# Sistem Bus

(INTERKONEKSI antar BAGIAN UTAMA KOMPUTER)


**Tim Dosen COA**

**Fakultas Informatika**

**Universitas Telkom**

*September 2022*

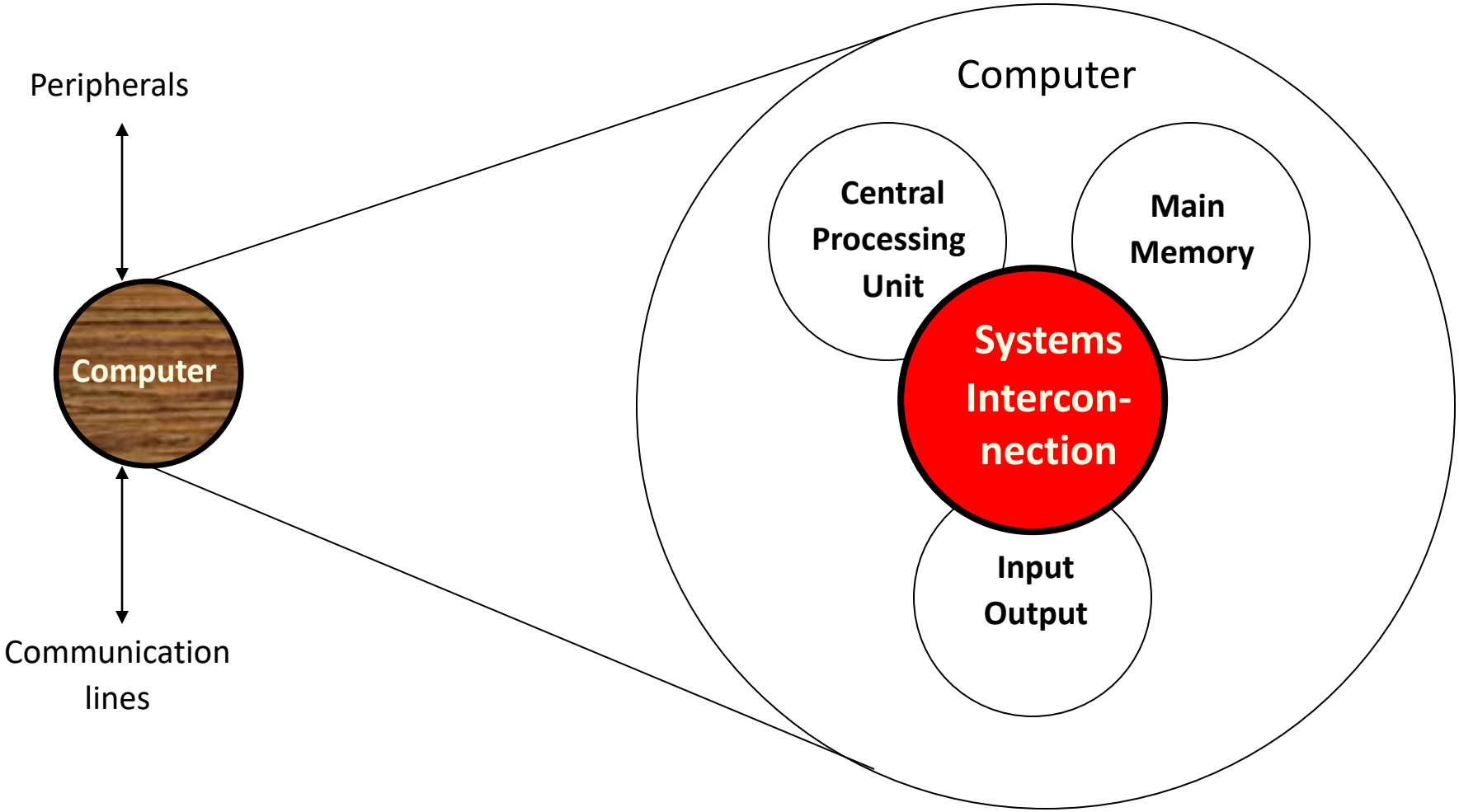
# Rencana Studi

Fakultas Informatika
School of Computing
Telkom University

| Rencana Studi | | Rincian Nilai Kuis | | | | | | | | | | | | | | | | Rincian Nilai Tugas | | | | | | | | | | Rincian Nilai Hasil Proyek | | | | | | | | | | Bobot Tiap CLO (%) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pertemuan Ke- | Materi | CLO 1 | | | CLO 2 | | | | | CLO 3 | | | | CLO 4 | | | | CLO 1 | | | CLO 2 | | | | | | CLO | CLO 3 | | | | | | CLO 4 | | | | | | | | | |
| | | Kuis (Kognitif) (20 %) | | | | | | | | | | | | | | | | Tugas Partisipatif (35%) | | | | | | | | | | Hasil Proyek (45%) | | | | | | | | | | | | | |
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 1a | 1b | 1c | 2a | 2b | 2c | 2d | 2e | 3a | 3b | 1a | 1b | 2a | 1c | 2b | 2c | 3a | 3b | 4a | 3c | 4b | 3d | 4c | 1 | 2 | 3 | 4 |
| 1 | Sistem komputer | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | | | | |
| 2 | Input/Output | - | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 14 | -- | -- | -- |
| 3 | Sistem Bus | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | | | | |
| 4 | Organisasi memori | - | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | | | | |
| 5 | Cara kerja memori utama (RAM) | - | - | - | | 1 | - | - | - | - | - | - | - | - | - | - | | - | - | - | | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | | | | |
| 6 | Memori sekunder | - | - | - | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | -- | 26 | -- | -- |
| 7 | Cara kerja *cache memory* (bag-1) | - | - | - | - | - | - | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | | | | |
| 8 | Cara kerja cache memory (bag-2) | - | - | - | - | - | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | | | | |
| 9 | Arsitektur SAP-1 | - | - | - | - | - | - | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | | | | |
| 10 | Arsitektur SAP-2 | - | - | - | - | - | - | - | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 2 | - | - | 2 | 1 | - | - | - | - | - | - | - | - | - | - | -- | -- | 30 | -- |
| 11 | Arsitektur SAP-3 | - | - | - | - | - | - | - | - | - | - | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 3 | - | - | - | 7 | 2 | | - | - | - | - | - | - | - | | | | |
| 12 | Instruksi *Extended* dan *Indirect* | - | - | - | - | - | - | - | - | - | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 7 | - | - | - | - | - | - | - | - | | | | |
| 13 | Arsitektur MIPS | - | - | - | - | - | - | - | - | - | - | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | - | - | - | - | - | - | - | | | | |
| 14 | Instruksi MIPS | - | - | - | - | - | - | - | - | - | - | - | - | - | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 1 | - | - | - | - | - | -- | -- | -- | 30 |
| 15 | Assembly MIPS (bag-1) | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 2 | - | - | - | | | | |
| 16 | Assembly MIPS (bag-2) | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 9 | 7 | - | | | | |

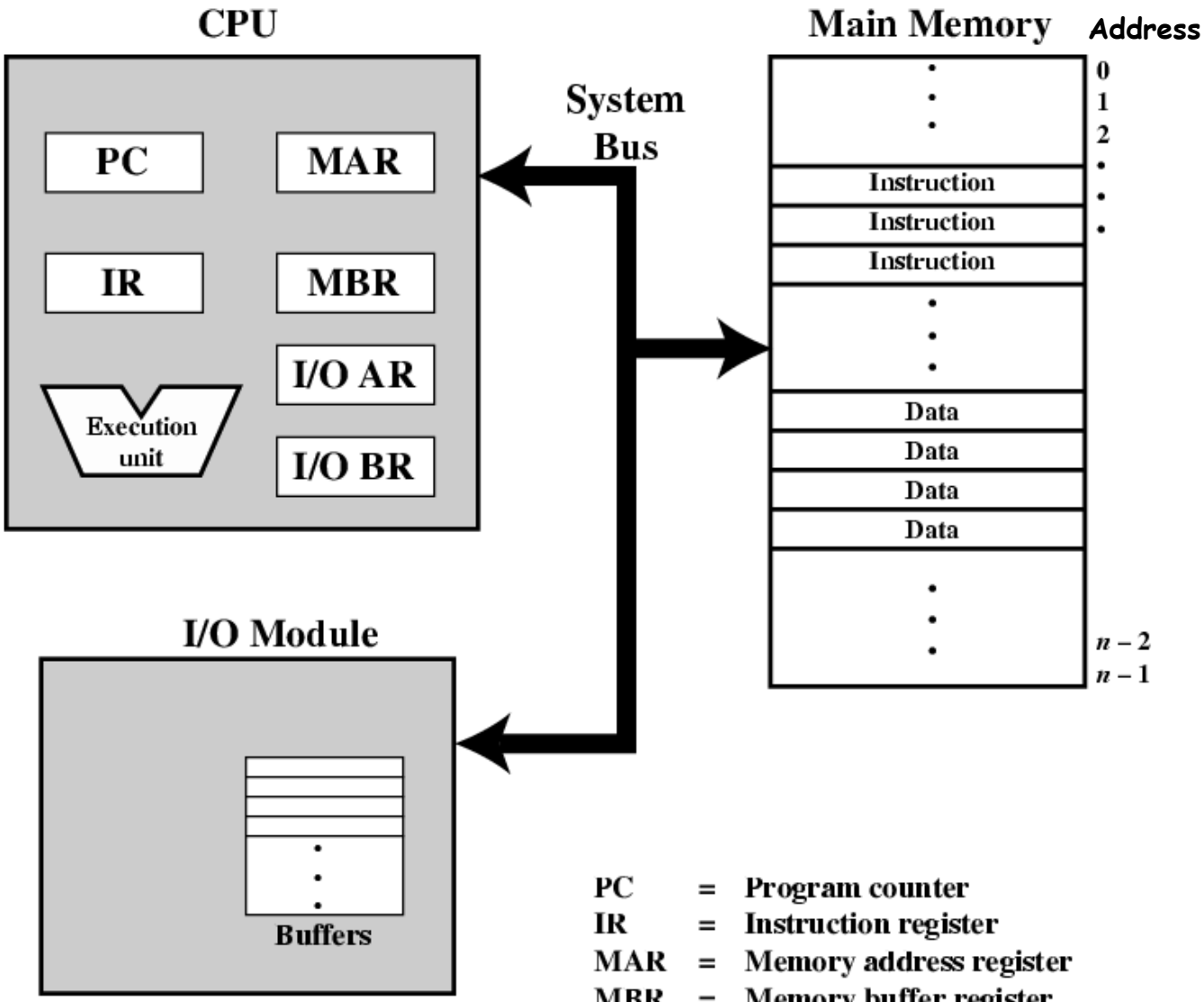
# Part 1: Bagaimana cara 4 komponen utama komputer berinteraksi?

