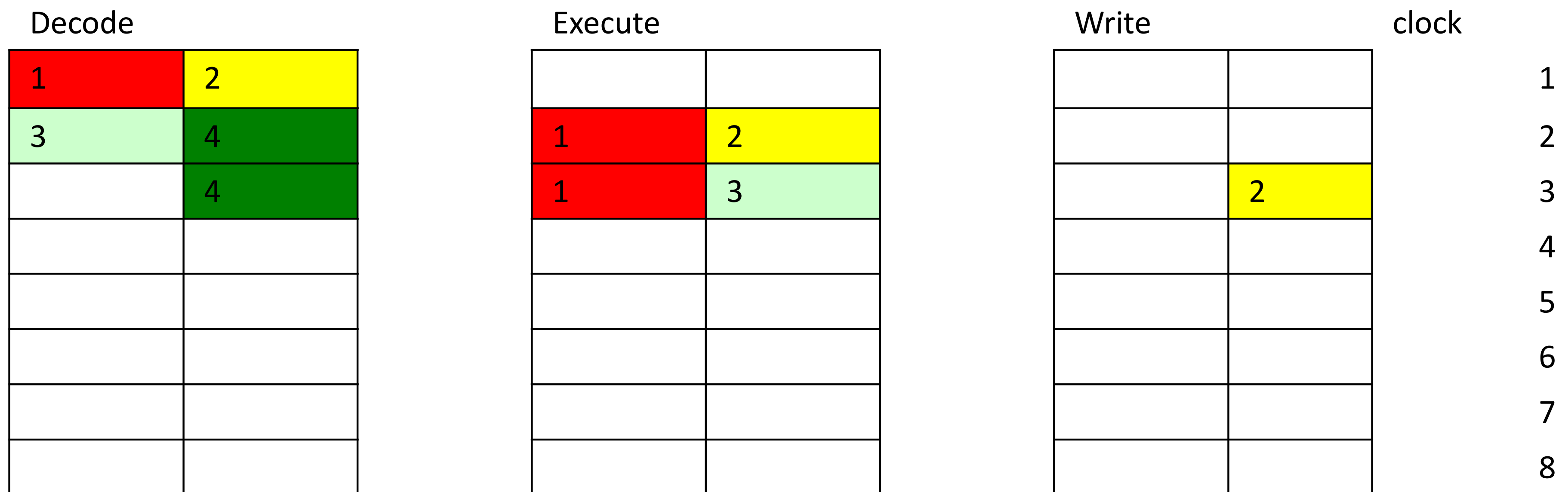
UNIVERSITAS Telkom

In-order Issue Out-of-order Completion

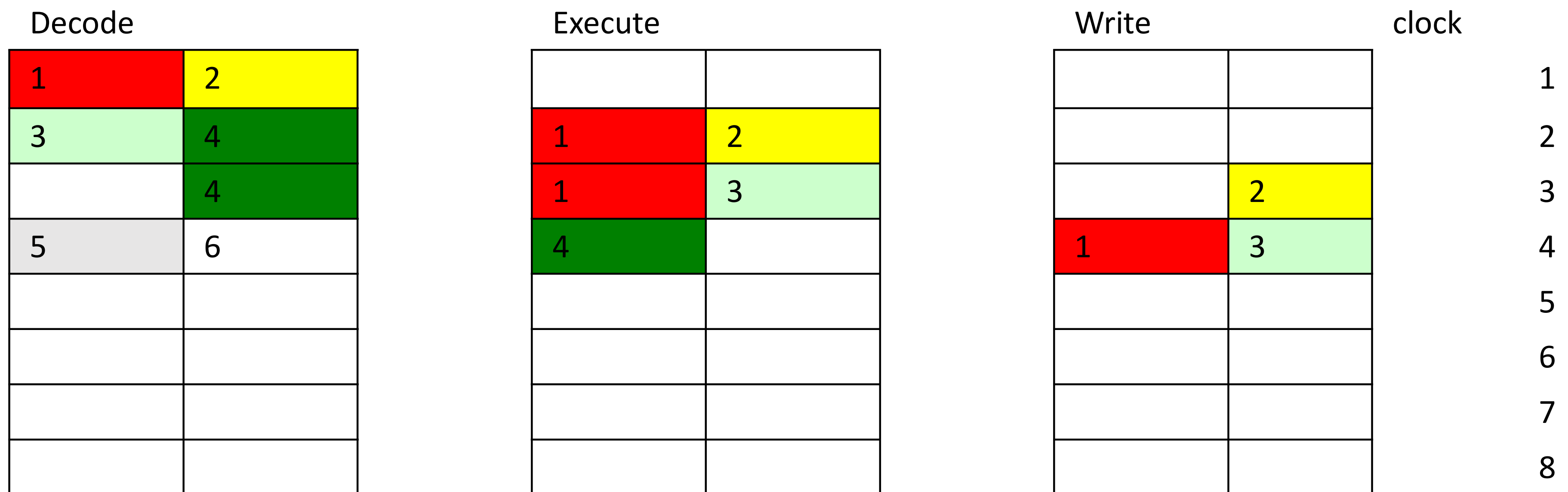


UNIVERSITAS Telkom

In-order Issue Out-of-order Completion

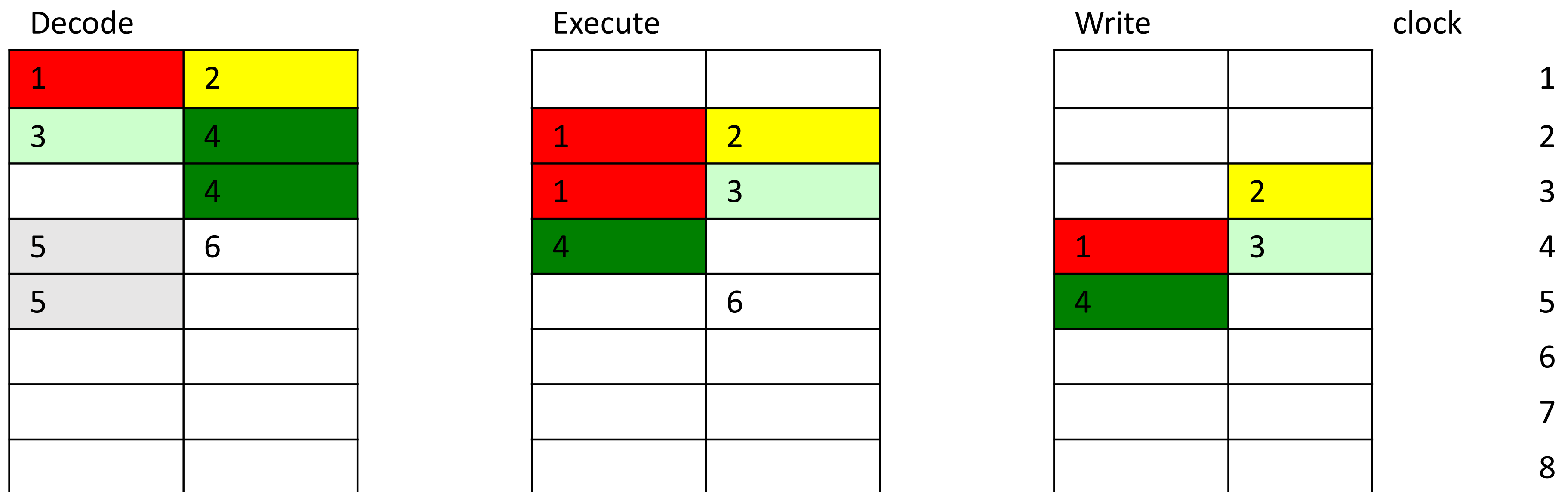


In-order Issue Out-of-order Completion



UNIVERSITAS Telkom

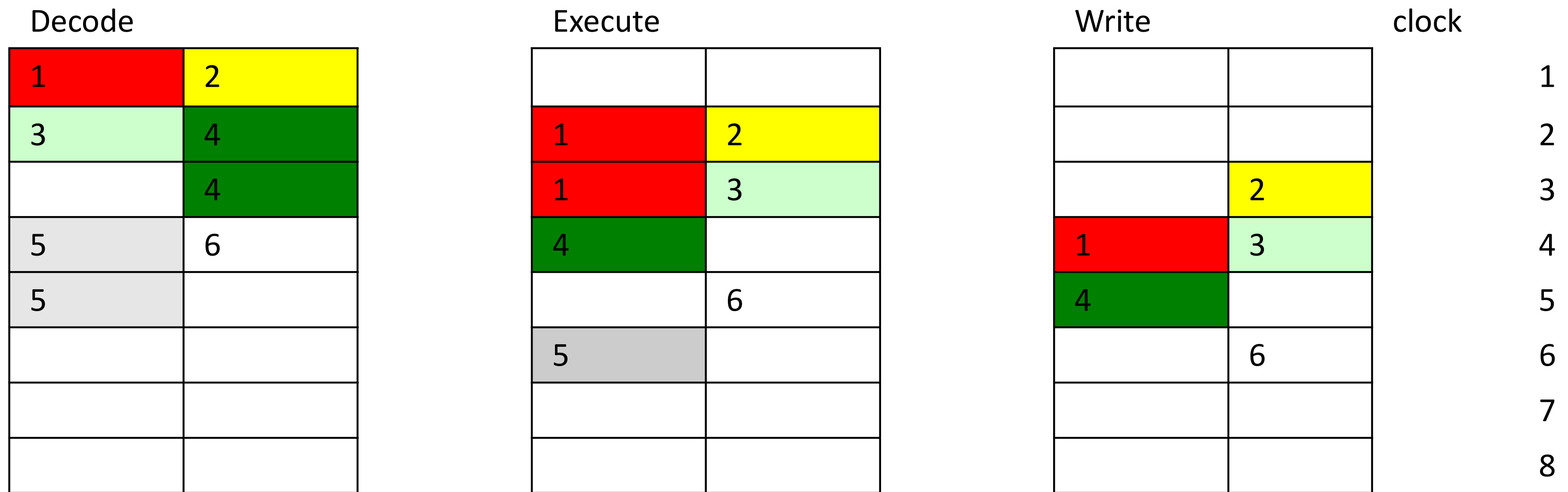
