**UAS**

**Arsitektur & Organisasi Komputer**

****

**Disusun Oleh:**

Prames Ray Lapian - 140810210059

**PROGRAM STUDI S-1 TEKNIK INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PADJADJARAN**

**JATINANGOR**

**2022**

1. CPMK3 [30] Suatu disk dengan *seek time* rata-rata 4 ms, kecepatan rotasi 18.000 rpm, dan sektor berisi 512-byte data dengan 500 sektor per track. Delay rotasi rata-rata1,5 ms. Akan dibaca file yang terdiri dari 3.000 sektor.
   1. Hitung waktu yang diperlukan untuk membaca file tsb, dengan file disimpan pada sektor dalam track yang berurutan!

Waktu untuk membaca track pertama adalah sebagai berikut :

Pencarian rata-rata 4 ms

Delay rotasi rata-rata 1.5 ms

Membaca 500 sektor 3.3 ms = (60/18000)

Jumlah = 8.8 ms

Misalkan track yang tersisa sekarang dapat dibaca tanpa pencarian waktu. Artinya, operasi I/O dapat mengikuti aliran dari disk. Lalu, biasanya kita perlu memperhatikan penundaan rotasi untuk 5 trek yang tersisa. Jadi masing-masing trek berturut-turut dibaca dalam 1.5 + 3.3 = 4.8 ms. Untuk membaca seluruh file:

Total waktu = 8.8 + (5\* 4.8) = 32.8 ms = 0.0328 detik

* 1. Hitung waktu yang diperlukan untuk membaca file, dengan file disimpan pada sektor secara acak!

Pencarian rata-rata 4 ms

Delay rotasi 1.5 ms

Membaca 1 sektor 0.0066 ms

Jumlah = 5.5066 ms

Total waktu = 3000 \* 5.5066 = 16,519.8 ms = 16.5198 detik

* 1. Hitung besar file yang dibaca (dalam MegaByte)!

total data yang akan dibaca = 512 \* 3000

= 1,536,000 byte

= 1.536 Mbyte

1. CPMK3 [30] Sebuah inputan bernilai biner 8 bit: 1101 1110. Data inputan tersebut setelah melalui memory menjadi: 1001 1110.
   1. Tuliskan rumus metoda Hamming code yang digunakan untuk cek error bit data!

C1 = D1 ⊕ D2 ⊕ D4 ⊕ D5 ⊕ D7

C2 = D1 ⊕ D3 ⊕ D4 ⊕ D6 ⊕ D7

C4 = D2 ⊕ D3 ⊕ D4 ⊕ D8

C8 = D5 ⊕ D6 ⊕ D7 ⊕ D8

* 1. Cek kesalahan pada data menggunakan metoda *error detection* dari Hamming code! Tuliskan nilai yang diperoleh dari metoda *error detection*, dan bagaimana mengaitkan ke posisi bit data yag salah!

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Posisi | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| data | 1 | 1 | 0 | 1 | xxx | 1 | 1 | 1 | xxx | 0 | xxx | Xxx |
| var | D8 | D7 | D6 | D5 | C8 | D4 | D3 | D2 | C4 | D1 | C2 | C1 |

Cek Input: 1101 1110

C1 = 0 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 1 = 0

C2 = 1 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 1 = 1

C4 = 1 ⊕ 1 ⊕ 1 ⊕ 1 = 0

C8 = 1 ⊕ 0 ⊕ 1 ⊕ 1 = 1

Posisi data bit menjadi: 1001 1110

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Posisi | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| data | 1 | 0 | 0 | 1 | xxx | 1 | 1 | 1 | xxx | 0 | xxx | Xxx |
| var | D8 | D7 | D6 | D5 | C8 | D4 | D3 | D2 | C4 | D1 | C2 | C1 |

Cek input bit: 1001 1110

C1 = 0 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 0 = 1

C2 = 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 0 = 0

C4 = 1 ⊕ 1 ⊕ 1 ⊕ 1 = 0

C8 = 1 ⊕ 0 ⊕ 0 ⊕ 1 = 0

Perbandingan cek bit lama dan baru:

C8 C4 C2 C1

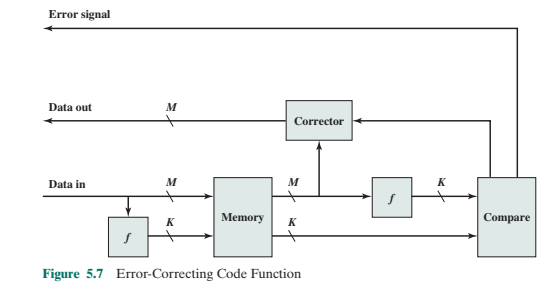
1  0  1  0   (lama)

⊕  0  0   0 1   (baru)

1  0  1  1

Sehingga terjadi error pada data di posisi ke-11 (D7)

* 1. Tuliskan nilai dari setiap variable (CETAK MERAH) yang ada pada di bawah!



Min

Kin

Mf

Kcr

Kco

Mcr

Crr

Mf = 1101 1110

Min = 1101 1110

Kin = 1010

Kco = 1010

Mcr = 1001 1110

Kcr = 0001

Crr = 1011

1. CPMK2, CPMK3 [40] Perhatikan instruksi dalam Bahasa Assembly berikut:

MOV EAX, efa ;// isi register EAX dengan data yang ada di memori pada alamat efa

MOV EBX, EAX ;// isi register EBX dengan data pada register EAX

ADD BAX, EAX, EBX ;// register BAX 🡨 EAX + EBX

MUL BBX, #2, EBX ;// BBX 🡨 2 \* EBX

WRT ffa, BAX ;// simpan/tulis isi register BAX ke memori dengan alamat ffa

WRT ffb, BBX ;// simpan/tulis isi register BBX ke memori dengan alamat ffb

Setiap instruksi dijalankan melalui tahap: F (fetch); D (decode); E (execute), dan W (write). Data bisa diakses oleh instruksi lainya bila sudah melewati tahap E (execute), dan setiap tahap membutuhkan waktu eksekusi satu satuan waktu.

Hitung waktu yang diperlukan untuk mengeksekusi instruksi di atas secara sekuensial!

* 1. Buat proses eksekusi secara *pipeline*!

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Proses/waktu | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 1 | F | D | E | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  | F | D | E | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  | F | D | E |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  | F | D | E |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | E | W |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | E | W |

Jadi, dengan cara Sekuensial CPU akan selesai mengeksekui seluruh perintah dalam 22 Clock

* 1. Hitung waktu eksekusi secara *pipeline*!

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Proses/waktu | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | F | D | E | W |  |  |  |  |  |  |
| 2 |  | F | D | E | W |  |  |  |  |  |
| 3 |  |  |  | F | D | E |  |  |  |  |
| 4 |  |  |  |  | F | D | E |  |  |  |
| 5 |  |  |  |  |  | F | D | E | W |  |
| 6 |  |  |  |  |  |  | F | D | E | W |

Jadi dengan pipeline, CPU akan selesai mengeksekusi seluruh perintah dalam 10 clock.

* 1. Pada proses pipeline, apakah terjadi CONSTRAINTS? Sebutkan jenis CONSTRAINTSnya dan terjadi pada intruksi yang mana?
  2. Buat proses eksekusi secara superscalar dengan masing-masing tahap memiliki 2 unit fungsional! Apakah ada perbaikan waktu dibandingkan dengan proses pipeline, berapa perbedaanya?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Proses/waktu | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | F | D | E | W |  |  |  |  |  |  |
| 2 | F | D | E | W |  |  |  |  |  |  |
| 3 |  | F | D | E |  |  |  |  |  |  |
| 4 |  | F | D | E |  |  |  |  |  |  |
| 5 |  |  | F | D | E | W |  |  |  |  |
| 6 |  |  | F | D | E | W |  |  |  |  |

Jadi dengan pipeline *superscalar*, CPU akan selesai mengeksekusi seluruh perintah dalam 6 clock. Superscalar