# Struktur Komputer – Interkoneksi Sistem



# Komponen Utama Komputer (1)





# Komponen Utama komputer (2)



- MAR
  - Tempat untuk menampung alamat memori berikutnya yang akan dibaca/ditulis
- MBR
  - Tempat untuk menampung data yang akan ditulis ke memori atau data yang akan dibaca dari memori
- I/O AR
  - Tempat untuk menampung alamat *device* yang akan dikontrol
- I/O BR
  - Digunakan untuk menampung data yang dipertukarkan antara *device* dengan CPU
- IR
  - Menyimpan instruksi yang baru saja diambil (*fetched*)
- PC
  - Menyimpan alamat instruksi berikutnya

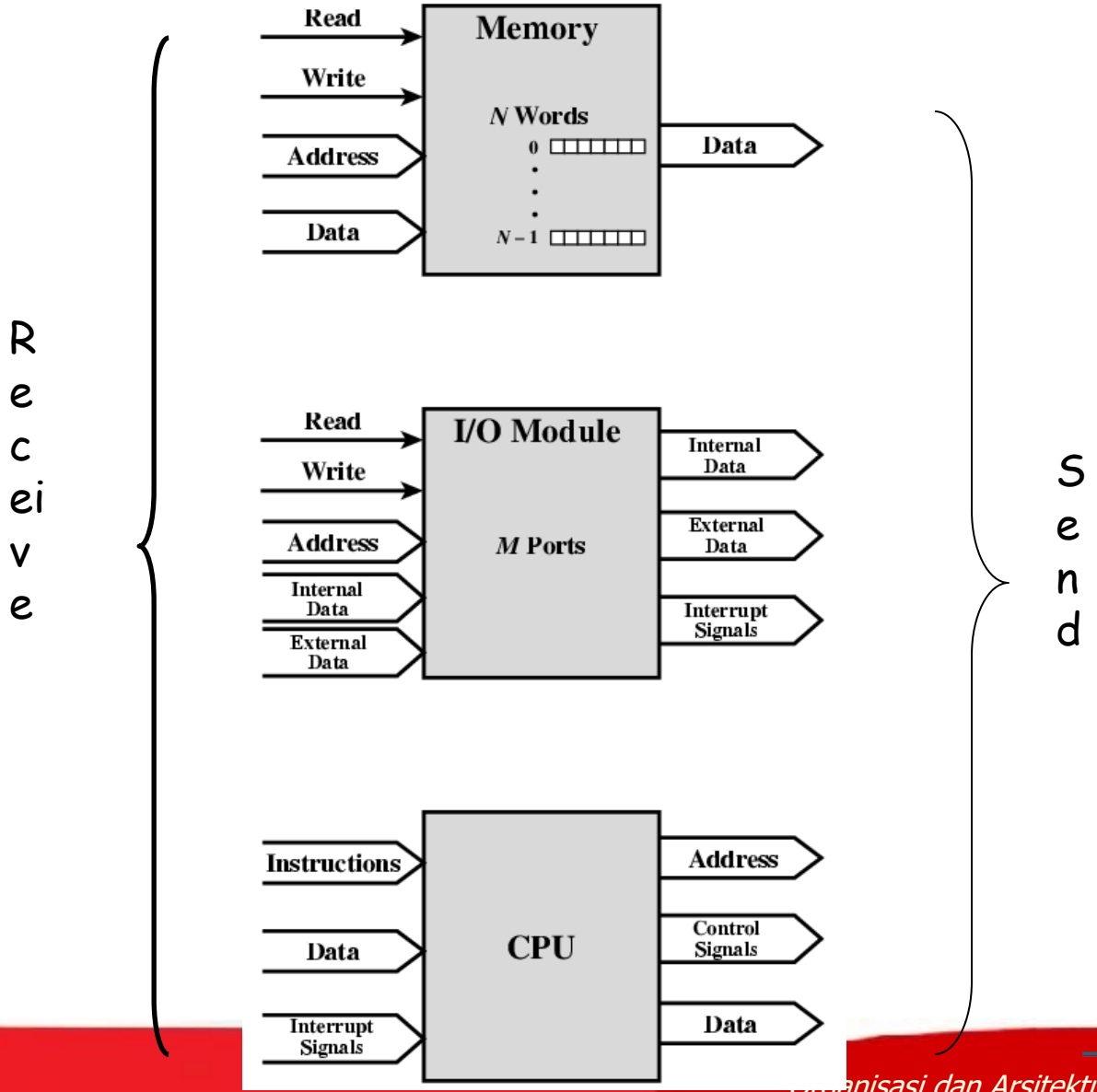
# Interkoneksi *antar bagian komputer*



- Mengapa antar bagian komputer perlu saling terkoneksi?
  - Agar data dapat diproses
  - Agar ...
- Interkoneksi yang mungkin terjadi antar bagian utama komputer:
  - Memori ⇨ CPU: proses **baca** instruksi atau data
  - Memori ⇦ CPU: proses **tulis** data
  - CPU ⇨ I/O: proses **kirim** data ke I/O
  - CPU ⇦ I/O: proses **baca** data dari I/O device
  - I/O ⇨ Memori: transfer data dari I/O ke memori (DMA)
  - I/O ⇦ Memori: transfer data dari memori ke I/O (DMA)

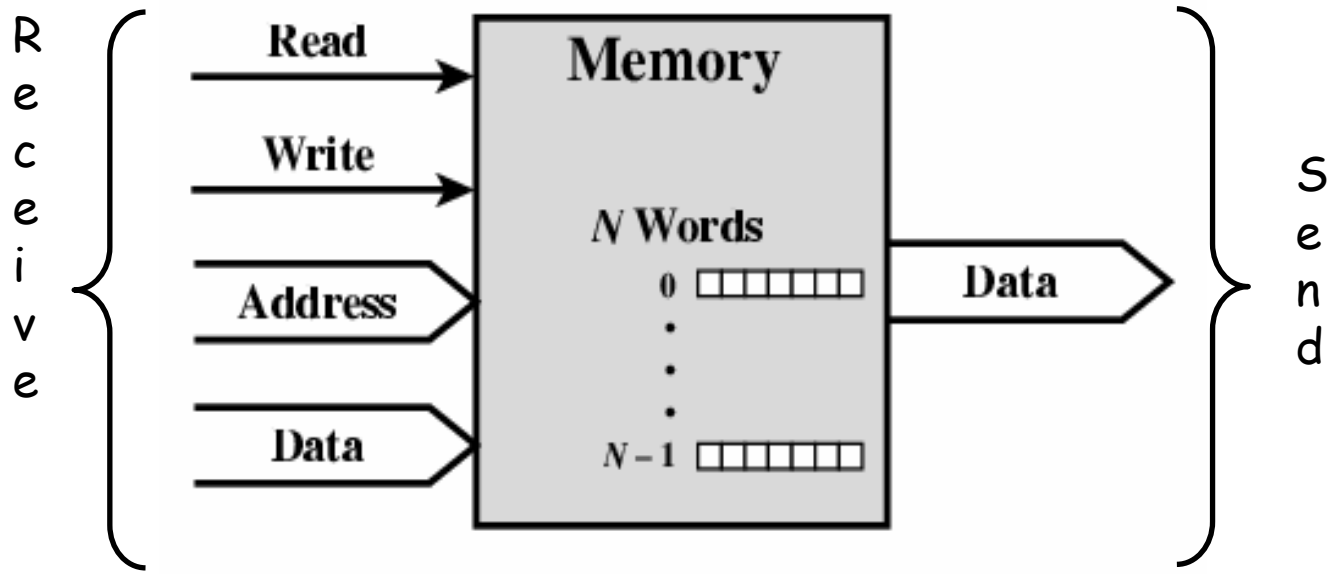
# Model koneksi tiap bagian komputer



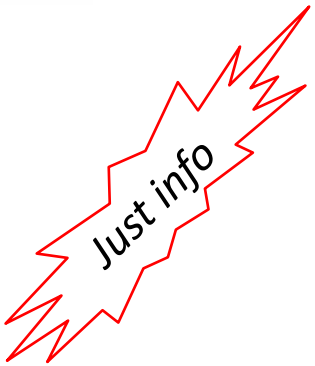


# Koneksi pada Memori



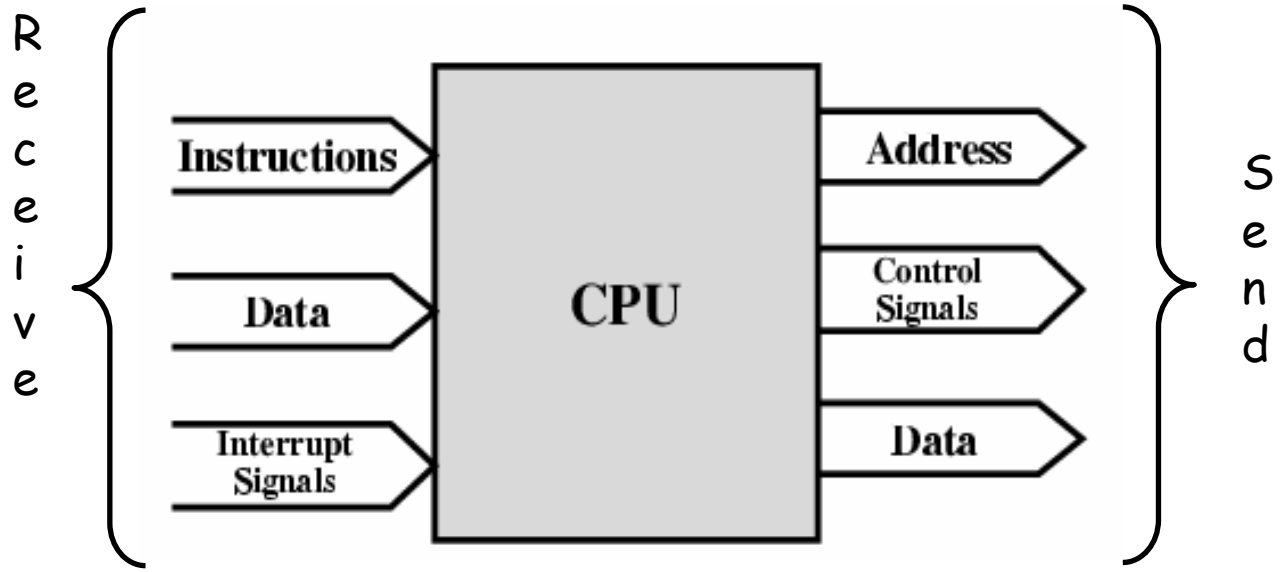



- Terima dan kirim data
- Terima alamat (lokasi memori)
- Terima signal kontrol:
  - *Read*
  - *Write*