In-order Issue Out-of-order Completion



Decode		Execute		Write		clock
1	2					1
3	4	1	2			2
	4	1	3		2	3
5	6	4		1	3	4
5			6	4		5
						6
						7
						8

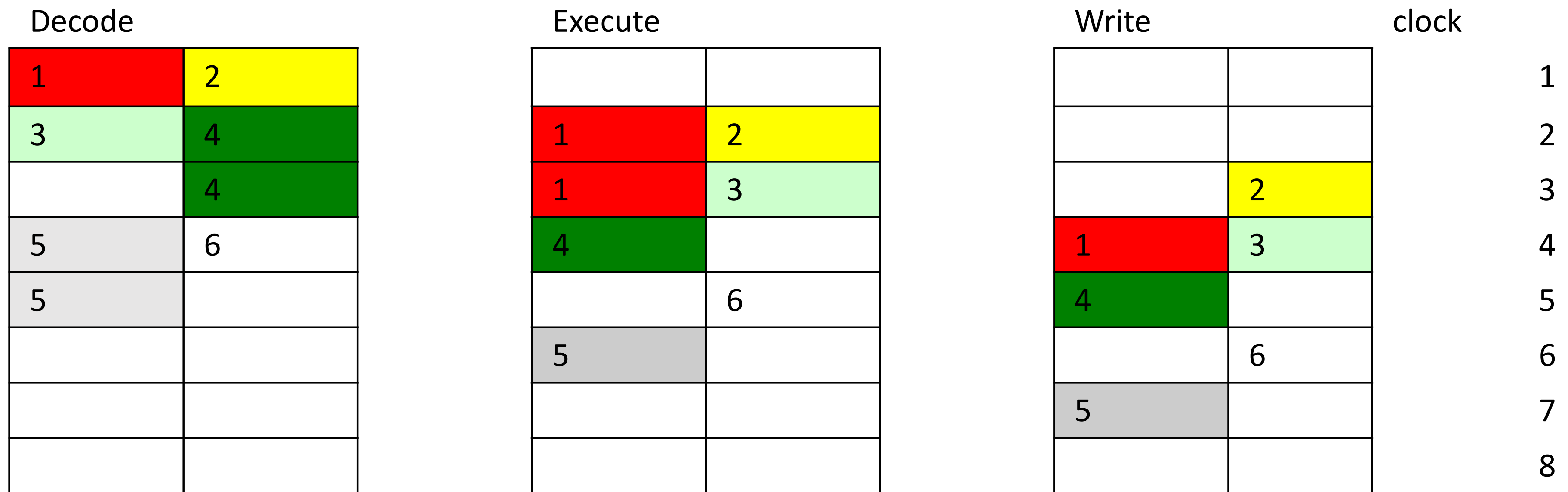
UNIVERSITAS Telkom

In-order Issue Out-of-order Completion



Decode		Execute		Write		clock
1	2					1
3	4	1	2			2
	4	1	3		2	3
5	6	4		1	3	4
5			6	4		5
		5			6	6
						7
						8

