LAPORAN UJIAN TENGAH SEMESTER (UTS)
Mata Kuliah: Organisasi Komputer
Semester: 4
Dosen Pengampu: : Gentur Wahyu N. W., M.Kom.

Identitas Mahasiswa

- **Nama**: Rasya Aditya Amelia Putra
- **NIM**: 231240001385
- **Tanggal Pengumpulan**: 8 Mei 2025

Daftar Isi

1.Studi Kasus Analitis (Bagian A)

Arsitektur yang dianalisis adalah **arsitektur server berbasis x86** yang umum digunakan di lingkungan **data center**. Arsitektur ini terdiri dari prosesor multi-core (x86_64), **RAM ECC** (Error-Correcting Code Memory), **storage RAID** (Redundant Array of Independent Disks), dan sistem **bus** yang menghubungkan semua komponen.

Komponen utama:

- **CPU (Central Processing Unit):** Umumnya terdiri dari beberapa core. Tiap core bertugas menjalankan instruksi dari client (misalnya permintaan HTTP).
- **RAM ECC:** Menyediakan memori utama dengan kemampuan mendeteksi dan memperbaiki kesalahan data. Sangat krusial untuk sistem yang membutuhkan keandalan tinggi.
- **RAID Storage:** Sistem penyimpanan yang terdiri dari beberapa disk, dikonfigurasi untuk toleransi kesalahan dan kecepatan (RAID 0, 1, 5, 10).
- **Bus Sistem (Front Side Bus, Memory Bus, I/O Bus):** Jalur komunikasi antar komponen.
- **Northbridge & Southbridge (pada beberapa sistem):** Chipset yang mengatur lalu lintas data antara CPU, RAM, dan I/O.

1. Komunikasi Antar Komponen

Komunikasi antar komponen terjadi melalui bus sistem. Proses komunikasi ini sangat krusial agar semua komponen dapat bekerja serempak dan efisien.
Contoh alur komunikasi:

- Client mengirim permintaan → CPU menerima instruksi → CPU meminta data dari RAM atau menyimpan ke RAID.
- CPU berkomunikasi dengan RAM melalui **memory controller** (sering terintegrasi dalam CPU modern).
- CPU mengakses RAID melalui **SATA/NVMe controller** via I/O Bus.

Jenis komunikasi:

- **Data Bus:** Mengangkut data antar komponen.
- **Address Bus:** Menunjukkan lokasi memori/data yang akan diakses.
- **Control Bus:** Mengontrol jenis operasi (read/write) dan sinkronisasi antar perangkat.

2. Diagram Blok Komponen

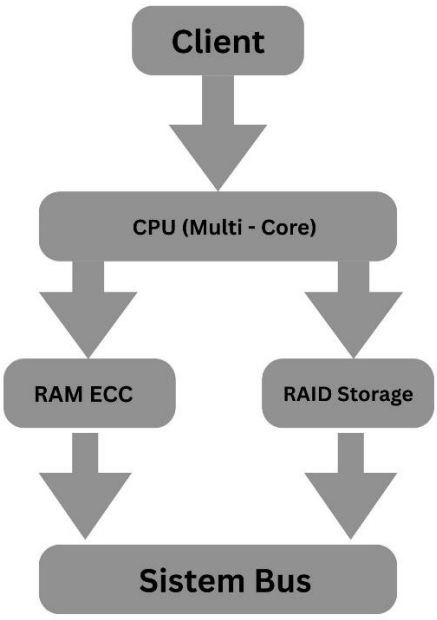

Diagram Blok Komponen

Keterangan:

- CPU mengendalikan komunikasi dan pemrosesan.
- RAM ECC menyimpan data sementara dengan keamanan ekstra.
- RAID menyimpan data secara permanen dan redundan.
- Semua komponen terhubung via System Bus yang membawa data, alamat, dan sinyal kontrol.