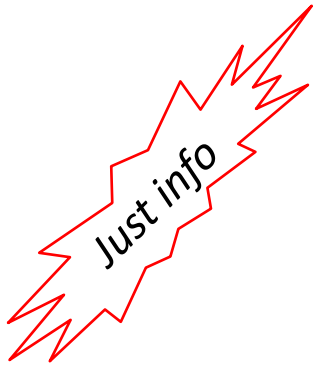


# Koneksi pada CPU



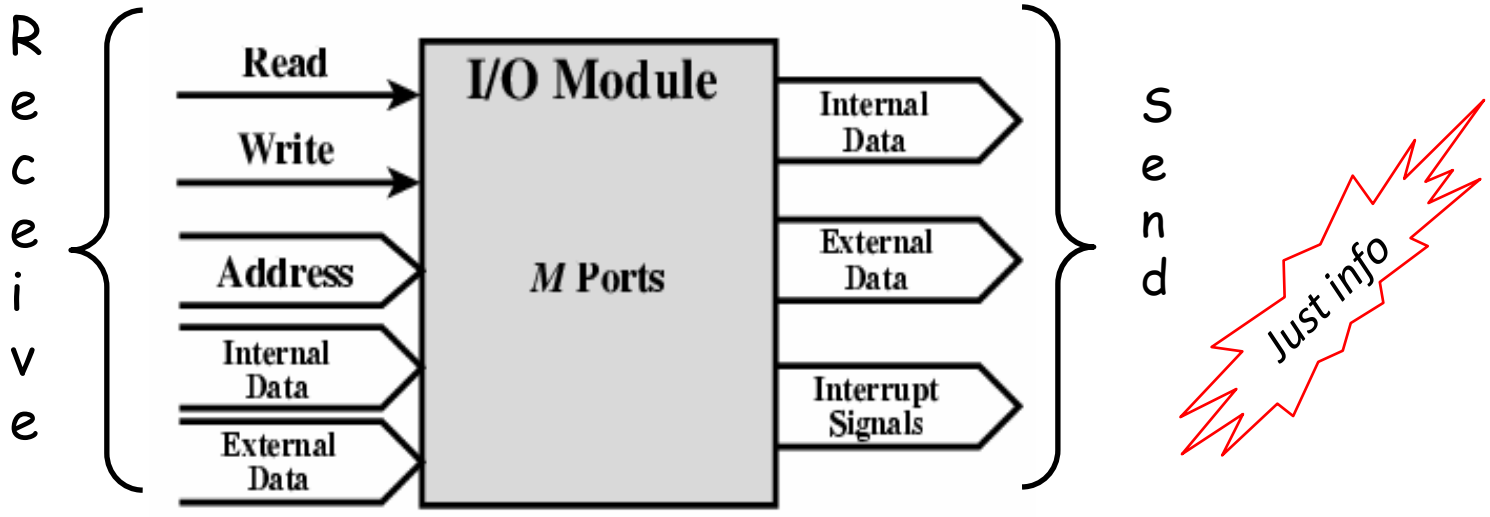



- Baca instruksi dan data
- Tulis data ke luar (sesudah diproses)
- Kirim alamat ke luar
- Kirim signal kontrol ke unit lain
- Terima *interrupt* dan lakukan aksi



# Koneksi pada *Input/Output* (1)



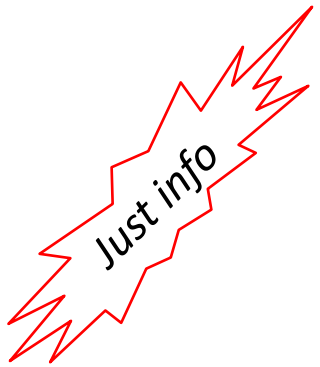



- Terima *signal* kontrol dari komputer (*read* atau *write*)
- Terima alamat dari komputer
  - ✓Misal: nomor port untuk identifikasi *peripheral*
- Terima data internal (dari komputer) dan data eksternal (dari *device* lain)
- Kirim data internal dan eksternal
- Kirim signal *interrupt* (*control*)

# Koneksi pada *Input/Output* (2)



- Dari sisi komputer I/O diperlakukan seperti memori
- Saat I/O berfungsi sebagai *Output*:
  - Terima data dari komputer
  - Kirim data ke *peripheral*
- Saat I/O berfungsi sebagai *Input*:
  - Terima data dari *peripheral*
  - Kirim data ke komputer



# Koneksi?...→→→ Bus

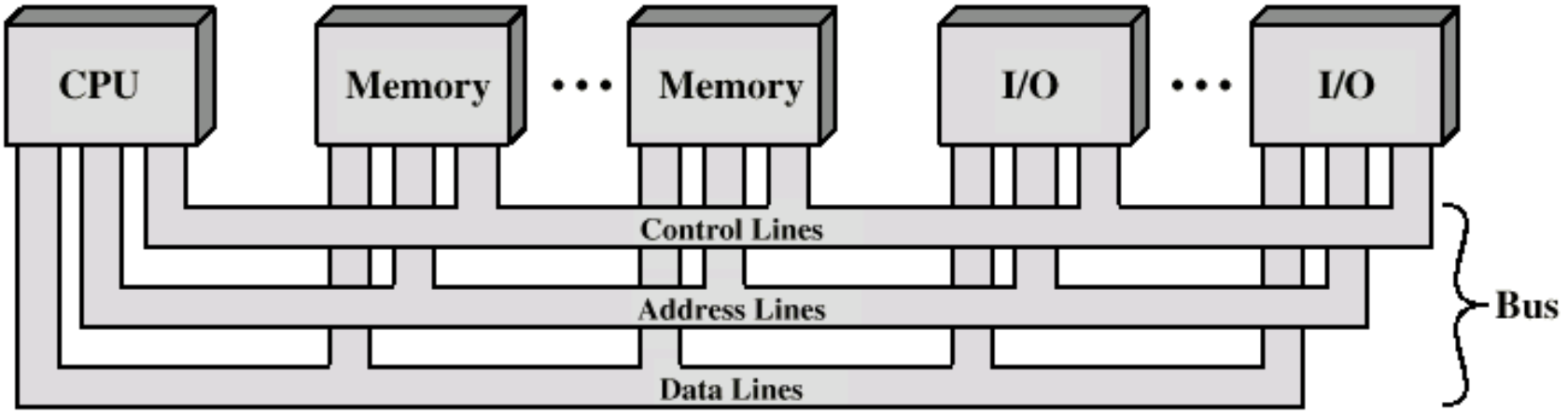


- Apakah BUS itu?
  - Saluran komunikasi yang menghubungkan dua *device* atau lebih
  - Biasanya bersifat *broadcast* → data menyebar ke seluruh *device* yang terhubung ke bus
  - Dalam satu waktu hanya satu *device* yang dapat mengirimkan data, tetapi *device* yang membaca data boleh lebih dari satu
  - Bus sering dikelompokkan:
    - Beberapa *channel* (jalur) digabung ke dalam satu bus
    - Misal: bus data 32 identik dengan 32 jalur terpisah masing masing satu bit
  - Bentuk fisik:
    - Jalur paralel (50 hingga ratusan) pada PCB (*printed circuit board*)
    - Pita kabel (seperti kabel untuk harddisk)
    - Konektor strip pada *mother board*
      - misal: ISA, PCI
    - Sekumpulan kabel

# Part 2: Apa saja jenis bus yang ada di dalam komputer?

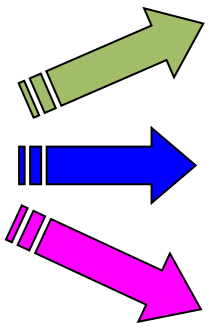
# Skema Interkoneksi Bus






**System bus:** jalur yang menghubungkan komponen utama komputer (CPU, memori, dan I/O)

**Struktur bus**



**Bus data**

**Bus alamat**

**Bus kontrol**

# Bus Data



- Fungsi: membawa data antar bagian utama komputer
  - Data di sini dapat berupa data atau instruksi
- Jumlah jalur data yang digunakan disebut lebar bus
- Lebar bus data menentukan performansi sistem
  - Makin lebar bus data →→ performansi sistem meningkat
  - Contoh: panjang data = 16 bit
    - Lebar bus data 8 bit →→ harus diambil **2x**
    - Lebar bus data 16 bit →→ cukup diambil **1x**
  - Macam lebar bus data: 8, 16, 32, 64 bit, dll

# Bus Alamat



- Mengidentifikasi asal atau tujuan data
  - Misal: CPU dapat membaca data yang ada di memori jika alamat data tersebut telah ditentukan
- Lebar bus menentukan ukuran maksimum memori yang dapat digunakan
  - Misal: Pentium I mempunyai lebar bus alamat 32 bit sehingga memori maksimumnya adalah $2^{32}$ = 4 gigabytes
  - Intel Pentium 4 dapat mempunyai memori maksimum = ...???
- Lebar bus alamat digunakan pula untuk pengalamatan port I/O

# Bus Kontrol

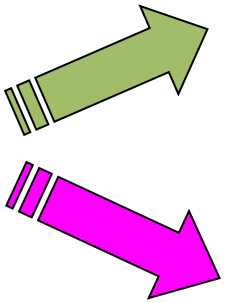


- Untuk mengatur pengaksesan dan penggunaan jalur data dan alamat
- Memberikan *timing* (untuk eksekusi program)
- Memberikan *signal* kontrol sbb:
  - *Memory read* (data di memori → bus data)
  - *Memory write* (data di bus data → memori)
  - *I/O read* (data di port I/O → bus data)
  - *I/O write* (data di bus data → *port I/O*)
  - *Transfer ACK* (data telah diterima/ditaruh dari/ke **bus**)
  - *Bus request* (permintaan untuk menggunakan bus)
  - *Bus grant* (status bus boleh digunakan)
  - *Interrupt request* (permintaan *interrupt*)
  - *Interrupt ACK* (*interrupt* telah diterima)
  - *Clock signals* (mensinkronkan operasi)
  - *Reset* (inisialisasi semua modul)