In-order Issue Out-of-order Completion



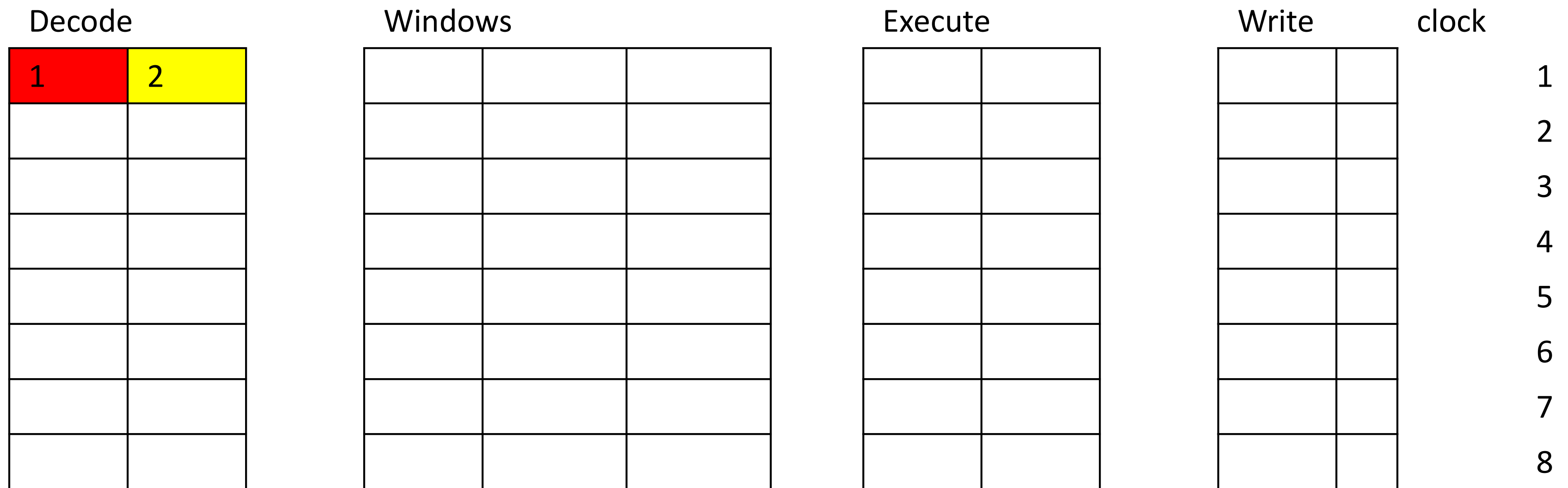
UNIVERSITAS Telkom

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

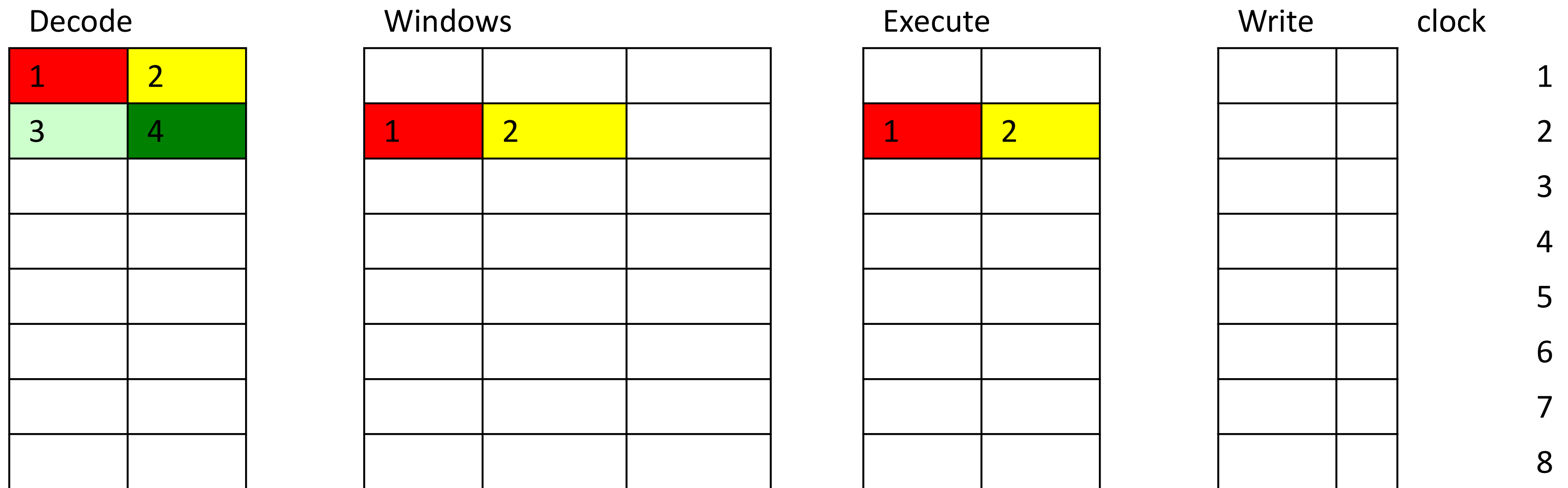
- Pipeline untuk decode dengan pipeline untuk eksekusi terpisah
- Dapat terus melakukan fetch dan decode sampai pipeline ini penuh
- Ketika sebuah unit fungsional menjadi tersedia sebuah instruksi dapat di eksekusi
- Dikarenakan instruksi telah di decode sebelumnya, prosesor dapat melihat instuksi-instruksi yang akan dijalankan (look ahead)