3. Siklus Kerja Fetch - Decode - Execute (sederhana)
Siklus kerja dasar CPU saat mengeksekusi instruksi:

1. Fetch (Ambil Instruksi):
   CPU mengambil instruksi dari memori utama (RAM).
   Instruksi ini ditentukan oleh Program Counter (PC).
2. Decode (Dekode Instruksi):
   Instruksi yang diambil diinterpretasikan oleh kontrol unit.
   CPU memahami instruksi apakah itu "tambah data", "akses memori", atau "tulis ke disk".
3. Execute (Eksekusi Instruksi):
   CPU menjalankan perintah, bisa berupa operasi aritmatika/logika (oleh ALU), memindahkan data, atau mengakses perangkat I/O seperti RAID.
4. Write Back:
   Hasil eksekusi disimpan kembali ke register, RAM, atau disk tergantung jenis operasi.

4. Bandwidth Bus dan Bottleneck Potensial

Bandwidth Bus adalah ukuran seberapa banyak data yang bisa ditransfer melalui bus dalam satuan waktu (misalnya MB/s atau GB/s).
Komponen bandwidth penting:

- Memory Bus Bandwidth: Menghubungkan CPU dan RAM.
- I/O Bus Bandwidth: Menghubungkan CPU dengan storage (RAID).
- Interconnect antar Core CPU: Bandwidth antar core juga penting untuk komunikasi internal.

Bottleneck Potensial:

- Jika I/O Bus lambat (misalnya masih SATA bukan NVMe), akses ke RAID akan menjadi lambat walaupun CPU dan RAM cepat.
- Jika bandwidth antar core CPU terbatas, maka proses multithread bisa menjadi tidak efisien.
- RAM ECC lebih lambat dibanding RAM biasa, jadi perlu dikompensasi dengan cache CPU yang optimal.

Solusi:

- Gunakan bus yang lebih lebar dan cepat (misal PCIe Gen4).
- Gunakan interkoneksi internal cepat antar CPU core (seperti AMD Infinity Fabric atau Intel Mesh).
- Pastikan pemilihan RAID level sesuai kebutuhan kecepatan dan redundansi.

2. Program Konversi Bilangan (Bagian B)

Dalam sistem arsitektur komputer, berbagai representasi bilangan digunakan untuk komunikasi dan pemrosesan data. Komputer secara internal hanya memahami data dalam bentuk biner (basis 2). Namun, untuk keperluan tertentu, manusia menggunakan sistem bilangan lain seperti oktal (basis 8) dan heksadesimal (basis 16) karena lebih ringkas untuk merepresentasikan data biner.

Contohnya:

- Biner cocok untuk pemrosesan di level logika gerbang (AND, OR, NOT).
- Oktal digunakan pada sistem-sistem tertentu seperti permission di UNIX (contoh: chmod 755).
- Heksadesimal digunakan secara luas dalam pemrograman, debugging, dan representasi alamat memori (misalnya: 0xFF untuk nilai 255).

Tujuan Program

Membuat sebuah program sederhana dalam bahasa Python yang mampu:

- Menerima input bilangan desimal dari pengguna.
- Mengonversi nilai desimal tersebut ke dalam:
  - Biner (binary)
  - Oktal (octal)
  - Heksadesimal (hexadecimal)

Fungsi Python yang Digunakan:

Python menyediakan built-in functions untuk konversi bilangan:

- bin() → konversi ke biner
- oct() → konversi ke oktal
- hex() → konversi ke heksadesimal

Contoh:

```
angka = 255
print(bin(angka))   # Output: '0b11111111'
print(oct(angka))   # Output: '0o377'
print(hex(angka))   # Output: '0xff'
```

Algoritma Umum Program:

1. Minta input bilangan desimal dari pengguna.
2. Validasi input agar berupa angka positif.
3. Gunakan fungsi bin(), oct(), dan hex() untuk konversi.
4. Tampilkan hasil konversi dengan format yang mudah dibaca.

Contoh:

```
Masukkan bilangan desimal: 100
Konversi ke Biner: 0b1100100
Konversi ke Oktal: 0o144
Konversi ke Heksadesimal: 0x64
```

Manfaat dalam Konteks Arsitektur Komputer:

- Memudahkan pemahaman representasi data yang digunakan komputer.
- Berguna untuk debugging sistem low-level (alamat memori, register).
- Membantu mahasiswa atau teknisi mengenali pola bit dalam operasi logika atau perhitungan biner.