# Hirarki Bus



**Hirarki bus**



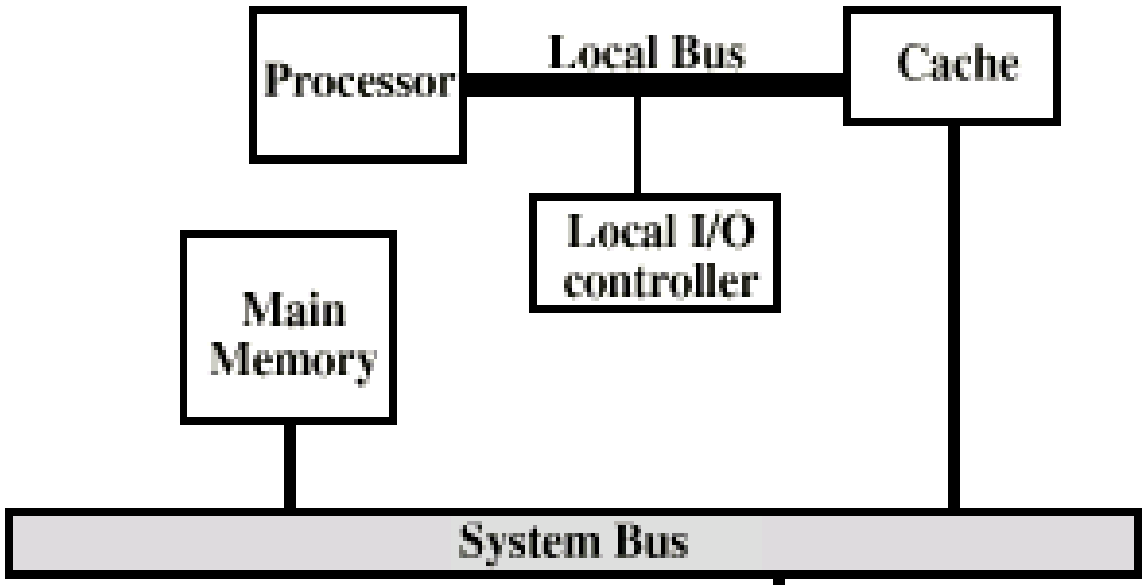
***Single bus***

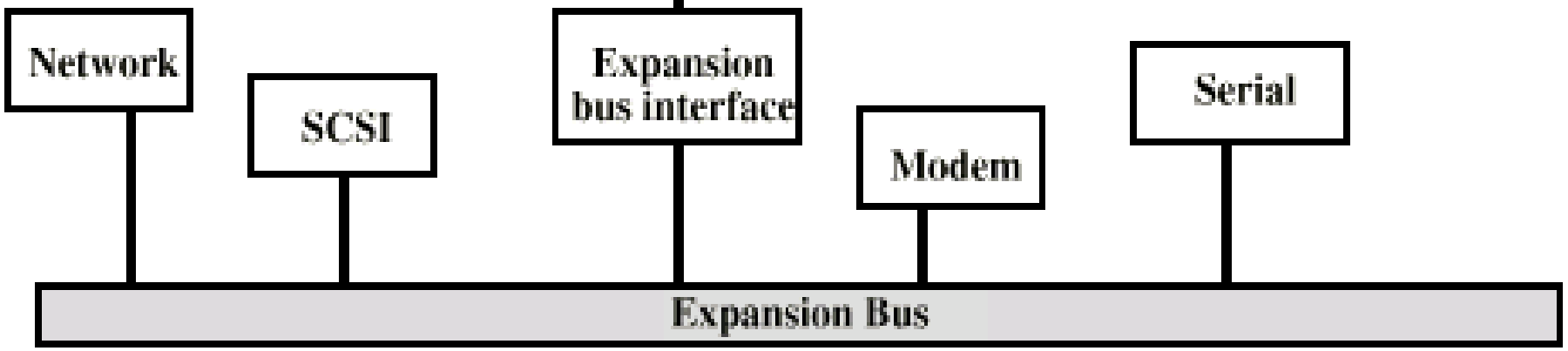
***Multiple bus***

- Apa kekurangan *single bus*?
  - Terjadi *propagation delay*
    - Bila *device* yang terhubung ke bus semakin banyak, maka saluran/bus yang digunakan menjadi semakin panjang
  - Terjadi *bottleneck*
    - Bila lalu lintas data melebihi kapasitas bus
- Bagaimana solusinya?
  - Gunakan MULTIPLE bus

# Multiple bus: *Traditional*





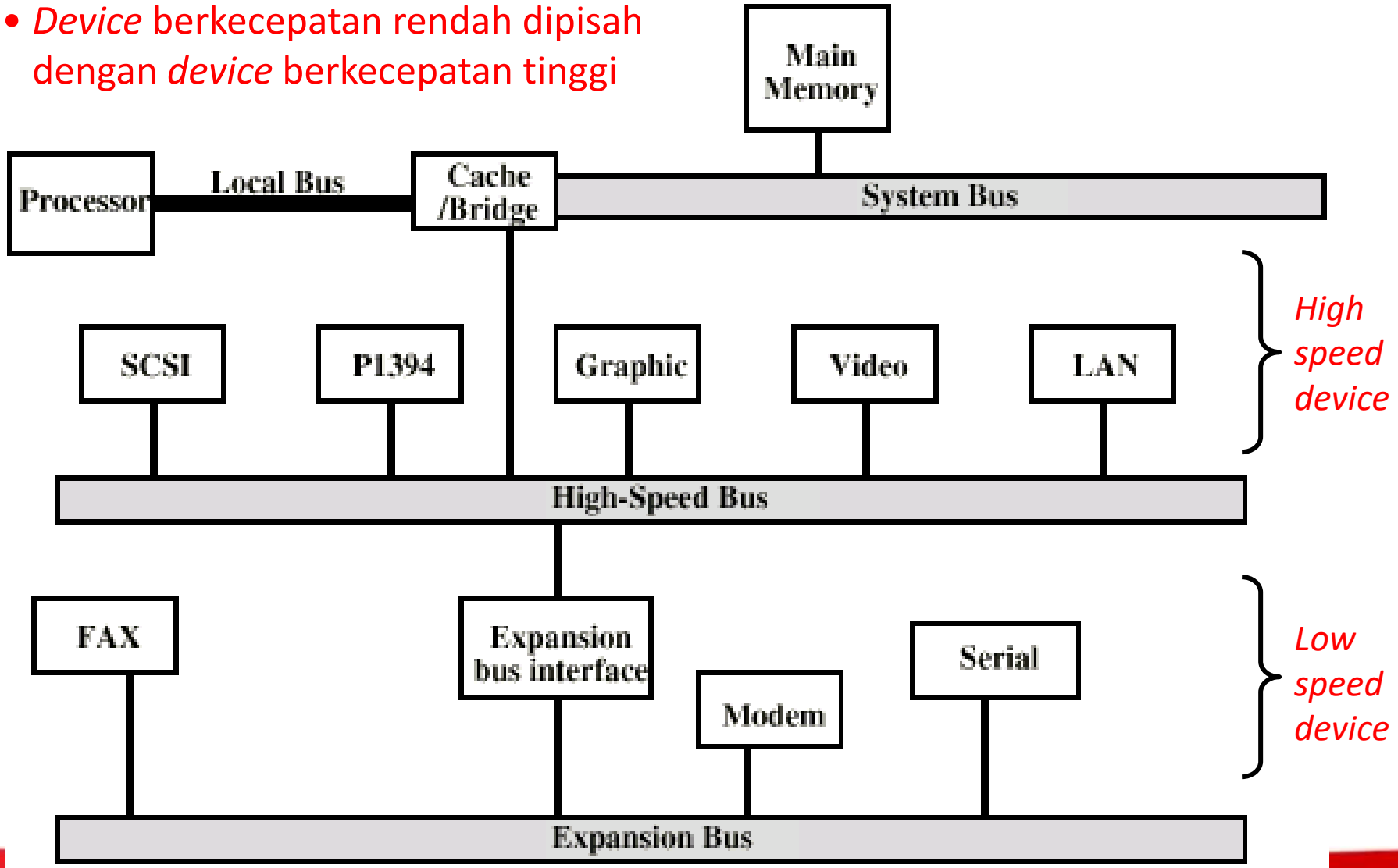




- Akses ke Main Memory tidak selalu melibatkan prosesor
- Bisa timbul masalah jika perbedaan kecepatan antara *device* cukup besar

# Multiple bus: High Performance



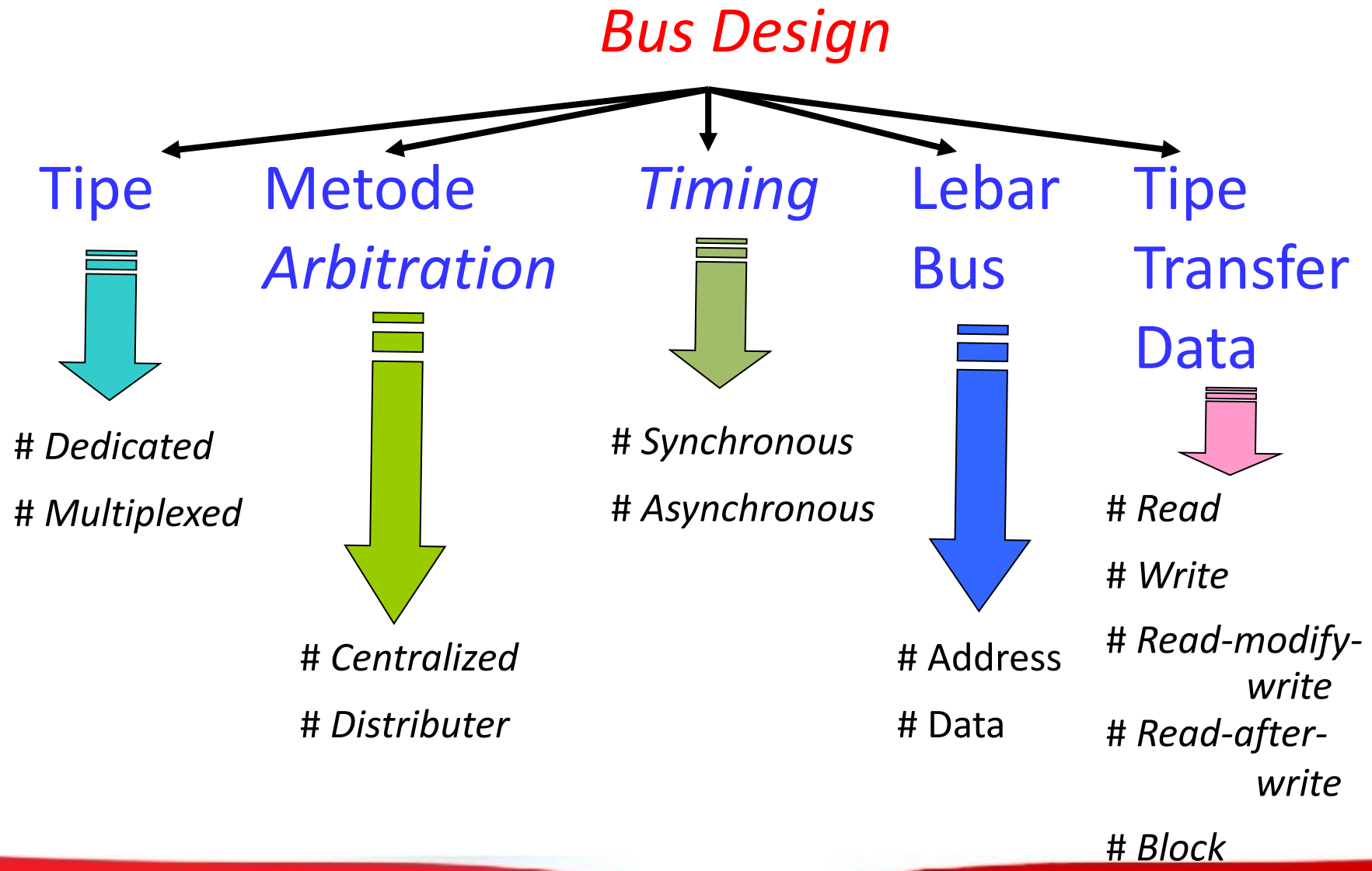
- *Device* berkecepatan rendah dipisah dengan *device* berkecepatan tinggi



# Part 3: Apa saja jenis-jenis bus berdasarkan elemen-elemennya?

# Elemen *Bus Design*





# Tipe Bus (1)



- *Dedicated*
  - Jalur data dan jalur alamat terpisah dan penggunaannya tetap/tidak berubah-ubah
  - Tipe bus yang banyak digunakan
- *Multiplexed*
  - Jalur bus digunakan untuk mengirimkan alamat dan data secara bergantian
  - Digunakan *control line*: ***address valid*** atau ***data valid***
  - Kelebihan: jumlah jalur lebih sedikit → hemat tempat → hemat biaya
  - Kerugian:
    - Penanganan lebih kompleks/rumit
    - Performansi berkurang: alamat dan data harus bergantian (tidak dapat paralel)

# Tipe Bus (2)



- *Physical dedication*
  - Termasuk tipe bus *multiplexed*
  - Jalur tertentu diperuntukkan bagi beberapa modul tertentu
  - Misal: bus I/O hanya untuk menghubungkan semua modul I/O → modul I/O tidak terhubung langsung ke bus sistem
  - Kelebihan: *throughput* meningkat, karena *bus contention* berkurang
  - Kekurangan: ukuran bertambah → biaya

