
Chapter 4

Internal Memory

Karakteristik

- ⌘ Lokasi
- ⌘ Kapasitas
- ⌘ Satuan transfer
- ⌘ Metode akses
- ⌘ Kinerja
- ⌘ Tipe fisik
- ⌘ Karakteristik fisik
- ⌘ Organisasi

Lokasi

- ⌘ CPU/Prosesor
- ⌘ Internal/utama
- ⌘ External/tambahan

Kapasitas

⌘ Ukuran *Word*

- ☑ Unit organisasi memori. Ukuran dari word = banyaknya bit yang digunakan

⌘ Banyaknya *Word*

- ☑ Atau bytes, dimana 1 byte = 8 bit. Panjang 1 word pada umumnya adalah 8, 16, dan 32

Unit Transfer (1)

⌘ Internal

- ☑ Biasanya dibangun oleh lebar bus data
- ☑ Sama dengan banyaknya saluran data ke dalam dan keluar dari modul memori
- ☑ Merupakan banyaknya bit yang dibaca atau dituliskan ke dalam memori

⌘ External

- ☑ Data yang ditransfer dalam jumlah yang lebih besar dari *word*, yang disebut *block*

Unit Transfer (2)

⌘ Unit pengalamatan

- ☑ Lokasi terkecil di mana pengalamatannya unik
- ☑ Pada beberapa sistem unit pengalamatannya adalah word
- ☑ Cluster on M\$ disks

Metode Akses (1)

⌘ Sequential

- ☑ Memori diorganisir kedalam unit-unit data yang disebut *rekord*
- ☑ Mulai sejak awal dan akan membaca sampai akhir
- ☑ Waktu akses tergantung pada lokasi data dan lokasi sebelumnya (berubah-ubah)
- ☑ ex. tape

Metode Akses (2)

⌘ Langsung

- ☑ Blok-blok individual mempunyai alamat unik
- ☑ Akses dengan cara melompat area dan pencarian sequential
- ☑ Waktu akses tergantung pada lokasi data dan lokasi sebelumnya
- ☑ ex. disk

Hirarki Memori

⌘ Registers

- ☒ Terdapat pada CPU untuk kontrol atau dipakai oleh pemrogram melalui set instruksi mesin.

⌘ Internal atau memori utama

- ☒ Dikembangkan dng suatu cache berkecepatan tinggi
- ☒ Cache perangkat untuk pergerakan data antara memori utama dan register prosesor untuk meningkatkan kinerja.
- ☒ "RAM"

⌘ External memory

- ☒ Disk magnetic

Metode Akses (2)

⌘ Random/acak

- ☑ Pengalamatan individual mengidentifikasi dengan tepat lokasi
- ☑ Waktu akses tidak tergantung pada lokasi dan akses sebelumnya
- ☑ ex. RAM

⌘ Associative/asosiatif

- ☑ Data ditempatkan oleh perbandingan bagian isi penyimpanan
- ☑ Waktu akses tidak tergantung pada lokasi dan akses sebelumnya
- ☑ e.x. cache

Kinerja RAM

⌘ Waktu akses

- ☑ Waktu yang diperlukan untuk operasi baca tulis dari menampilkan alamat sampai operasi penyimpanan data atau penggunaan data

⌘ Waktu siklus memori

- ☑ Waktu diperlukan memori untuk 'recover' sebelum akses berikutnya
- ☑ Waktu siklus adalah waktu akses + recovery

⌘ Kecepatan transfer

- ☑ Kecepatan saat data bisa dipindahkan

Kinerja non-RAM

⌘ Waktu akses

- ☑ Waktu yang dibutuhkan untuk melakukan operasi baca tulis pada lokasi yang diinginkan

⌘ Waktu siklus memori

- ☑ Harus memenuhi $T_N = T_A + \frac{N}{R}$

☑ Dimana T_N = Waktu rata-rata baca tulis N bit

T_A = Waktu akses rata-rata

N = jumlah bit

R = kecepatan transfer dalam bit/sec (bps)

⌘ Kecepatan transfer

- ☑ Kecepatan saat data bisa dipindahkan

Tipe Fisik

⌘ Semiconductor

- ☑ RAM

⌘ Magnetic

- ☑ Disk & Tape

⌘ Optical

- ☑ CD & DVD

⌘ Lainnya

- ☑ Hologram

Karakteristik Fisik

- ⌘ Decay, kerusakan data
- ⌘ Volatility, data hilang saat sumber daya mati
- ⌘ Erasable, mudah dihapus
- ⌘ Konsumsi Daya

Hirarki

- ⌘ Registers
- ⌘ L1 Cache
- ⌘ L2 Cache
- ⌘ Main memory
- ⌘ Disk cache
- ⌘ Disk
- ⌘ Optical
- ⌘ Tape

Penempatan

- ⌘ Selama eksekusi sebuah program, memori direkomendasikan ke cluster
- ⌘ e.g. loops

Cluster ?

