Organisasi Komputer

Materi 7: Pipelining

I Nyoman Kusuma Wardana
Sistem Informasi STMIK STIKOM Bali

Materi Perkuliahan

- **Pendahuluan**
- **Instruction Pipeline**

STIKOM BALI

PENDAHULUAN

PENDAHULUAN

- Terdapat 2 teknik dasar utk meningkatkn kecepatan eksekusi dr prosesor, sbb:

1. Meningkatkan **kecepatan clock** → shg menurunkan waktu eksekusi
2. Meningkatkan **jumlah instruksi** yg dpt dikerjakan scr bersama

- ***Pipeline*** → salah satu contoh solusi kedua

PENDAHULUAN

Konsep umum pipeline

- ***Pipeline*** → teknik utk **membagi tugas** (***task***) dr suatu urutan eksekusi menjadi bbrp ***sub-task***
- Setiap **sub-task** dijalankan berdasarkan **fungsi unit** tertentu
- Setiap **unit** terkoneksi **scr seri** dan semuanya dijalankan secara **simultan**

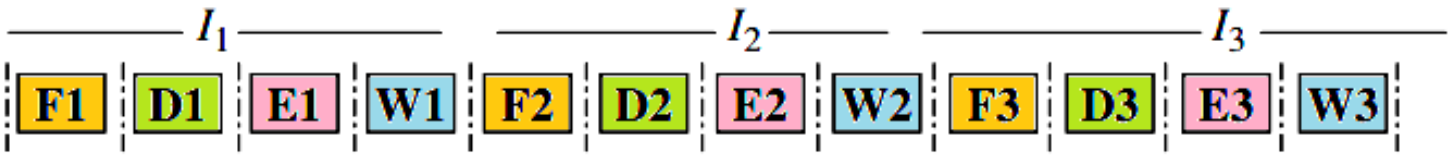
PENDAHULUAN

- ***Pipeline*** → **meningkatkan kinerja** (performa) drpd urutan scr tradisional

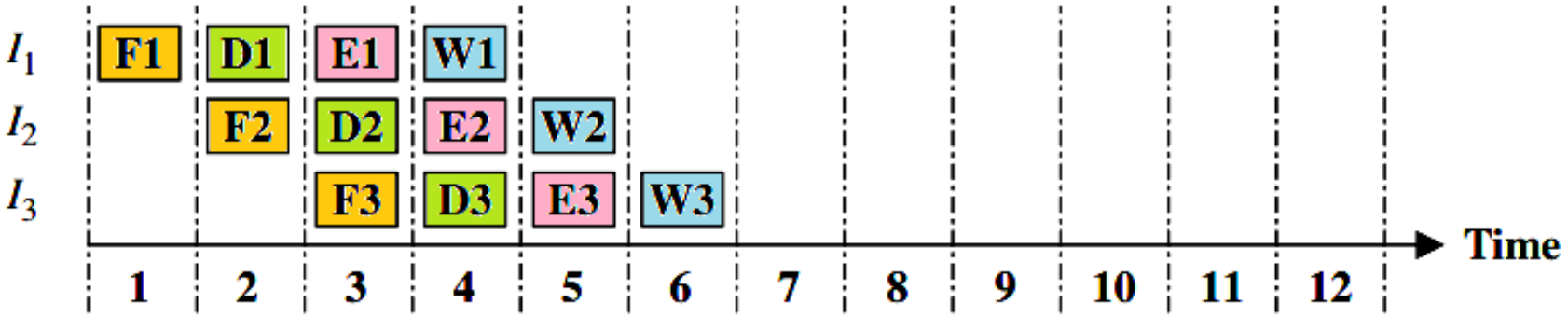
- Misal terdapat bbrp tahap, sbb:
- ***Fetching*** → **membaca** instruksi
- ***Decoding*** → **menterjemahkan** instruksi
- ***Execution*** → **menjalankan** instruksi
- ***Writing*** → **menulis** (menyimpan) hasil