# *Bus Arbitration* (1)



- *Bus arbitration* ≈ pengaturan bus
- Mengapa penggunaan bus perlu diatur?
  - Karena dalam satu saat:
    - Hanya boleh ada satu modul yang menggunakan bus
    - Dimungkinkan lebih dari satu modul ingin menggunakan bus
      - Misal: CPU dan DMA controller
- Jenis *bus arbitration*:
  - *Centralized* (terpusat)
  - *Distributed* (tersebar)

# *Bus Arbitration* (2)

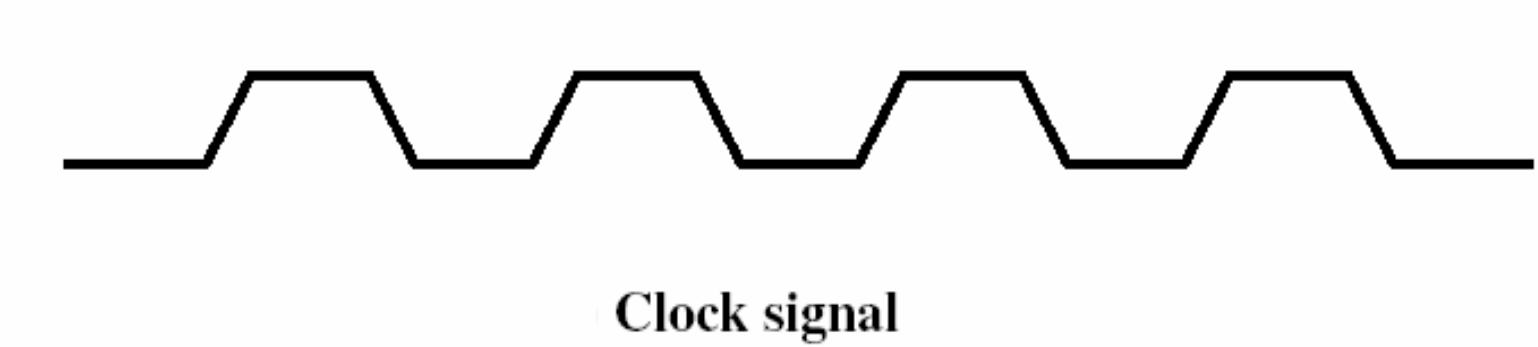


- *Centralized Arbitration*
  - Ditunjuk sebuah modul/*hardware* yang bertugas mengatur penggunaan bus dan disebut Bus *Controller* atau *Arbiter*
  - Realisasinya:
    - Dapat berupa modul terpisah, atau
    - Bagian dari CPU
- *Distributed Arbitration*
  - Tidak ada *controller* tunggal (terpusat)
  - Setiap modul dapat mengakses bus berdasarkan *control logic* pada setiap modul

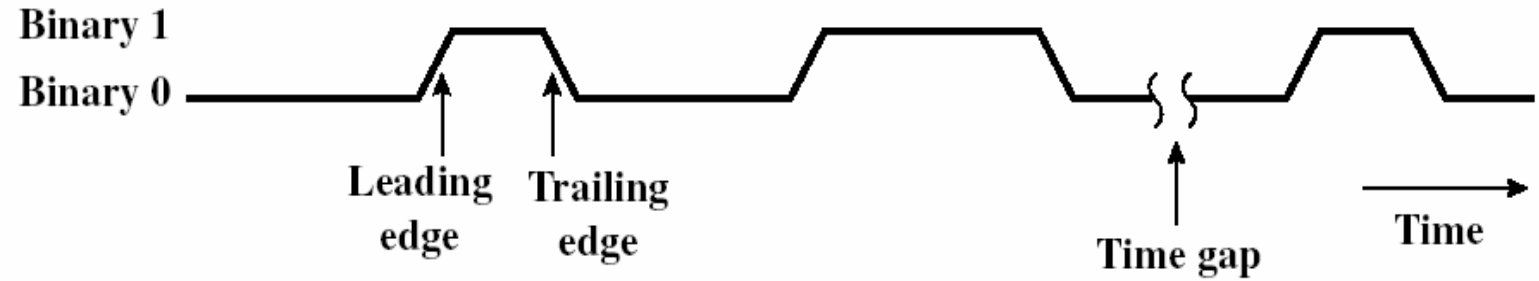
# Timing (1)



- Pengaturan *event* pada bus
- Berhubungan dengan *clock*



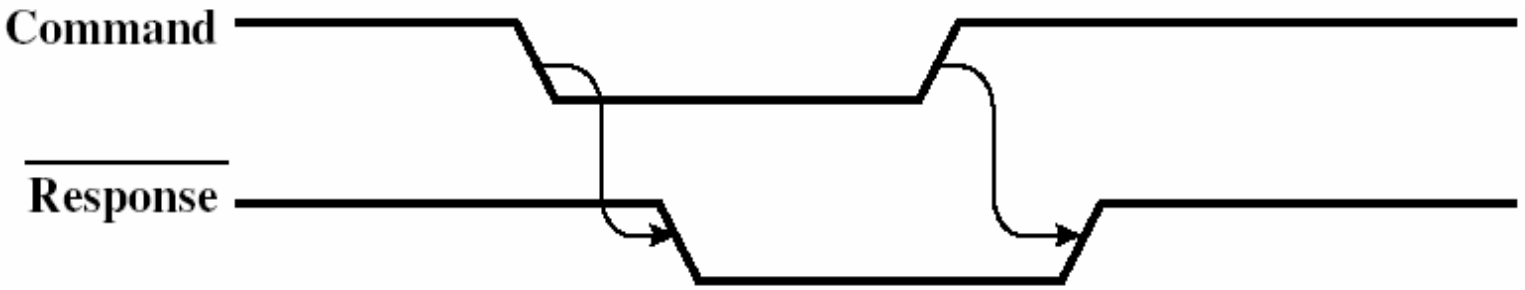
Clock signal



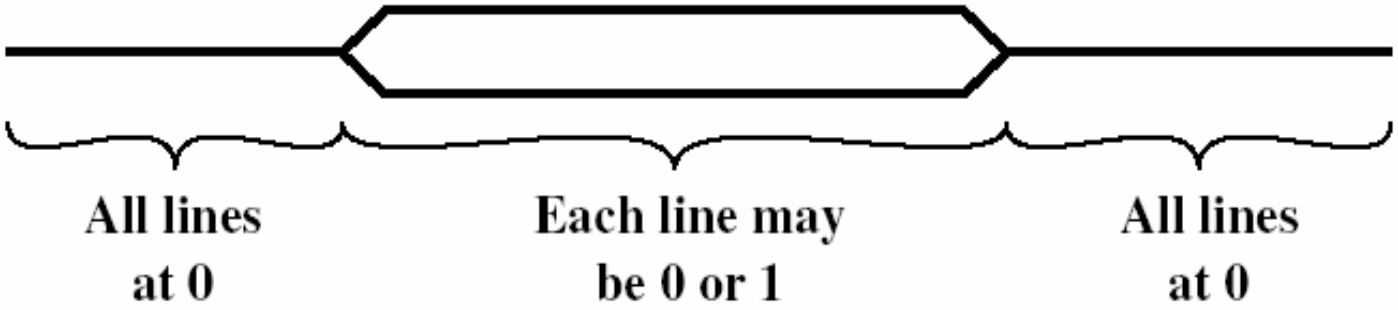

Signal as a function of time

# Timing (2)






Cause-and-effect dependencies




Groups of lines

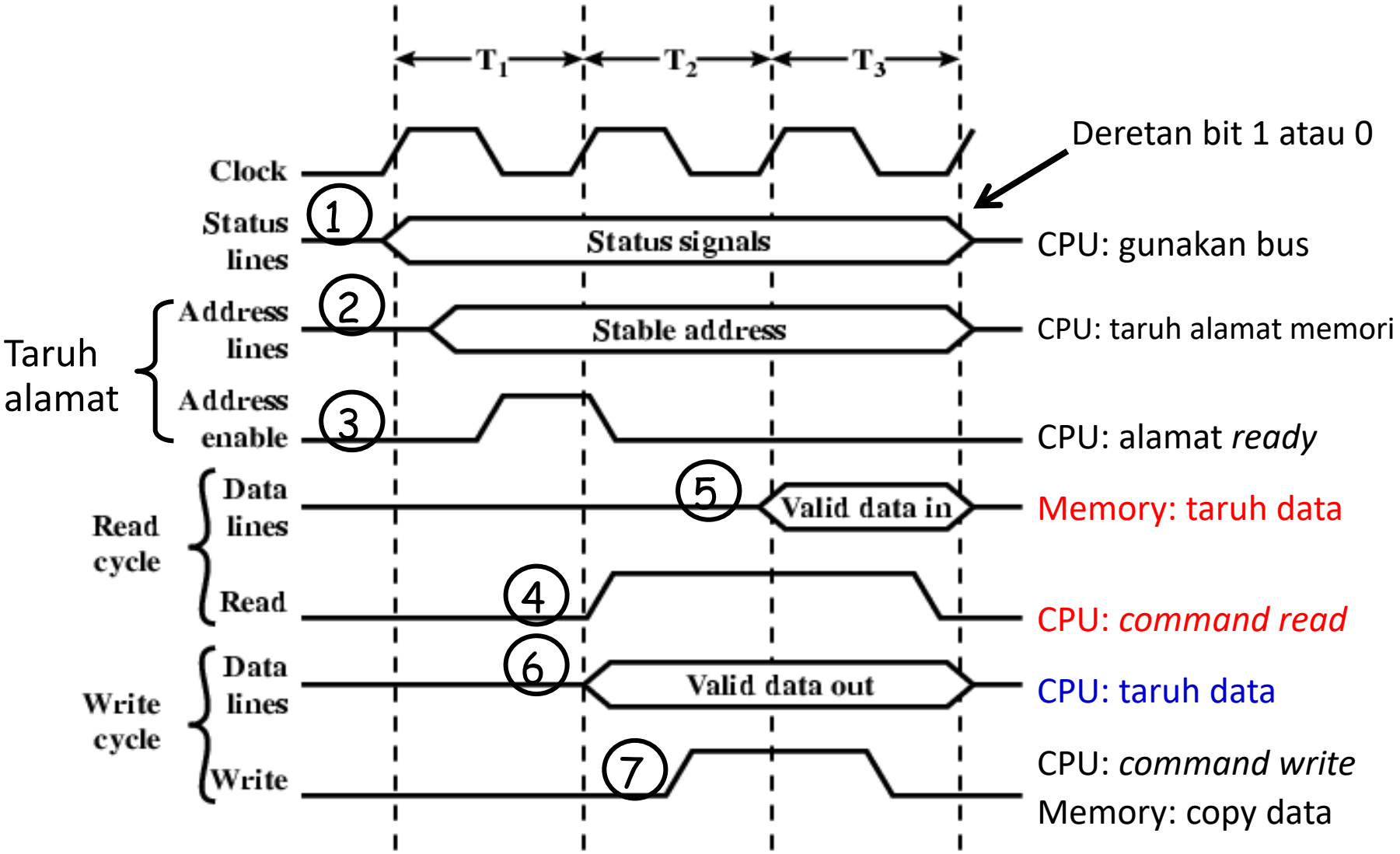
# *Synchronous* Timing



- Terjadinya *event* pada bus didasarkan pada clock
- Satu siklus bus terdiri dari sepasang bit 1-0
- Semua *device* dapat membaca *clock line*
- Sinkronisasi biasanya pada *ujung awal clock*
- Biasanya satu *event* untuk satu siklus