Memori Semikonduktor

⌘ RAM

- ☑ Disebut juga memori semikonduktor
- ☑ Read/Write (operasi baca tulis)
- ☑ Volatile
- ☑ Tempat penyimpanan sementara
- ☑ Statis or dinamis

Dynamic RAM (DRAM)

- ⌘ Bits disimpan seperti kapasitor mengisi muatan
- ⌘ Mengisi ulang secara berkala untuk memelihara penyimpanan data
- ⌘ Konstruksi yang paling sederhana
- ⌘ Sel memori kecil (per bit)
- ⌘ Murah
- ⌘ Memerlukan rangkaian penyegar pendukung
- ⌘ Lambat
- ⌘ Memori Utama

Static RAM

- ⌘ Bits disimpan seperti on/off switches
- ⌘ Tidak ada pengisian ulang
- ⌘ Tidak ada penyegaran memori
- ⌘ Konstruksi lebih rumit
- ⌘ Sel memori lebih besar (per bit)
- ⌘ Lebih mahal
- ⌘ Tidak memerlukan rangkaian penyegar
- ⌘ Lebih cepat
- ⌘ Digunakan untuk memori cache

Read Only Memory (ROM)

- ⌘ Nonvolatile
- ⌘ Microprogramming (bab berikutnya)
- ⌘ Pustaka subroutines untuk fungsi yang sering digunakan
- ⌘ Systems programs (BIOS)

Tipe ROM

⌘ Ditulis (diprogram) saat pembuatan

☒ Sangat mahal

⌘ Programmable (hanya satu kali)

☒ PROM

☒ Memerlukan peralatan khusus untuk program

⌘ "sering" di baca

☒ Erasable Programmable (EPROM)

☒ Dapat dihapus dengan UV

☒ Electrically Erasable (EEPROM)