PENDAHULUAN

- Bandingkan penggunaan **pipeline** & **proses sekuensial** (tradisional), sbb:



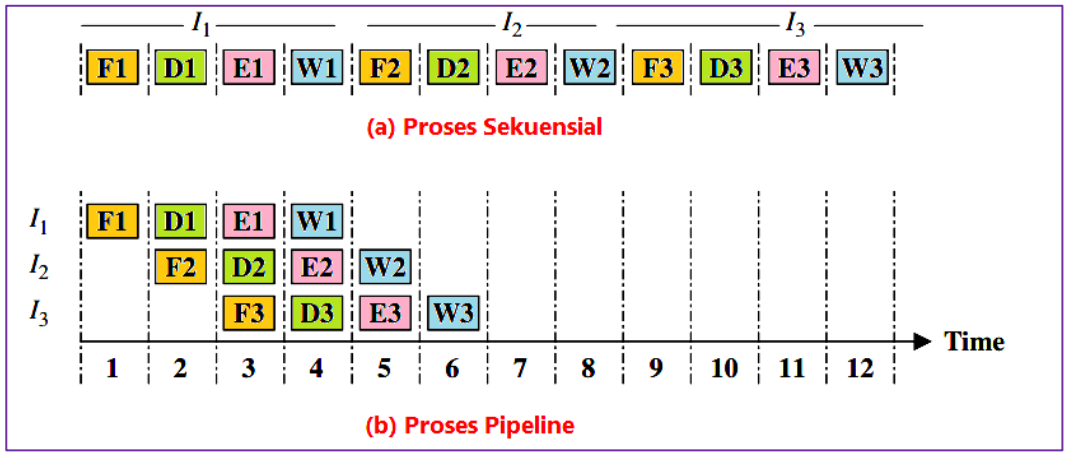
(a) Proses Sekuensial



(b) Proses Pipeline

Pipeline & proses sekuensial

PENDAHULUAN

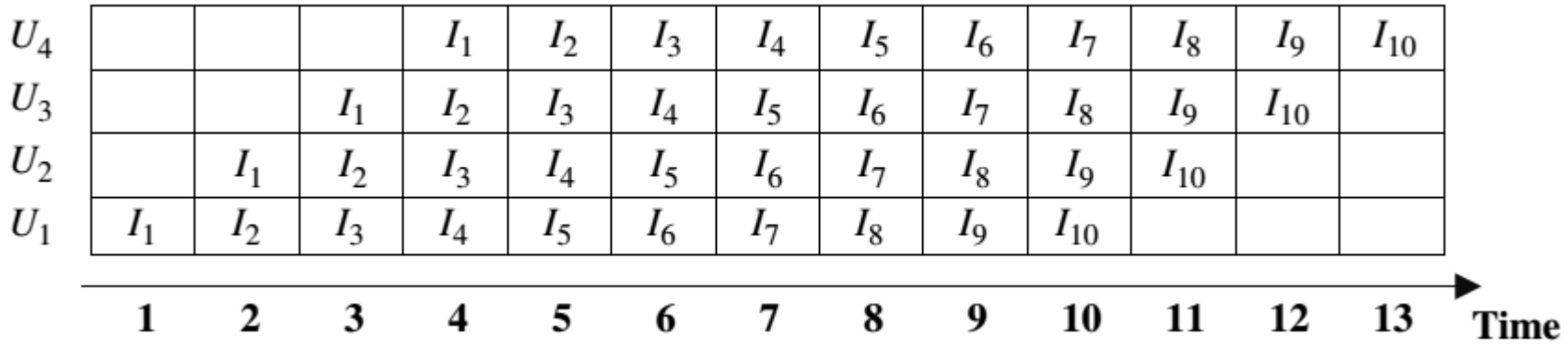


- ***Proses sekuensial*** → total waktu yg dibutuhkan utk memproses 3 instruksi (I1, I2, I3) = 12 unit waktu
- ***Pipeline*** → hanya 6 unit
- Kemungkinan menghemat **sampai 50%** dr total wkt eksekusi dpt diperoleh