UNIVERSITAS Telkom

Outof-order issue out of-order completion

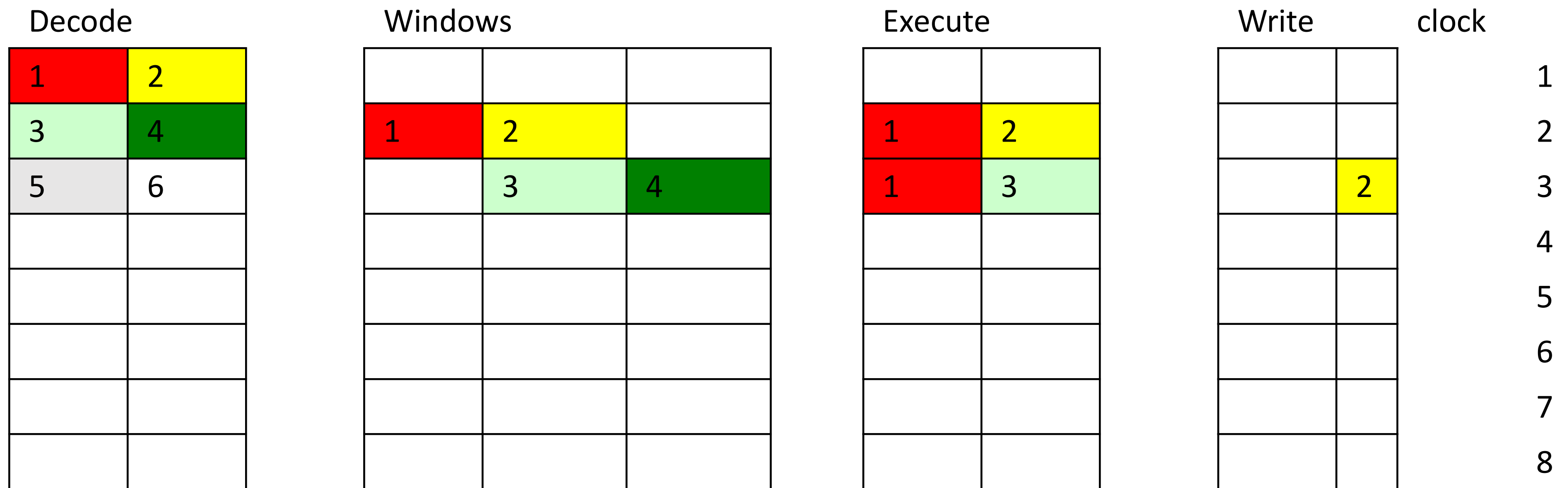


Outof-order issue out of-order completion



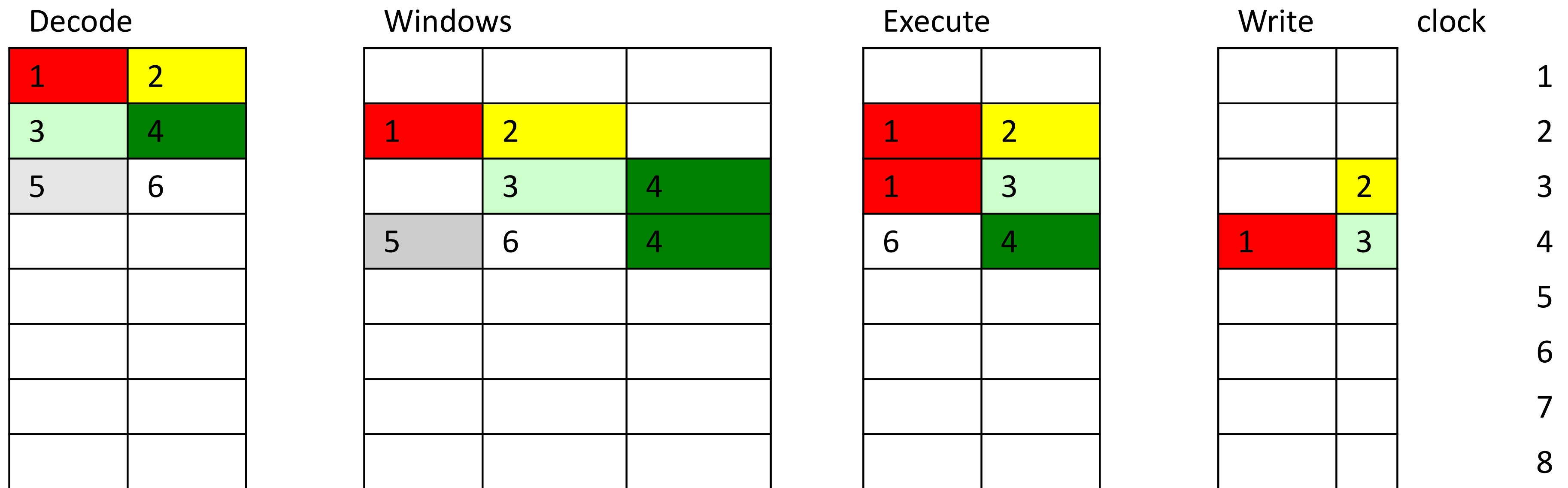
UNIVERSITAS Telkom

Outof-order issue out of-order completion



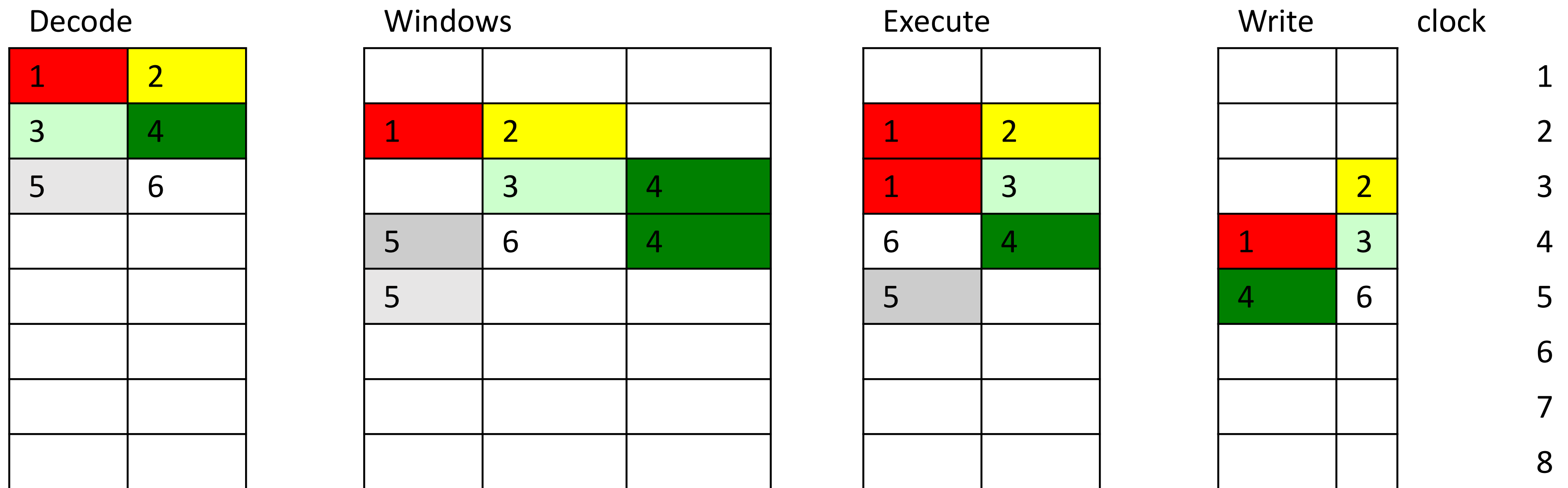
UNIVERSITAS Telkom

Outof-order issue out of-order completion



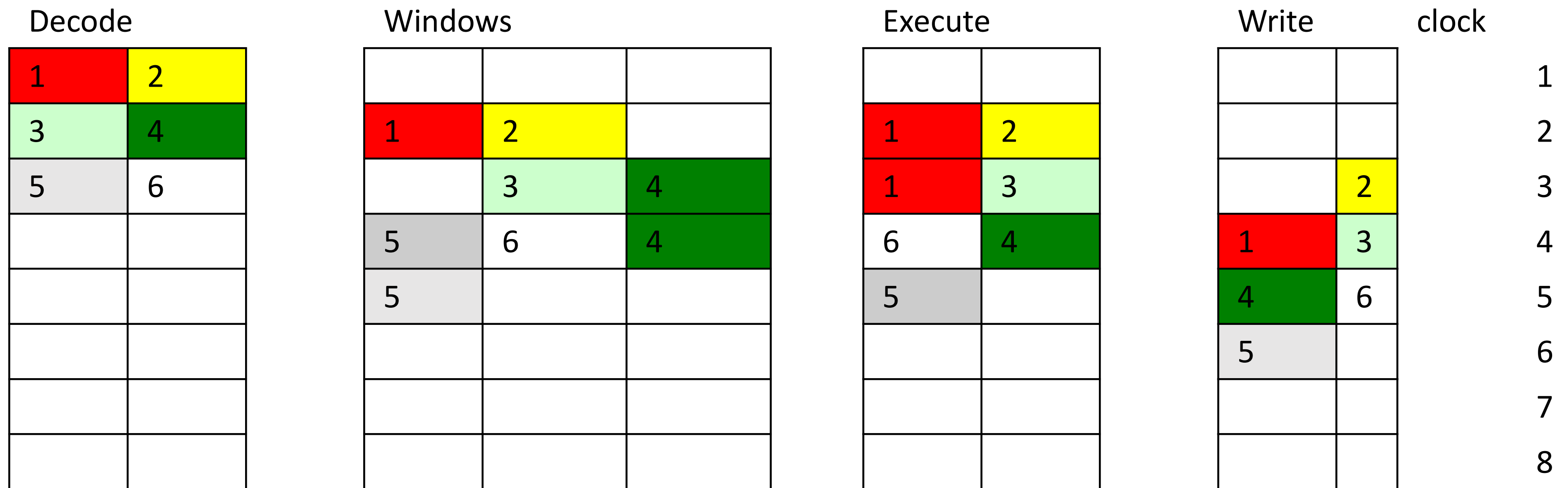
UNIVERSITAS Telkom

Outof-order issue out of-order completion



UNIVERSITAS Telkom

Outof-order issue out of-order completion



Decode		Windows			Execute		Write		clock
1	2								1
3	4	1	2		1	2			2
5	6		3	4	1	3		2	3
		5	6	4	6	4	1	3	4
		5			5		4	6	5
							5		6
									7
									8

UNIVERSITAS
Telkom

Antidependensi

- R3:=R3 + R5; (I1)
- R4:=R3 + 1;   (I2)
- R3:=R5 + 1;   (I3)
- R7:=R3 + R4; (I4)

- R3:=R3 + R5; (I1)
- R3:=R5 + 1;   (I3)
- R7:=R3 + R4; (I4)
- R4:=R3 + 1;   (I2)

- I3 tidak dapat selesai sebelum I2 dimulai sebagaimana I2 memerlukan nilai di R3 dan I3 mengubah R3

UNIVERSITAS Telkom

Pengubahan Nama Register *(Renaming)*

- Dependesi output dan antidependensi terjadi dikarenakan isi register tidak menyatakan urutan yang benar sesuai dengan program
- Bisa menyebabkan stall di pipeline
- Register harus dialokasikan dinamis
  - Contoh: registers tidak diberi nama khusus

UNIVERSITAS Telkom

Contoh Pengubahan Nama Register

- R3b:=R3a + R5a (I1)
- R4b:=R3b + 1 (I2)
- R3c:=R5a + 1 (I3)
- R7b:=R3c + R4b (I4)
- RXy menyatakan register logik di instruksi
- RX adalah alokasi register di hardware

UNIVERSITAS Telkom

Pendugaan Cabang (*Branch Prediction)*

- 80486 melakukan fetch instruksi setelah pencabangan dengan instruksi tujuan pencabangan
- Memberikan delay 2 siklus jika pencabangan jadi dilakukan