# Synchronous Timing Diagram (Read and Write)





# *Asynchronous Timing*



- *Event* berikutnya terjadi berdasarkan *event* sebelumnya (tidak berdasarkan *clock*)
- *Synchronous vs Asynchronous*:
  - *Synchronous*:
    - (+) Implementasi mudah
    - (+) Pengujian mudah
    - (-) Kurang fleksibel (*clock rate* tetap → *device* dengan *clock rate* lebih tinggi tidak meningkatkan performansi sistem)
  - *Asynchronous*:
    - (+) Lebih fleksibel → *device* model lama (kecepatan rendah) dapat digunakan bersama-sama dengan *device* berkecepatan lebih tinggi

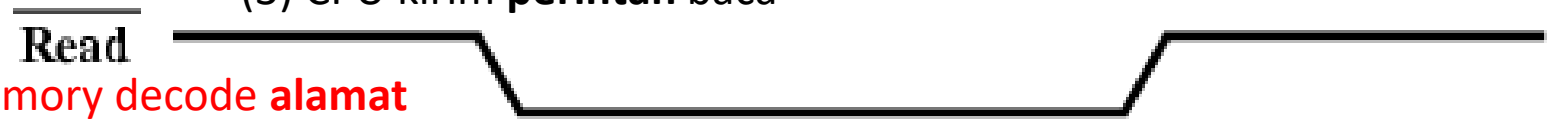
# Asynchronous Timing – Read Diagram

Fakultas Informatika
School of Computing
Telkom University

(1) CPU taruh signal **status** di jalur


(2) CPU taruh **Alamat** di bus


(3) CPU kirim **perintah** baca




--- (4) Memory decode **alamat**

(5) Memory taruh **data** di bus



(6) Memory kirim **signal ACK** data ready

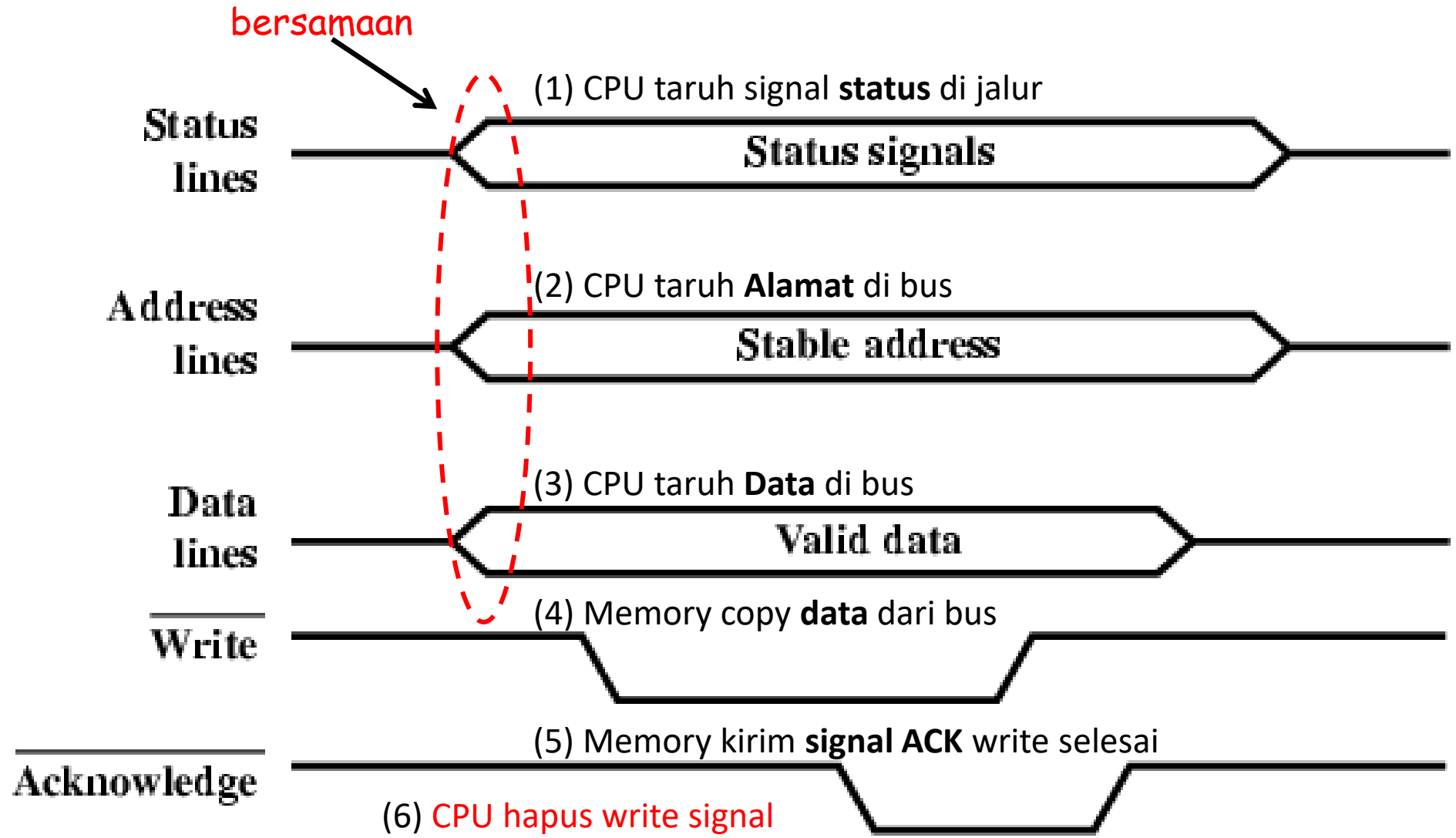
(7) CPU hapus read signal

(8) CPU hapus data+ACK

(9) CPU hapus alamat

# Asynchronous Timing – Write Diagram





# Lebar Bus (*bus width*)

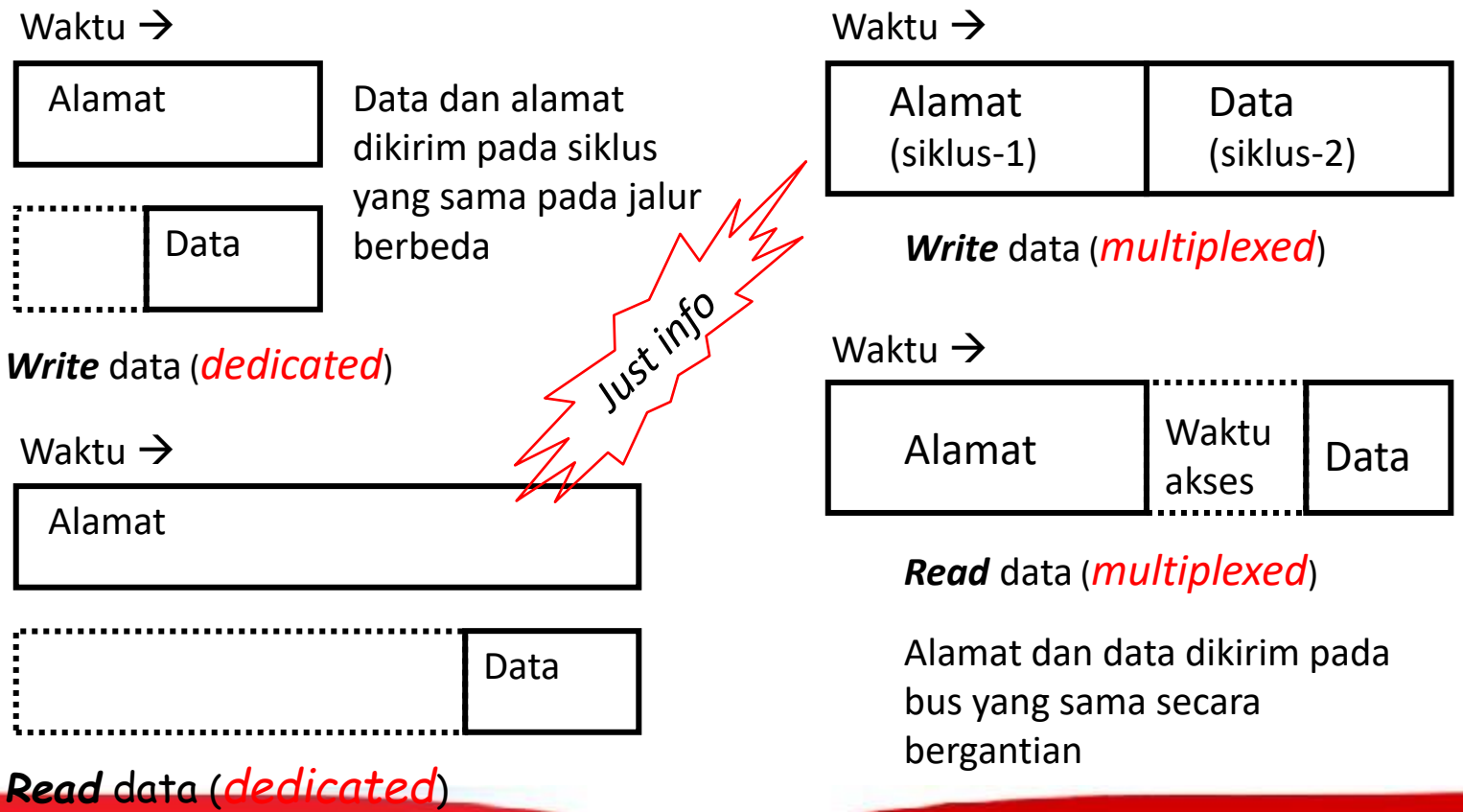


- Adalah banyaknya jalur yang digunakan
- Lebar bus berpengaruh terhadap:
  - Performansi
    - Makin lebar bus ***data*** yang digunakan → makin banyak jumlah bit data yang dapat dilewatkan dalam satu saat
      - Misal: Jika clock CPU sama, maka komputer dengan bus data 32 bit lebih cepat daripada komputer dengan bus data 16 bit
  - Kapasitas
    - Makin lebar bus ***alamat*** yang digunakan → makin besar memori yang dapat digunakan

# Model Transfer Data (1)



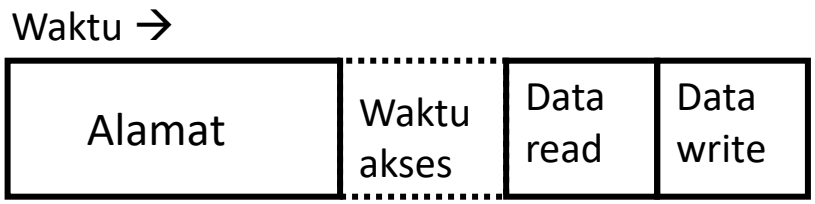
- Model transfer data antara jenis bus *dedicated* berbeda dengan *multiplexed*