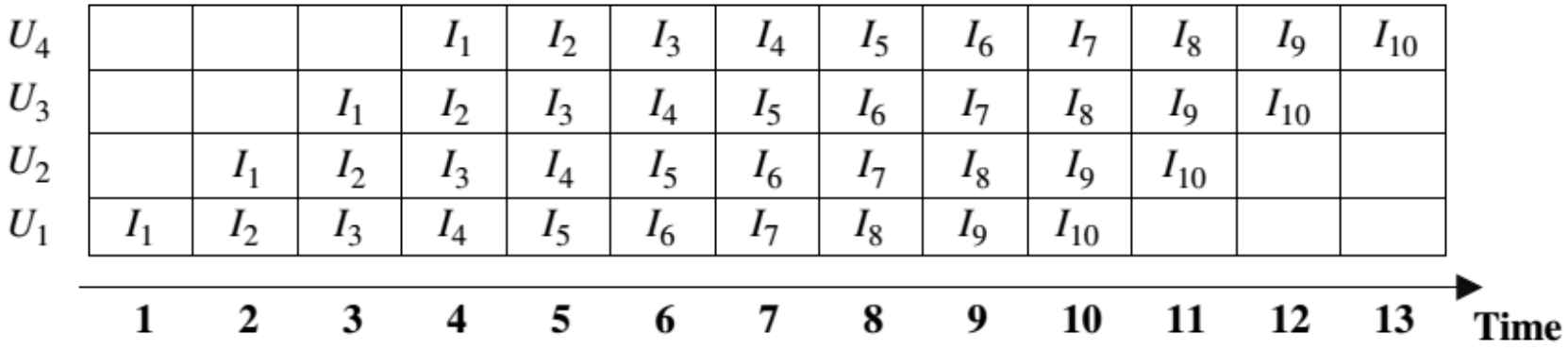
PENDAHULUAN

- ***Gantt's chart*** → **tabel waktu** yg digunakan utk mengukur kinerja pipeline dlm urutan tugas (task)

- Amati contoh, sbb:



PENDAHULUAN



- **Sumbu vertikal** = sub-unit (dlm hal ini sjumlah 4)
- **Sumbu horizontal** = unit waktu

- Pd tabel **Gantt** → diasumsikan bahwa waktu (T) yg dibtuhkan setiap sub-unit utk mnjalankan tugas adlh **sama** (dikenal sbg ***time unit***)

PENDAHULUAN

- Trdapat 3 parameter utk **mengukur kinerja dr pipeline**:
  1. ***Speed-up*** **S(n)**
  2. ***Throughput*** **U(n)**
  3. ***Efficiency*** **E(n)**

- Asumsi bahwa dlm analisis ini:

  Unit time ***T = t*** unit

PENDAHULUAN

Speed-up, S(n)

- Asumsi **jmlh task** (instruksi) = ***m***, dgn menggunakan ***n-tahap*** *pipeline*
- Maka utk menyelesaikan m task diperlukan waktu selama ***n + m – 1***

- $$Speed - up\ S(n) = \frac{\boldsymbol{Waktu\ menggunakan\ proses\ sekuensial}}{\boldsymbol{Waktu\ menggunakan\ pipeline}}$$

$$= \frac{\boldsymbol{m \times n \times t}}{\boldsymbol{(n+m-1) \times t}}$$

$$= \frac{\boldsymbol{m \times n}}{\boldsymbol{n+m-1}}$$

PENDAHULUAN

Throughput, U(n)

- Adalah **jumlah task** yg dieksekusi **per unit waktu**

- $Throughput\ U(n) = \frac{\boldsymbol{Jumlah\ task}}{\boldsymbol{Unit\ waktu}}$

$$= \frac{\boldsymbol{m}}{\boldsymbol{(n+m-1)\times t}}$$

PENDAHULUAN

Efficiency, E(n)