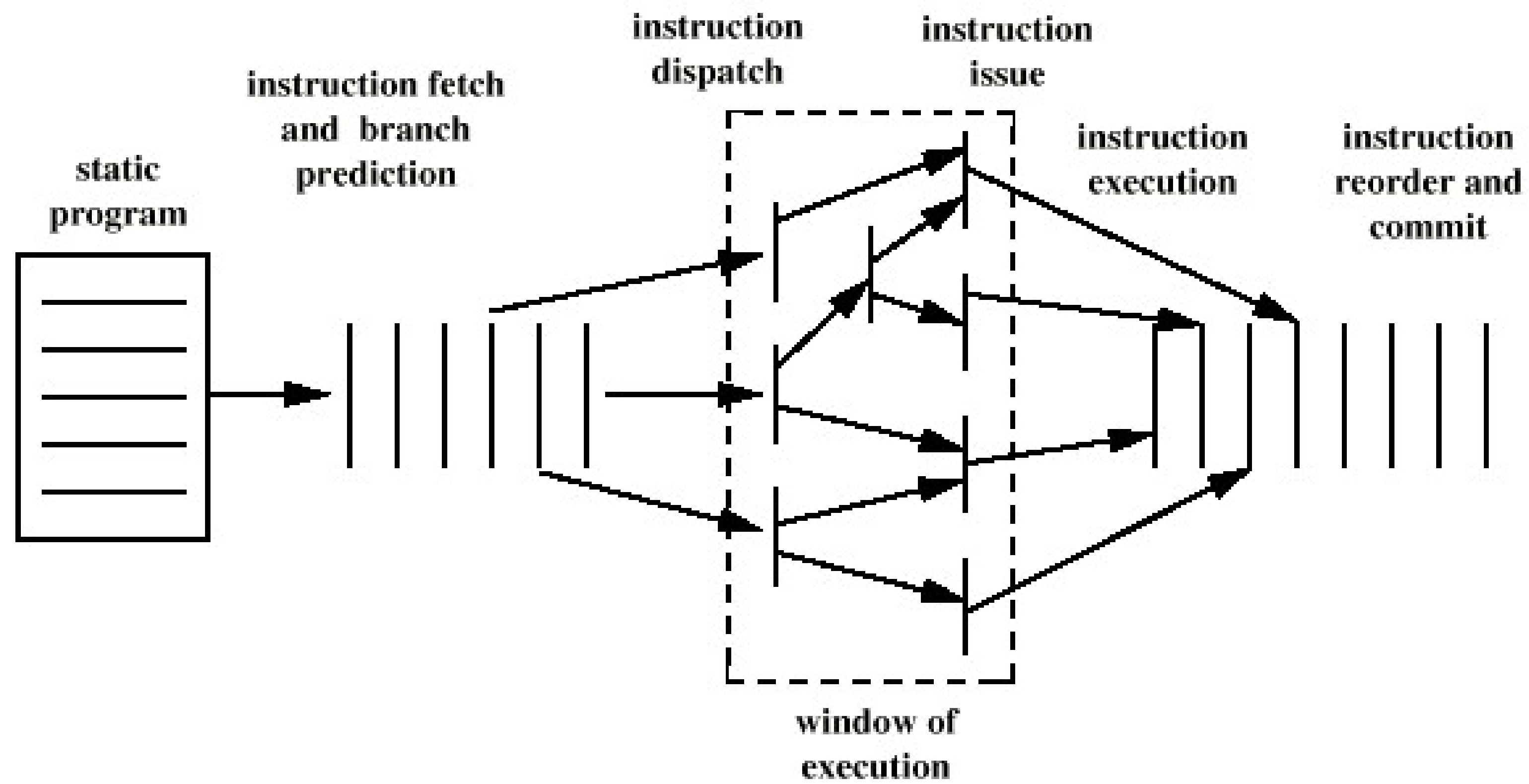
UNIVERSITAS
Telkom

RISC – Pencabangan Tertunda

- Hitung hasil dari pencabangan sebelum instruksi yang tidak berguna di pre-fetched
- Selalu eksekusi instruksi tunggal segera setelah pencabangan
- Pipeline selalu dibuat penuh sambil mem-fetch aliran instruksi baru
- Tidak begitu bagus untuk superscalar
  - Banyak instruksi harus dieksekusi di slot delay
  - Masalah dependensi instruksi
- Kembali ke pendugaan pencabangan

UNIVERSITAS Telkom

Eksekusi Superscalar



UNIVERSITAS Telkom

Penerapan Superscalar

- Fetch beberapa instruksi secara bersamaan
- Logika digunakan untuk menentukan dependensi yang melibatkan nilai-nilai register
- Mekanisme untuk mempertukarkan nilai-nilai ini
- Mekanisme untuk melakukan beberapa instruksi secara paralel
- Penyediaan resource untuk eksekusi paralel dari beberapa instruksi

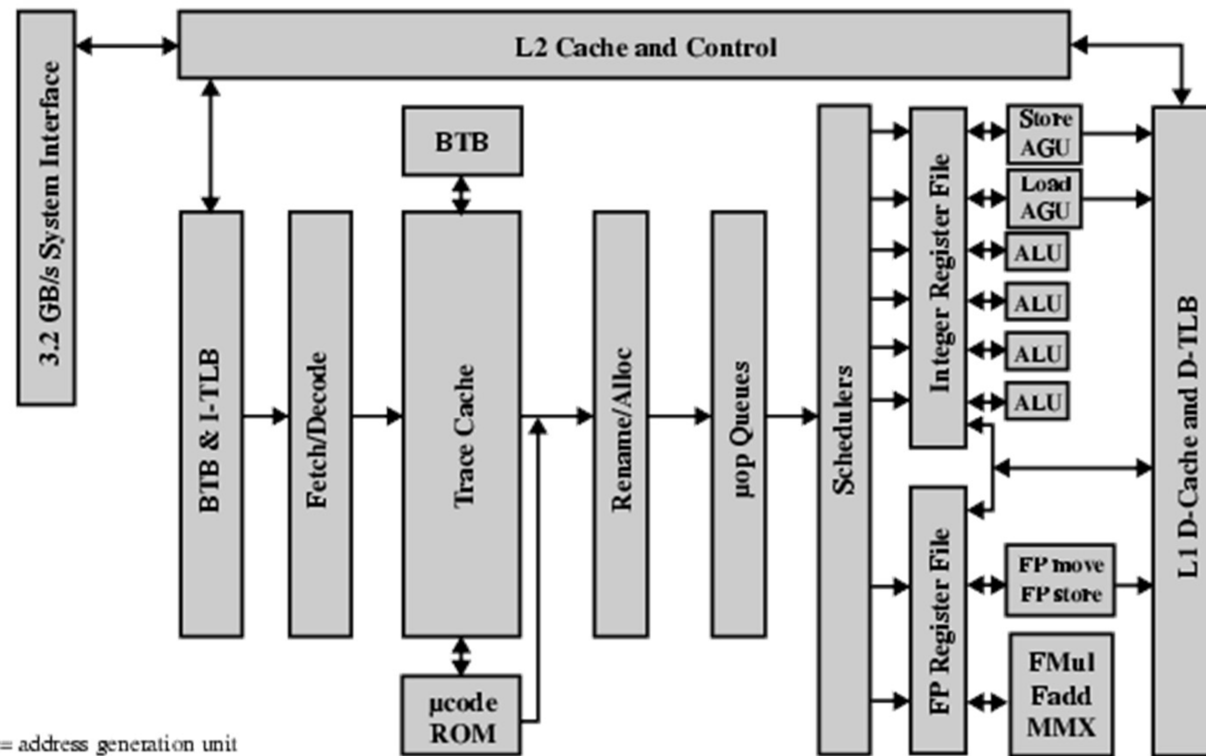
UNIVERSITAS Telkom

Pentium 4

- 80486 - CISC
- Pentium – beberpa komponen superscalar
  - Dua unit eksekusi interger terpisah
- Pentium Pro – Superscalar penuh
- Model-model berikutnya dirancang dengan peningkatan superscalar