# Model Transfer Data (2)



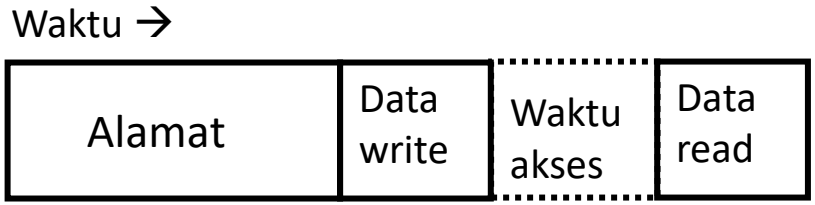
- Model transfer data yang lain:
  - Read-*modify*-write




Just info

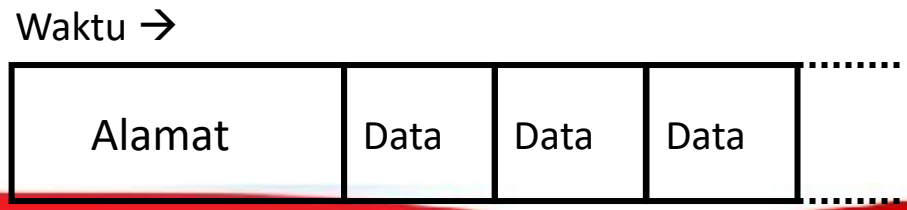
- *Data write* langsung dilakukan setelah *data read* selesai dilakukan (alamat sama)
- *Share memory* dapat terjaga (*indivisible*)

  - Read-*after*-write




- Data read langsung dilakukan setelah data write selesai dilakukan (alamat sama)
- *Share memory* dapat terjaga (*indivisible*)

  - Block data transfer



# Part 4: Bagaimana karakteristik bus PCI?

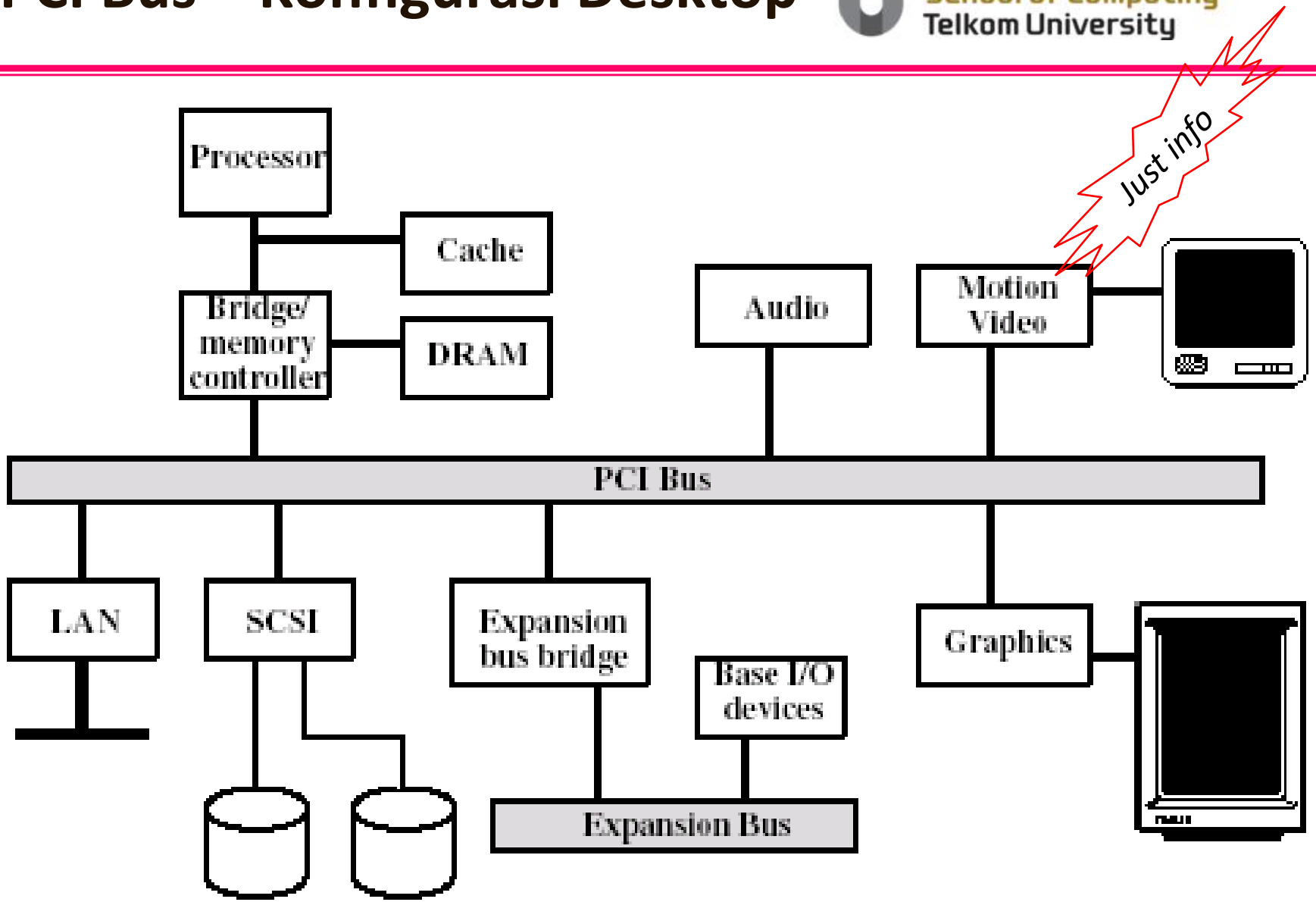
# PCI Bus



- PCI = *Peripheral Component Interconnection*
- Dirilis oleh Intel tahun 1990 pada Pentium-based
- Kelebihan:
  - Bus dengan *bandwidth* tinggi
  - Bus dengan *processor-independent*
  - Meningkatkan performansi sistem
  - Dapat memenuhi kebutuhan I/O secara ekonomis
  - Tetap kompatibel meskipun vendor berbeda
- Terdiri dari 32 atau 64 bit
- Transfer rate dengan 64 bit dan 66 MHz adalah 528 Mbytes/s (66x64/8) atau 4.224 Gbps
- Terdiri dari 49 jalur wajib dan 51 jalur tambahan
- Menggunakan model *multiplexed*, *synchronous timing* dan *centralized arbitration*
- Sinkronisasi terjadi saat perubahan *clock* dari high ke low (pertengahan siklus)

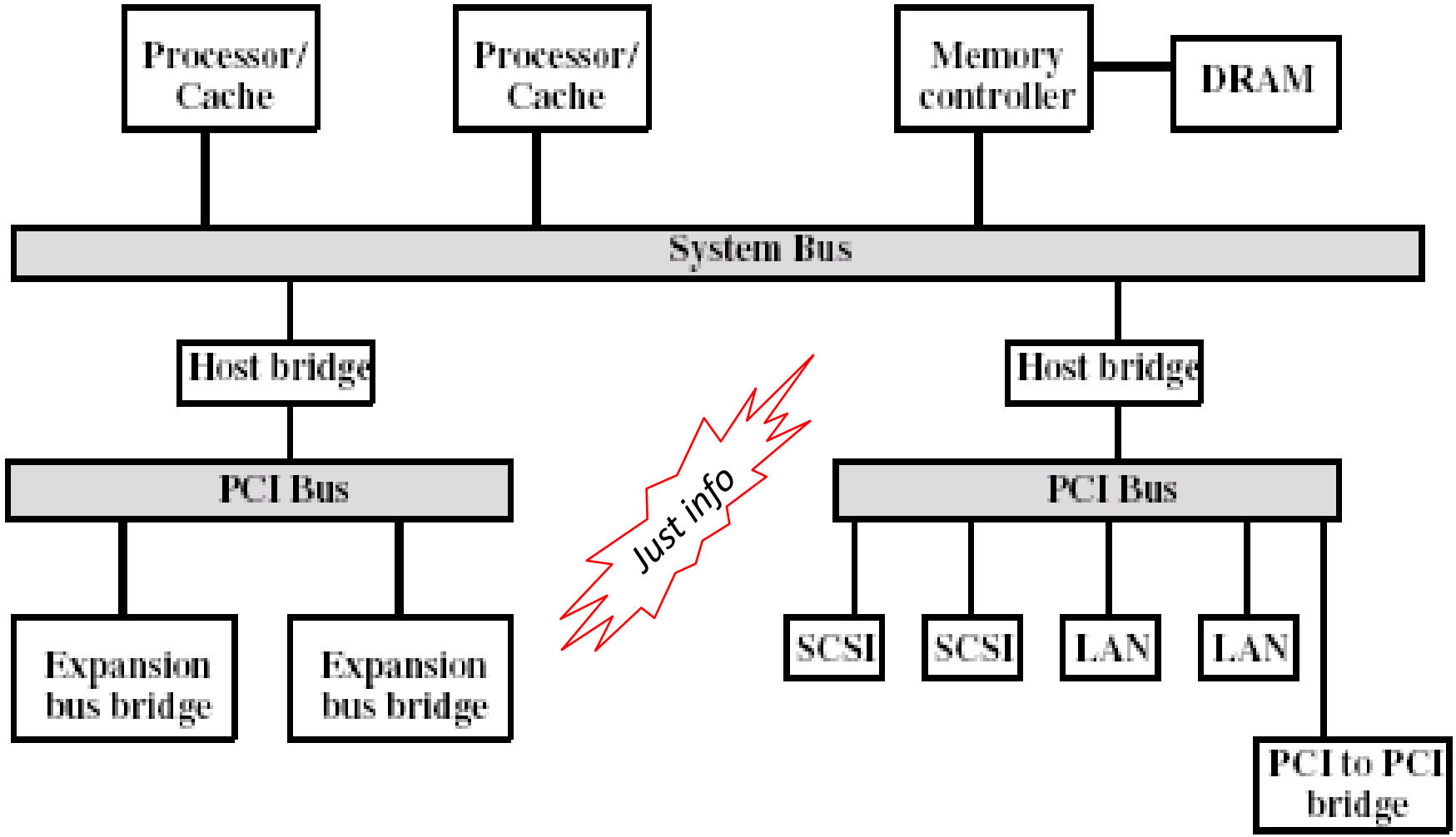
# PCI Bus – Konfigurasi Desktop





# PCI Bus – Konfigurasi Server





# PCI Bus Lines (wajib)



- Pin System
  - Terdiri dari clock dan reset
- Pin Alamat & Data
  - 32 jalur multiplekser untuk alamat/data
  - Jalur untuk validasi data
  - Jalur untuk mengartikan signal
- Pin Interface Control
  - Mengatur *timing* transaksi
  - Melakukan koordinasi antara *initiator* dan target
- Pin *Arbitration*
  - *Not shared*
  - Setiap PCI modul mempunyai sambungan langsung ke *PCI bus arbiter*
- Pin *Error reporting*
  - Untuk melaporkan kesalahan paritas atau kesalahan lain

# PCI Bus Lines (Optional)



- Pin *Interrupt*
  - Digunakan untuk mengirimkan interrupt
  - *Not shared*, setiap modul mempunyai saluran *interrupt* tersendiri
- Pin *Cache support*
  - Untuk mendukung memori PCI yang dapat di-*cache* ke prosesor atau *device* lain
- 64-bit Bus Extension
  - Jalur tambahan 32 jalur
  - *Time multiplexed*
  - Jalur untuk validasi data
  - Jalur untuk mengartikan *signal*
  - 2 jalur untuk meng-*enable device* yang dapat menggunakan transfer 64-bit
- *JTAG/Boundary Scan*
  - Untuk melakukan prosedur *testing*