- Adalah **perbandingan kecepatan aktual terhadap kecepatan maksimum**

- $$Efficiency\ E(n) = \frac{\boldsymbol{Speed-up}}{\boldsymbol{n}}$$
$$= \frac{\boldsymbol{m}}{\boldsymbol{n+m-1}}$$

STIKOM BALI

INSTRUCTION PIPELINE

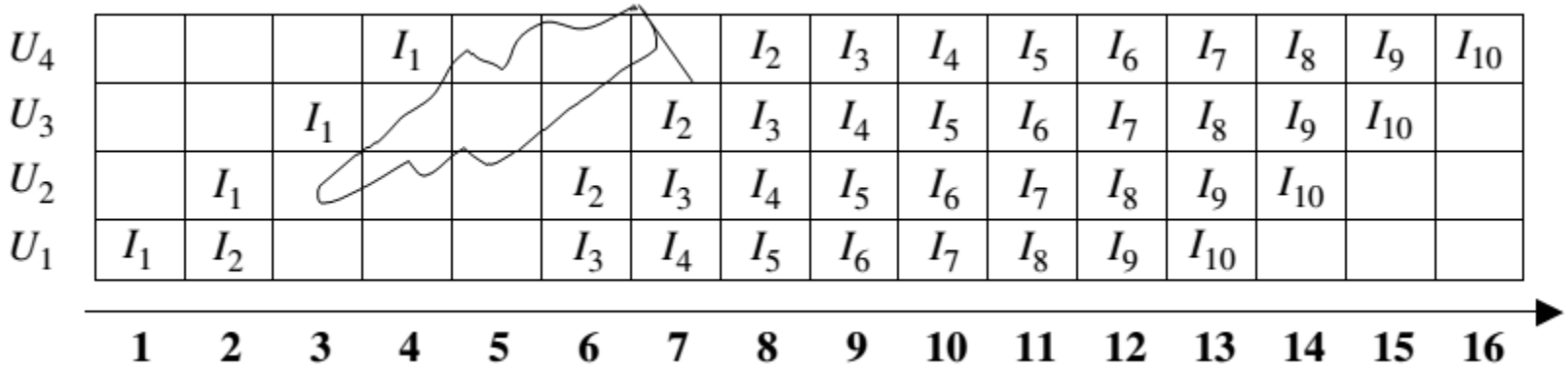
Instruction Pipeline

Pipeline Stall

- ***Pipeline*** mnjd ***stall*** jika 1 unit **membutuhkan waktu lebih lama** dlm menjalankan fungsinya
- Keadaan ini **memaksa** unit lain utk ***idle***
- **Contoh**: ketika terjadi ***cache miss***, yaitu ketika data yg diambil dr memori cache tidak diperoleh

Instruction Pipeline

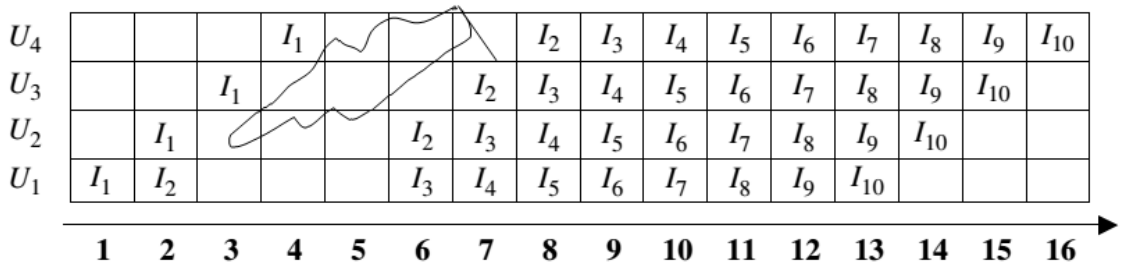
- Amati contoh berikut:



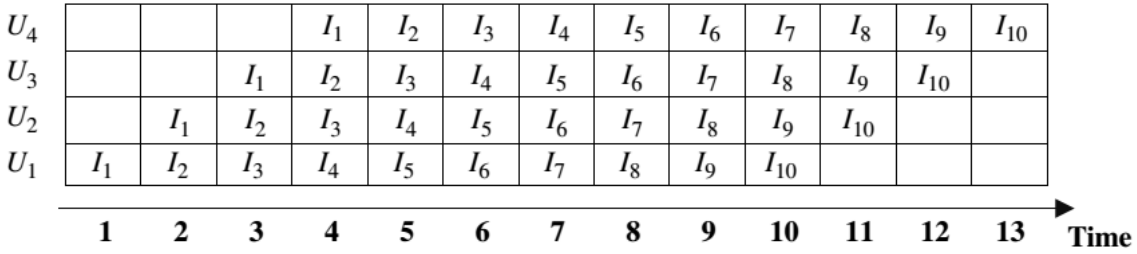
- Pembacaan **I2** membutuhkan **waktu ekstra**
- Akibatnya, pembacaan utk I3 akhirnya mengalami ***delay***
- Situasi ini disbt sbg pipeline ***bubble*** atau pipeline ***hazard***

PENDAHULUAN

- Pada contoh ini, ***pipeline hazard*** menyebabkan waktu eksekusi sebanyak **16 unit waktu**



- Bandingkan dgn tanpa hazard, yaitu sebanyak **13 unit waktu**



Instruction Pipeline

- Terdapat **berbagai alasan** mengapa pipeline hazard bisa terjadi
- Terdapat **dua yg utama** penyebab pipeline hazard, yaitu sbb:
  1. ***Instruction dependency***
  2. ***Data dependency***

Operasi yg **tepat** utk pipeline→ jika operasi yg dijalankan pd satu tahap **TIDAK TERGANTUNG** dr operasi yg dijalankan oleh tahap lainnya

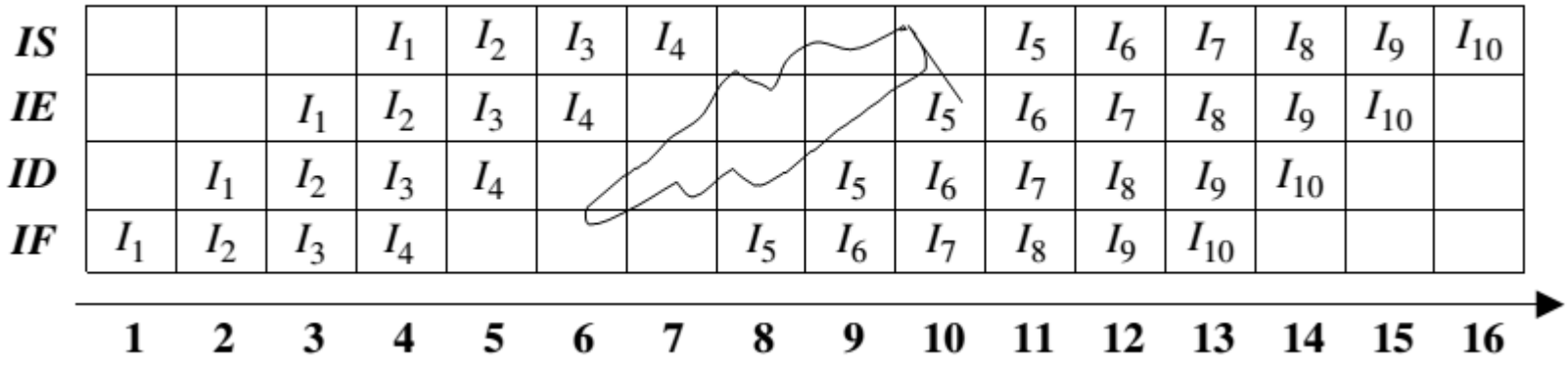
Instruction Pipeline

Instruction dependency

- **Instruction dependency** → pembacaan instruksi saat ini **bergantung** dr hasil eksekusi instruksi sebelumnya
- **Contoh**: lompat jika hasil negatif (***branch if negative***).
- Pada kasus ini, Instruksi selanjutnya **tidak akan dieksekusi** sblm diketahui hasil skrg positif/negatif