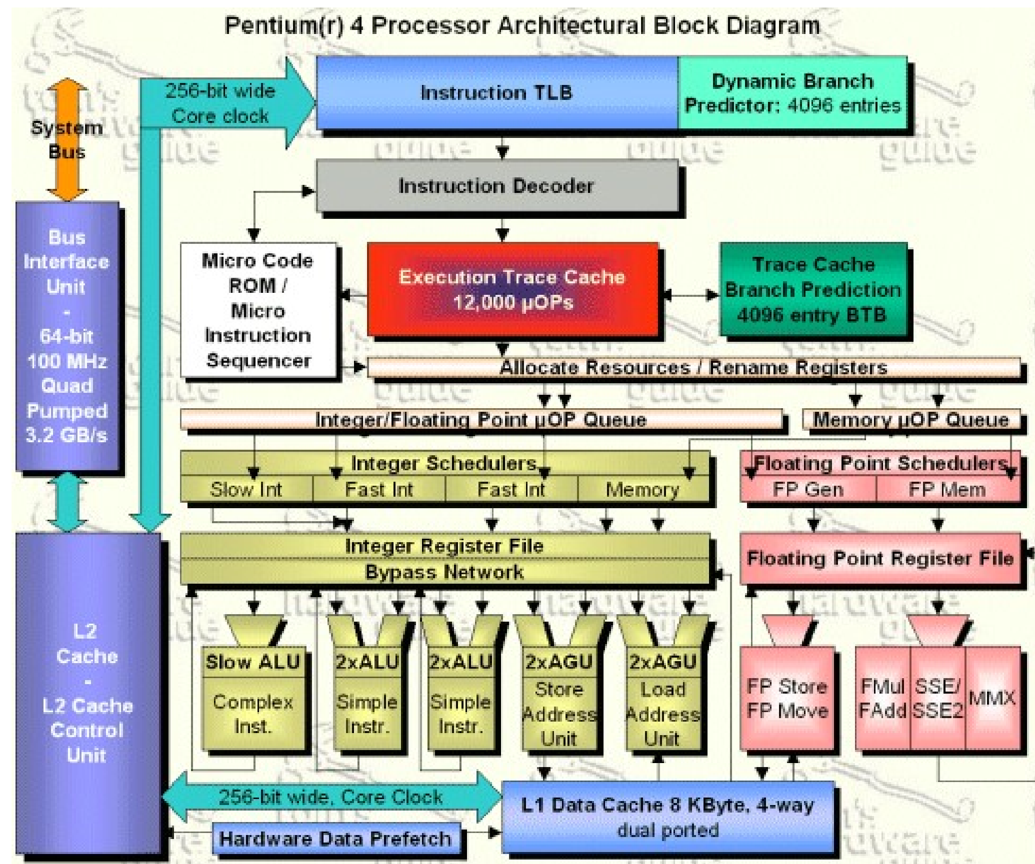
UNIVERSITAS Telkom

Pentium 4 Block Diagram



Universitas Telkom

Instruction Stream



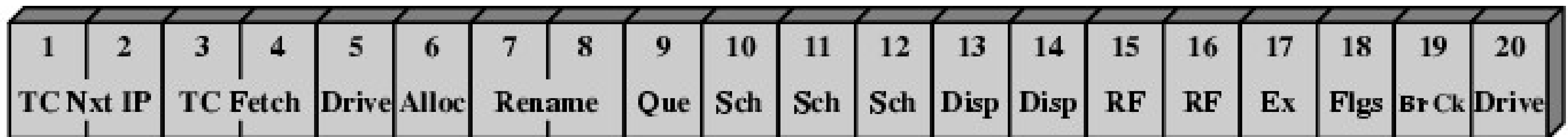
UNIVERSITAS Telkom

Operasi Pentium 4

- Fetch instruksi dari memori berurut sesuai program
- Terjemahkan instruksi ke satu atau lebih instruksi RISC panjang tetap (micro-operations)
- Eksekusi micro-ops di pipeline superscalar
    - micro-ops dapat dieksekusi out of order
- Kirim hasil dari micro-ops ke set register dalam urutan aliran program aslinya
- Tampak luar CISC dengan inti RISC
- Inti RISC pipeline terdiri dari paling tidak 20 tingkat
    - Beberapa micro-ops memerlukan lebih dari satu tingkat eksekusi
        - Pipeline lebih panjang

UNIVERSITAS
Telkom

Pipeline Pentium 4



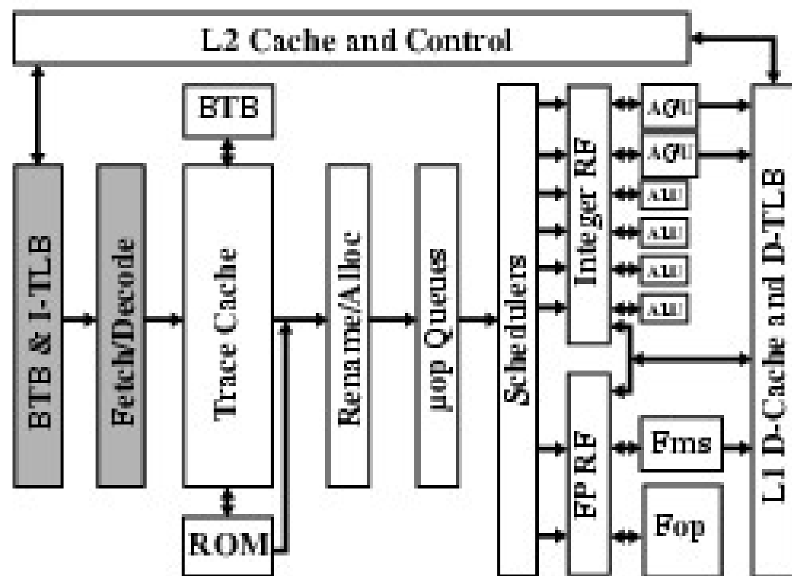
TC Next IP = trace cache next instruction pointer
TC Fetch = trace cache fetch
Alloc = allocate

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

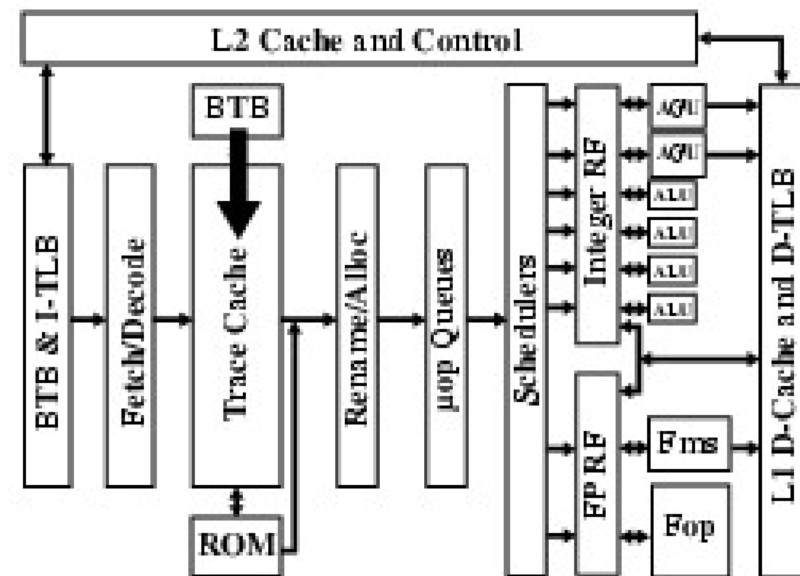
RF = register file
Ex = execute
Flgs = flags
Br Ck = branch check

UNIVERSITAS Telkom

Operasi Pipeline Pentium 4 (1)