# PCI *Commands*

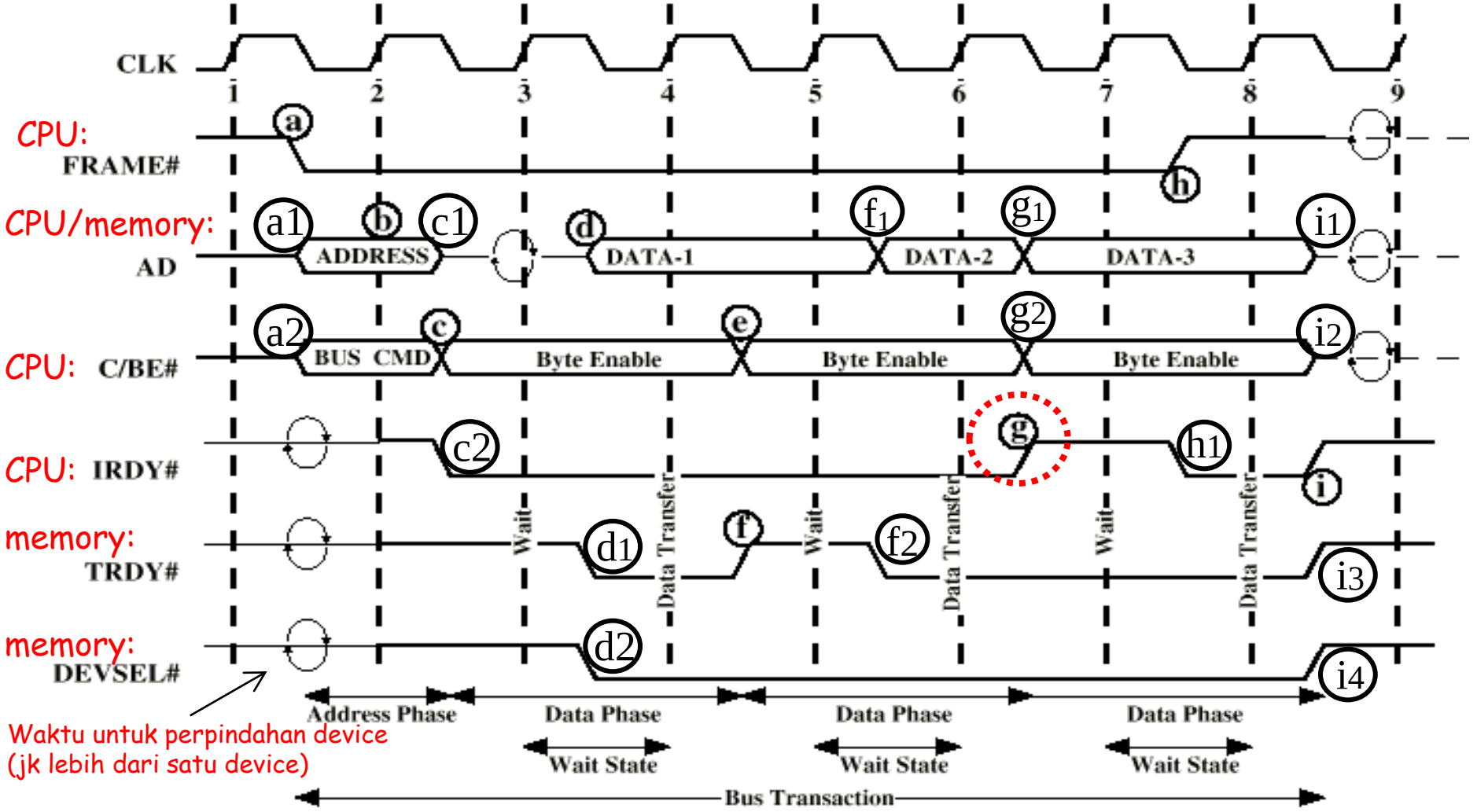


- Untuk mengatur transaksi antara *initiator* (*master*) and target
- Menentukan jenis transaksi yang akan dilakukan
  - Misal: *I/O read/write*
- Memberikan tanda (*signal*) suatu *event* telah selesai dilakukan
- Memberikan tanda suatu kondisi (siap menerima data atau data telah ditaruh di bus) kepada modul lain yang membutuhkan

# Part 5: Bagaimana cara kerja bus PCI ketika terjadi proses baca data?

# PCI *Read* Timing Diagram (1)

Fakultas Informatika
School of Computing
Telkom University



# PCI *Read* Timing Diagram (2)

Fakultas Informatika
School of Computing
Telkom University

Keterangan:

CPU: (a) Aktifkan FRAME, (a1) taruh alamat yang akan dibaca,
(a2) Taruh bus CMD

Memory: (b) mulai mendeteksi alamat di bus

CPU: (c1) Telah selesai menaruh alamat di jalur AD (*Address Data*),
(c) mengubah status jalur AD (C = command → BE = byte enable),
(c2) kirim signal siap baca data (IRDY)

Memory: (d) Taruh data-1 di bus, (d1) kirim signal data valid di bus (TRDY),
(d2) Kirim signal alamat selesai di-*decode* (DEVSEL)

CPU: (e) Tentukan status jalur = BE (untuk data berikutnya)

Memory: (f) Kirim signal data tidak valid → Persiapan untuk kirim data berikutnya

Memory: (f1) Taruh data ke-2, (f2) kirim signal data valid di *bus* (TRDY)

CPU: (g2) Tentukan status jalur = BE untuk data berikutnya,
(g) Tidak siap terima data (buffer full)

Memory: (g1) Pertahankan data ke-3 yang telah ditaruh di *bus*

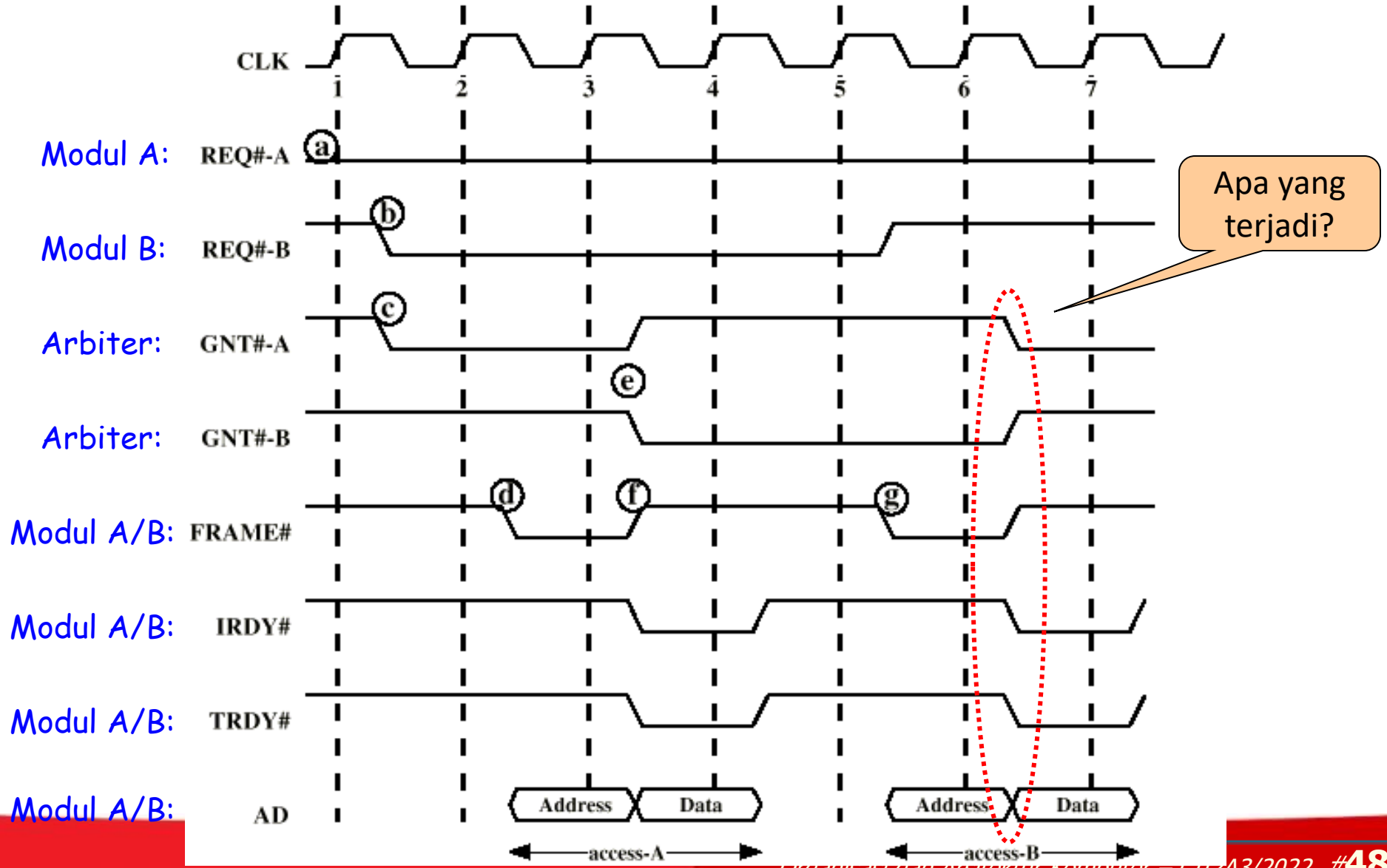
CPU: (h1) kirim signal siap baca data (IRDY),
(h) baca data selesai→disable FRAME

CPU dan memory: (i) – (i4) kembalikan jalur yang digunakan ke posisi normal

# Part 6: Bagaimana cara pengaturan penggunaan bus PCI secara bergantian?

# Contoh PCI Bus *Arbitration* (2 modul)





# Contoh PCI Bus Arbitration
## (2 modul)

Fakultas Informatika
School of Computing
Telkom University

Keterangan:

Arbiter: (a) men-scan jalur dan mendapati modul A telah aktif (*request bus*)
Modul B: (b) request bus di pertengahan siklus satu
Arbiter: (c) mengijinkan modul A menggunakan bus (GNT-A)
Modul A: (d1+d2) membaca status jalur IRDY dan TRDY dan ternyata statusnya idle → (d) modul a mengaktifkan FRAME,
(d3) mengirim alamat, (tdk digambar) mengirim signal C/BE
Arbiter: (e1) menghentikan servis terhadap modul A,
(e) memberi kesempatan kepada modul B (GNT-B)
Modul A: (f1) beritahu target dengan mengirim *signal* IRDY,
(f2) kirim signal data *valid* (TRDY),
(f3) taruh data terakhir di bus,
(f) normalkan FRAME
Modul B: (g) Aktifkan FRAME,
(g2) taruh alamat,
(g1) normalkan REQ-B
...

# Referensi

Fakultas Informatika
School of Computing
Telkom University

[STA19] Stalling, William. 2019. "*Computer Organization and Architecture: Designing for Performance*". 11th edition. Prentice Hall.