3. Simulasi Komunikasi Data (Bagian C)

Kasus: **Client mengirim permintaan data ke server, lalu server menyimpan data tersebut ke disk RAID.**

Langkah:

1. Client mengirim permintaan HTTP POST ke server.
2. CPU menerima permintaan dan menyimpannya sementara di RAM ECC.
3. CPU melakukan penjadwalan dan perintah write ke RAID melalui I/O controller.
4. Data ditulis ke RAID (RAID 5 misalnya, dengan parity).
5. CPU memberi respons ke client bahwa penyimpanan berhasil.

1. Diagram Alur Komunikasi Data

Berikut deskripsi diagram alur (bisa dibuatkan visual diagram jika diminta):

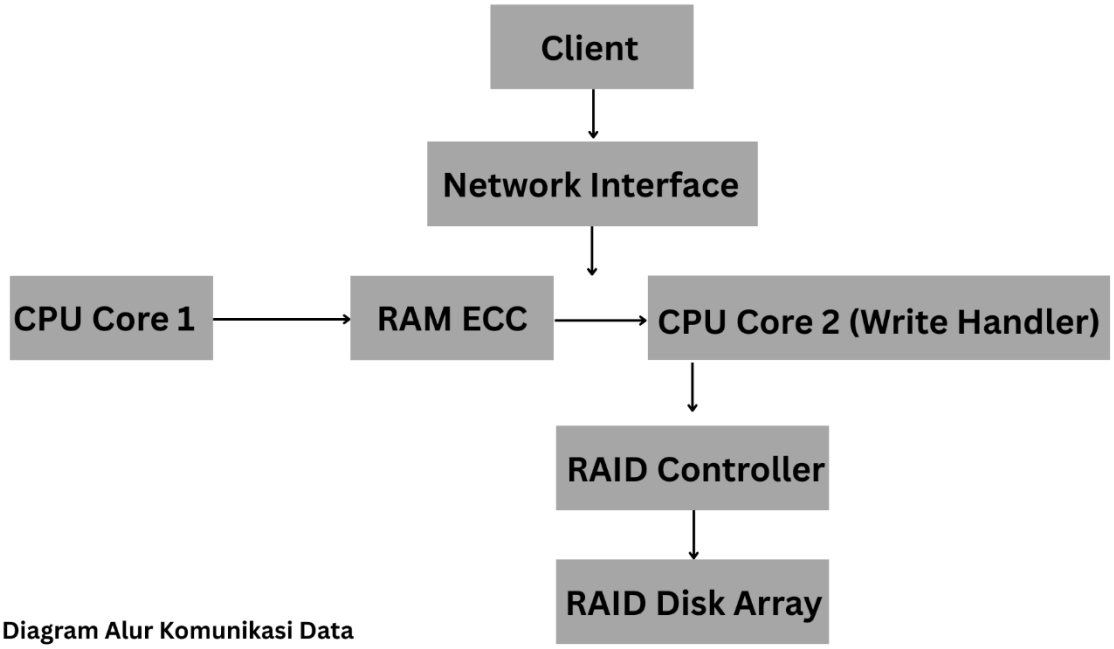

Diagram Alur Komunikasi Data

Keterangan Detail Alur Komunikasi Data:

1. **Client Request → Network Interface:**
    - Client mengirimkan permintaan (misalnya POST data JSON) melalui jaringan.
    - **Network Interface Card (NIC)** menerima data dan meneruskannya ke sistem operasi server.
2. **Interrupt Handler & CPU Processing:**
    - NIC menghasilkan **interrupt**, memberitahu CPU bahwa ada data masuk.
    - CPU mengaktifkan **interrupt handler** dan mulai memproses permintaan client.
    - Salah satu core CPU (misalnya Core 1) membaca data dari buffer jaringan dan menganalisis permintaan.
3. **Penyimpanan Sementara di RAM ECC:**
    - Setelah data diproses awal, CPU memuat data ke **RAM ECC** untuk sementara.
    - RAM ECC menyediakan **deteksi dan koreksi kesalahan**, menjamin data tidak korup selama berada di memori.
4. **Pengalihan Tugas ke Core CPU lain (Opsional - Multithreaded):**
    - Jika sistem menggunakan **multithreading** atau multitasking, proses penyimpanan ke disk bisa dijadwalkan ke core CPU lain (misalnya Core 2).
    - Scheduler dari OS akan menangani context switching antar core.
5. **Perintah Penulisan ke RAID Storage:**
    - CPU menginstruksikan sistem untuk menulis data ke storage.
    - Perintah dikirim melalui **I/O Bus** ke **RAID Controller** (bisa hardware RAID atau software RAID).
6. **RAID Controller Menulis ke Disk:**
    - RAID controller mengatur bagaimana data dibagi dan ditulis ke beberapa disk:
        - RAID 0: dibagi rata (striping, tanpa redundansi)
        - RAID 1: mirroring
        - RAID 5: striping + parity

    Jika menggunakan **RAID 5**, controller menghitung **parity** dan menulis data serta parity ke masing-masing disk.
7. **Acknowledgement & Completion:**
    - Setelah data berhasil ditulis, RAID controller mengirim sinyal **ACK** ke CPU.
    - CPU kemudian mengirimkan respon **200 OK / sukses** kembali ke client.

8. **Komunikasi Antar Komponen via System Bus:**
   Semua alur di atas terjadi melalui sistem **bus**, yaitu:
   - **Data Bus:** mengangkut data dari dan ke CPU, RAM, dan storage.
   - **Address Bus:** menunjukkan lokasi target data.
   - **Control Bus:** mengatur timing dan jenis operasi (read/write).

4. Esai Reflektif (Bagian D)

Selama menyusun laporan Ujian Tengah Semester ini, saya mendapatkan wawasan yang jauh lebih dalam tentang bagaimana komponen-komponen utama komputer saling terhubung dan bekerja dalam suatu sistem. Topik studi kasus tentang server berbasis x86 untuk data center membuka pemahaman saya mengenai pentingnya arsitektur komputer dalam dunia nyata, khususnya dalam skala besar seperti pusat data.

Saya menyadari bahwa performa sistem komputer tidak hanya bergantung pada kecepatan prosesor, tetapi juga sangat dipengaruhi oleh efisiensi komunikasi antar komponen, seperti CPU multicore, RAM ECC, dan sistem storage RAID. Pembahasan tentang bus sistem, bandwidth, dan potensi bottleneck memberi saya gambaran konkret tentang bagaimana data berpindah, diproses, dan disimpan dalam sistem yang kompleks.

Studi kasus simulasi komunikasi data juga membuat saya memahami pentingnya koordinasi sinyal dan kontrol antara perangkat keras melalui bus data, bus alamat, dan bus kontrol. Selain itu, saya juga belajar bagaimana kesalahan data dapat diminimalisir dengan teknologi seperti ECC (Error-Correcting Code) dan keandalan data ditingkatkan melalui sistem RAID.

Program konversi bilangan yang saya buat dalam Python juga mengingatkan saya bahwa semua sistem digital bermula dari bilangan biner, dan pemahaman tentang representasi data sangat penting untuk memahami cara komputer bekerja.

Melalui proses pengerjaan laporan ini, saya tidak hanya belajar teori, tetapi juga melihat bagaimana teori tersebut diterapkan langsung dalam arsitektur server modern. Ini membangun fondasi yang kuat bagi saya jika kelak ingin bekerja di bidang teknologi informasi, network engineer, atau sistem komputer dan data center.

Secara keseluruhan, saya merasa pengalaman ini memperkuat pemahaman saya terhadap mata kuliah Organisasi Komputer, sekaligus menumbuhkan minat yang lebih besar untuk mendalami bidang sistem komputer dan arsitekturnya**.**