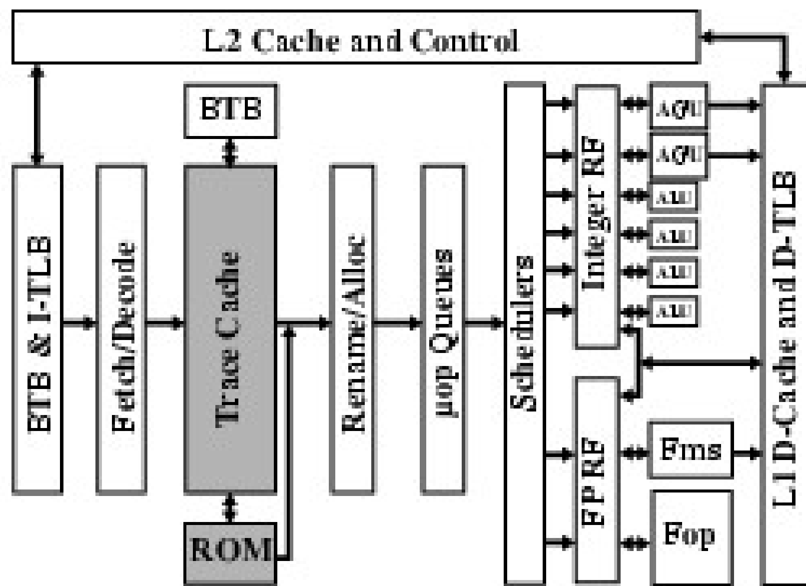
(a) Generation of micro-ops



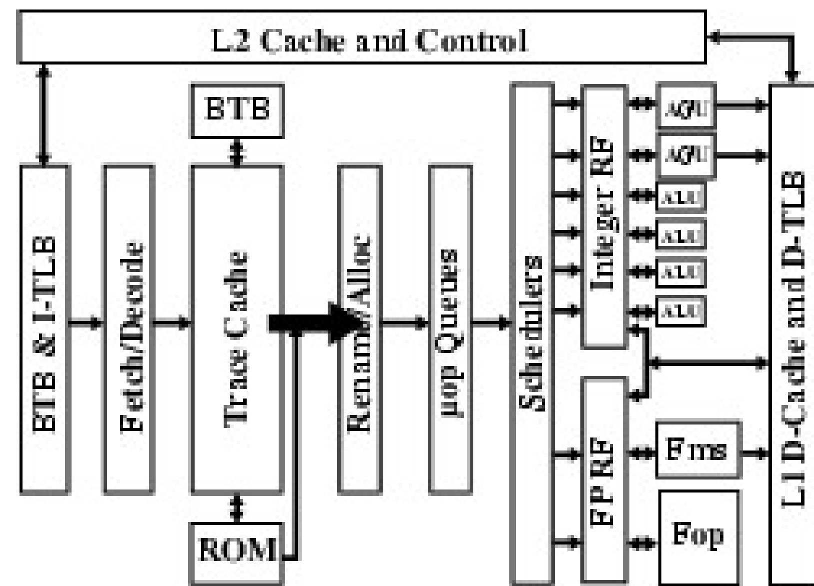
(b) Trace cache next instruction pointer

UNIVERSITAS Telkom

Operasi Pipeline Pentium 4 (2)



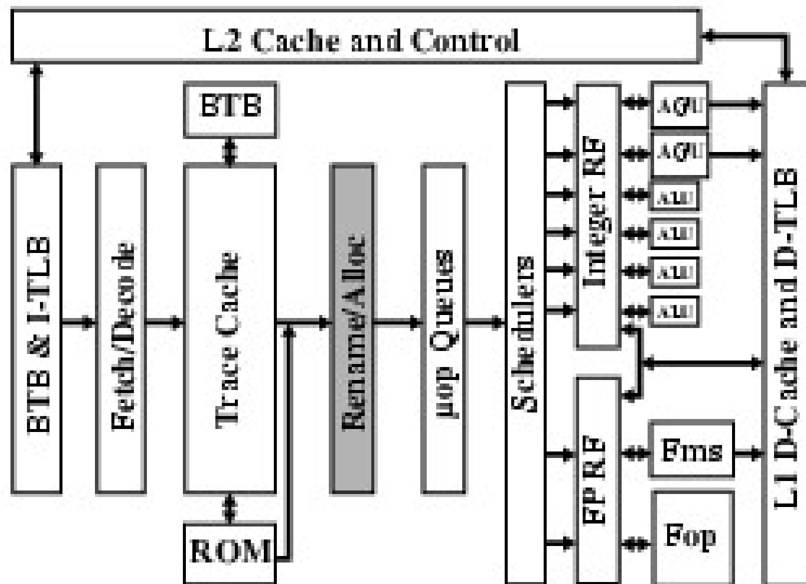
(c) Trace cache fetch



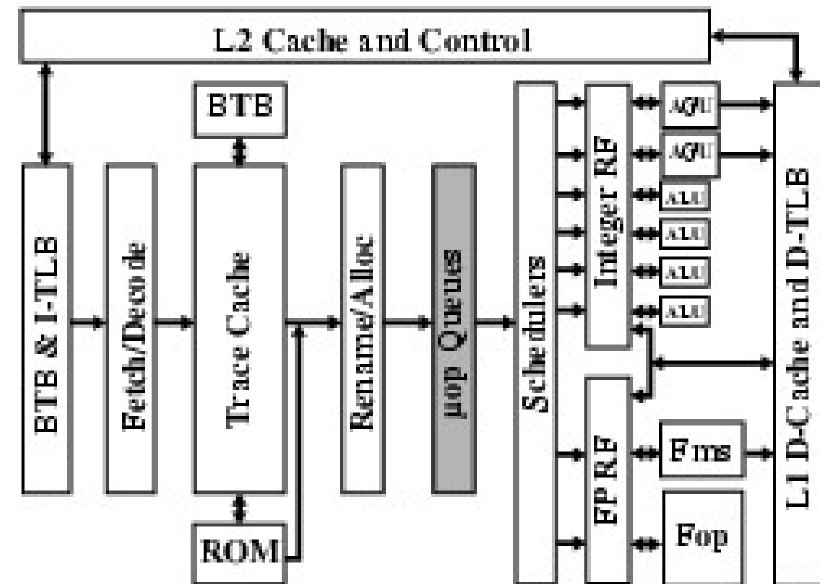
(d) Drive

UNIVERSITAS Telkom

Operasi Pipeline Pentium 4 (3)