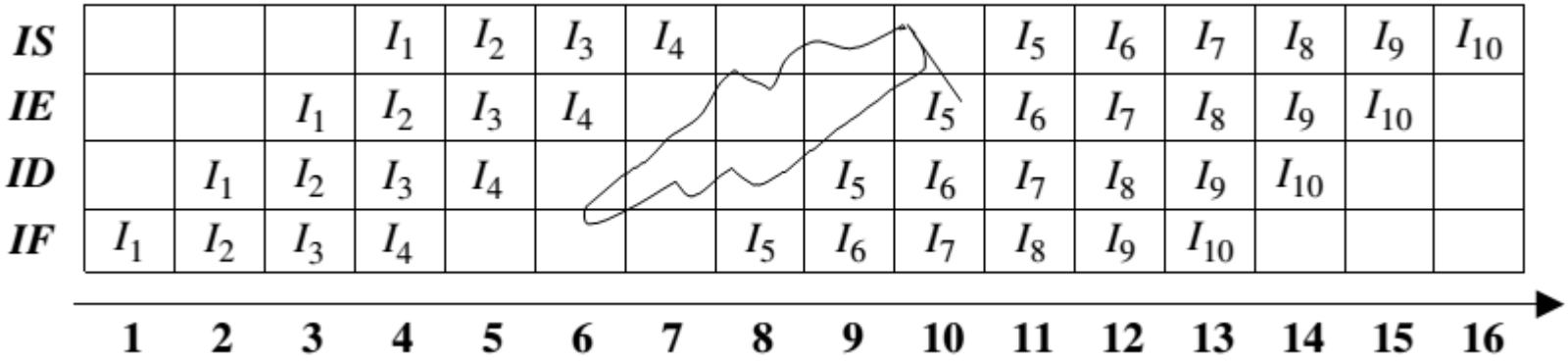
Instruction Pipeline

Contoh, amati gb berikut:



- **IF** : ***instruction fetch***
- **ID** : ***instruction decode***
- **IE** : ***instruction execut***e
- **IS** : ***instruction result store***

Instruction Pipeline



- Asumsi pd I4 terjadi percabangan.
- Jika syarat terpenuhi, maka akan terjd percabangan → pipeline stall sampai hasil eksekusi diperoleh

Instruction Pipeline

Data dependency

- **Data dependency** → terjadi jika operan sumber dr intsruksi yg akan datang (I_i) tergantung dr hasil dr eksekusi sebelumnya (I_j), dengan $i > j$
- **Perlu dicatat** → walaupun instruksi I_i dapat dibaca, namun **operand belum tersedia sampai** hasil dr I_j **telah diperoleh** dan disimpan

Instruction Pipeline

- Berikut contoh data dependency

ADD	$R_1, R_2, R_3;$	$R_3 \leftarrow R_1 + R_2$
SL	$R_3;$	$R_3 \leftarrow SL(R_3)$
SUB	$R_5, R_6, R_4;$	$R_4 \leftarrow R_5 - R_6$

- Instruksi SL (Shift Left/geser ke kiri) tidak bisa dijalankan **sebelum** nilai R3 (dari instruksi ADD R1,R2,R3) diperoleh.

Daftar Pustaka

- Abd-El-Barr, M., El-Rewini, H., ***Fundamentals of Computer Organization and Architecture***, John Wiley&Sons, Inc.
- Stallings, W., 2010, ***Computer Organization and Architecture: Designing for Performance*** 8th edition, Prentice Hall
- Hamacher,C., Vranezic, Z., Zaky, S., Manjikian, N., 2012, ***Computer Organization and Embedded Systems*** 6th edition, McGrawHill