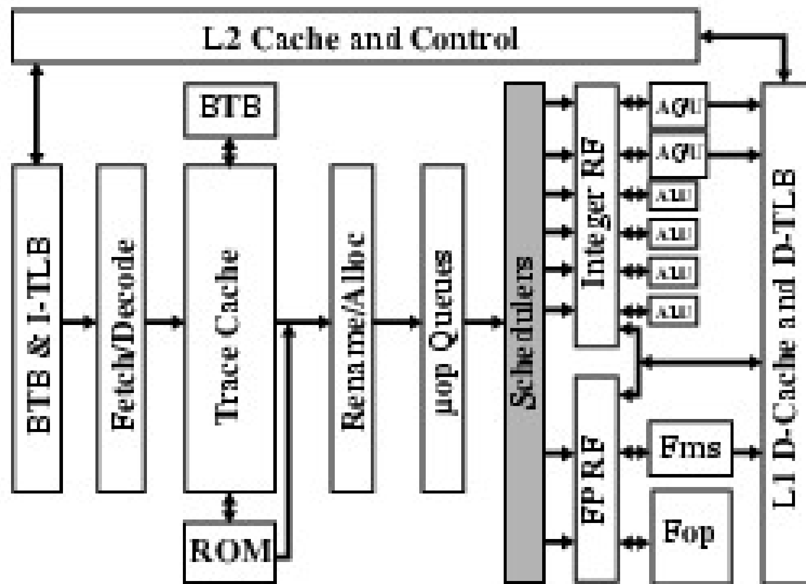
(e) Allocate; Register renaming



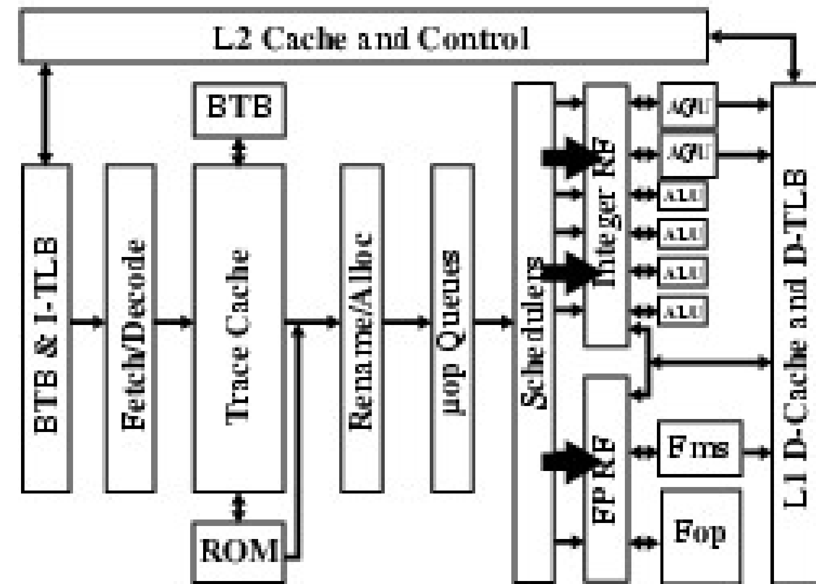
(f) Micro-op queuing

UNIVERSITAS Telkom

Operasi Pipeline Pentium 4 (4)



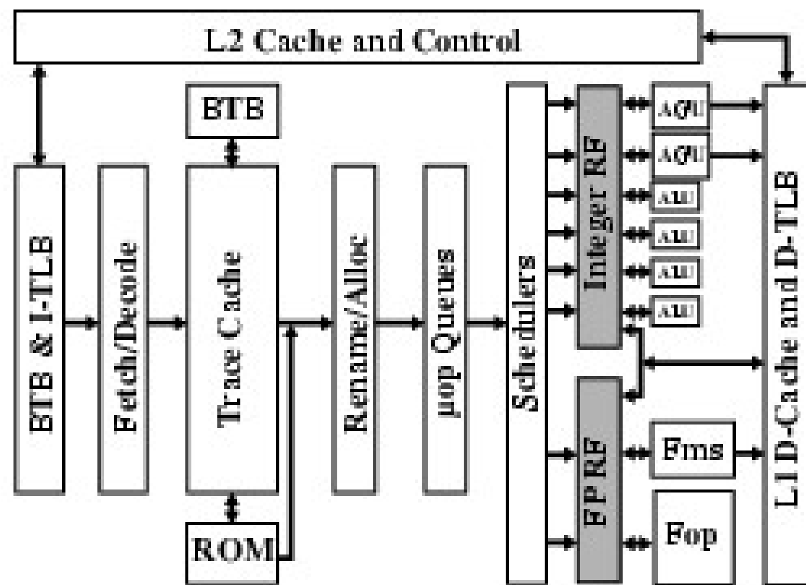
(g) Micro-op scheduling



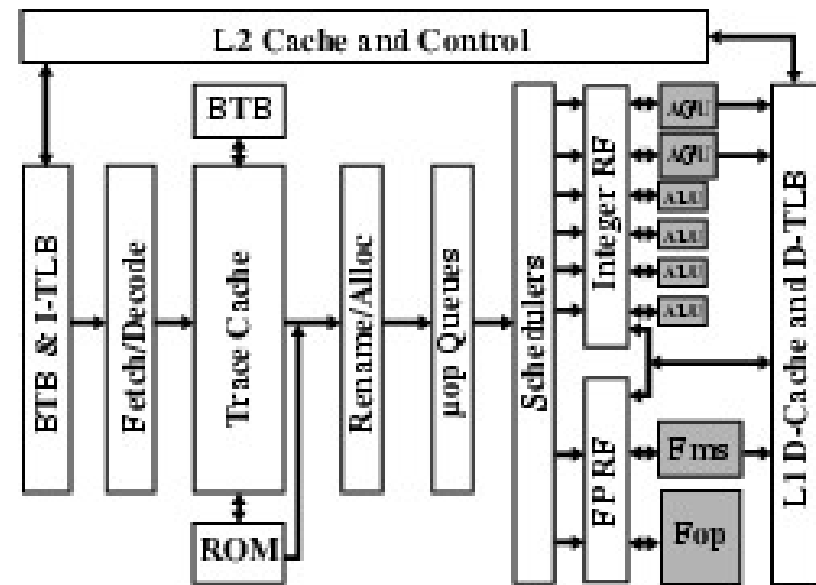
(h) Dispatch

UNIVERSITAS Telkom

Operasi Pipeline Pentium 4 (5)



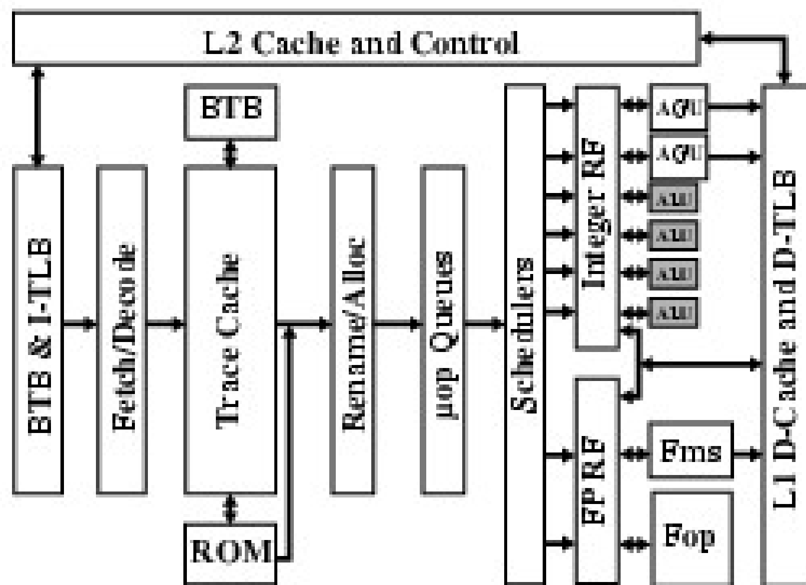
(i) Register file



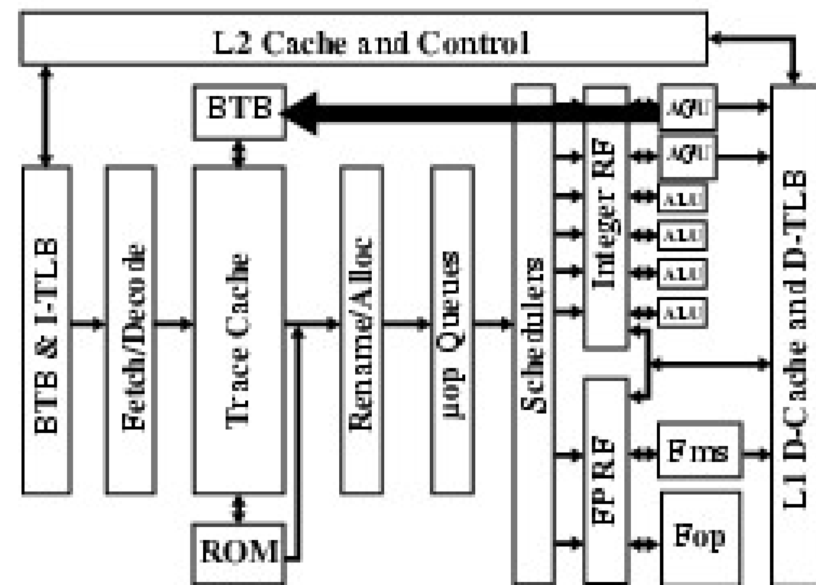
(j) Execute; flags

UNIVERSITAS Telkom

Operasi Pipeline Pentium 4 (6)



(k) Branch check



(l) Branch check result

UNIVERSITAS Telkom