☒ Menulis lebih membutuhkan waktu dari membaca

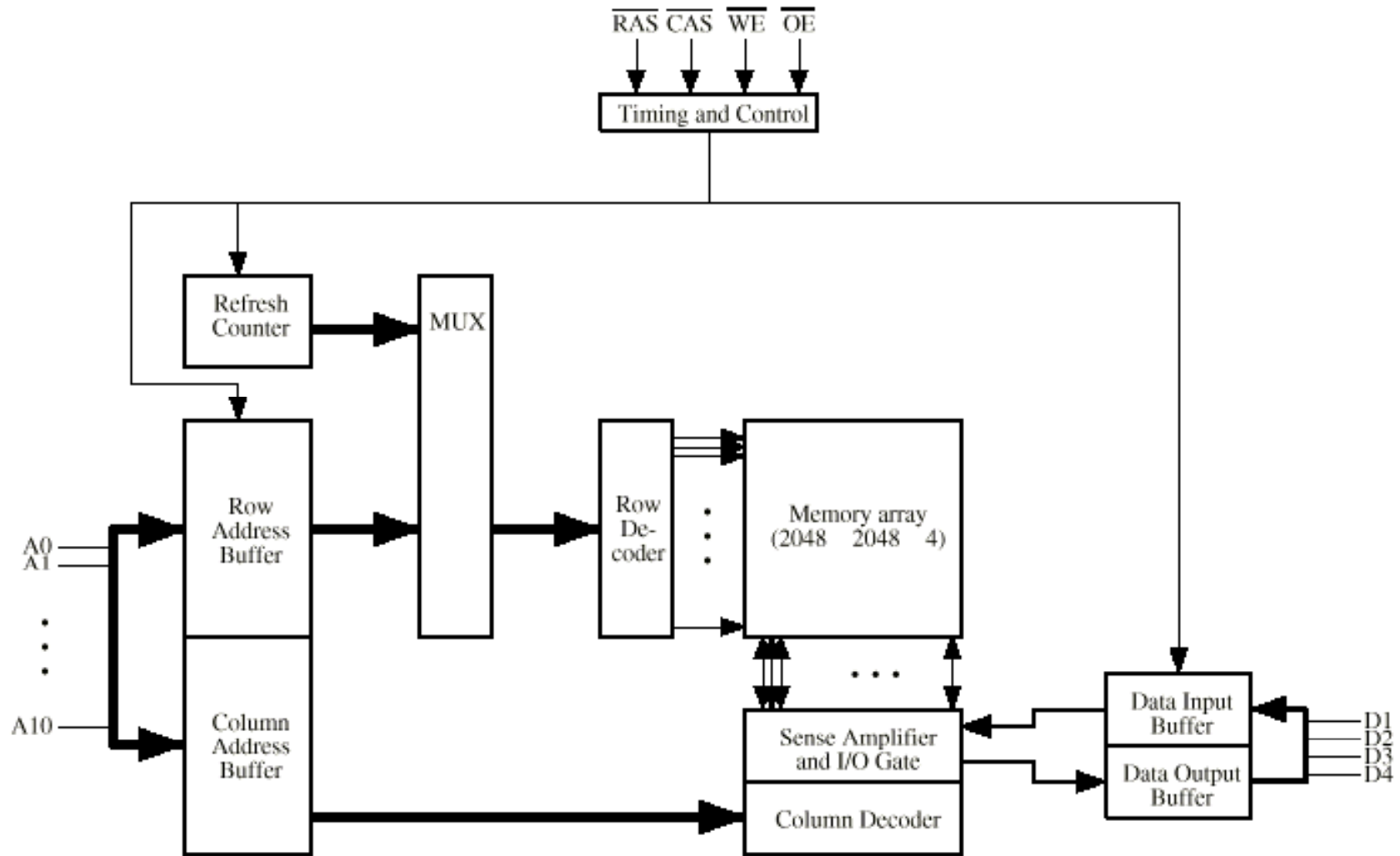
☒ Flash memory

☒ Dapat dihapus secara elektrik

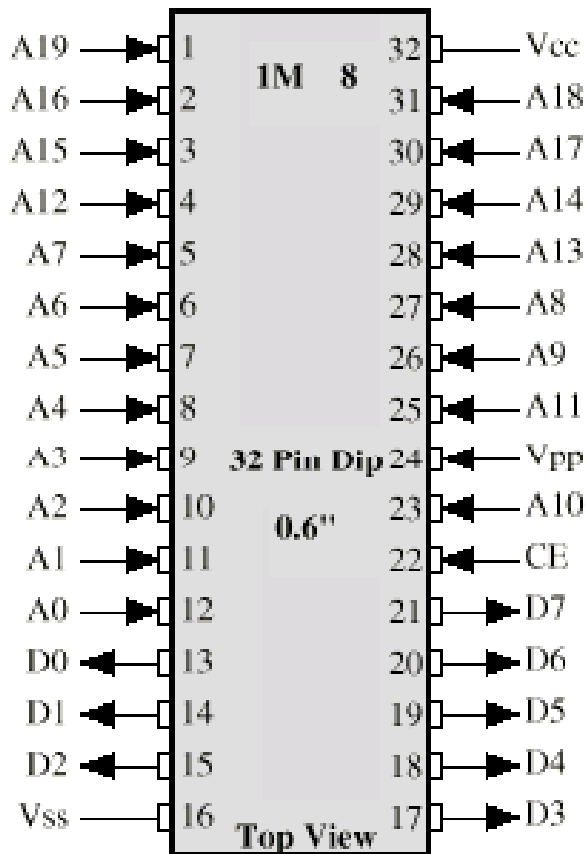
Organisasi secara lengkap

- ⌘ Sebuah chip 16 Mbit bisa diatur menjadi 1 M 16 bit word
- ⌘ Sebuah bit per chip memiliki 16 ruang berkapasitas 1 Mbit, dengan bit 1 dari setiap word berada di chip satu, dst
- ⌘ Sebuah chip 16 Mbit bisa tersusun sebagai array 2048 x 2048 x 4 bit
 - ☒ Dengan susunan tersebut mengurangi pin alamat
 - ☒ Perkalian alamat baris dan kolom
 - ☒ 11 pin untuk alamat ($2^{11}=2048$)
 - ☒ Menambah lebih dari satu pin, menggandakan kapasitasnya sampai 4X

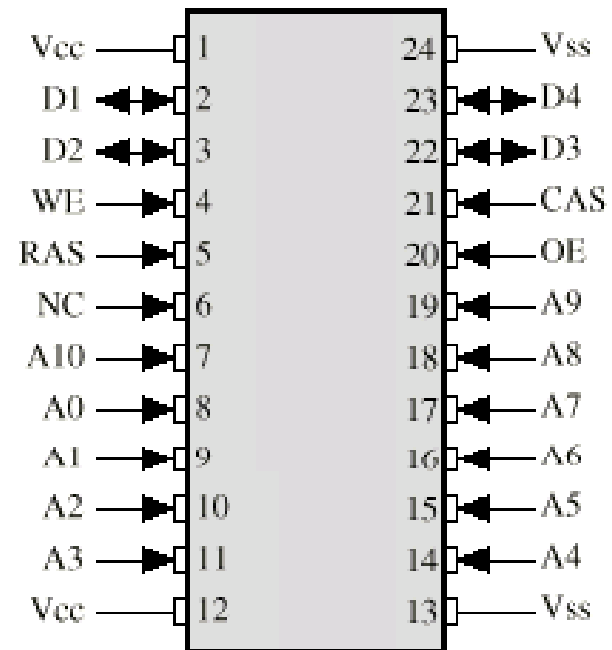
Typical 16 Mb DRAM (4M x 4)



Kemasan



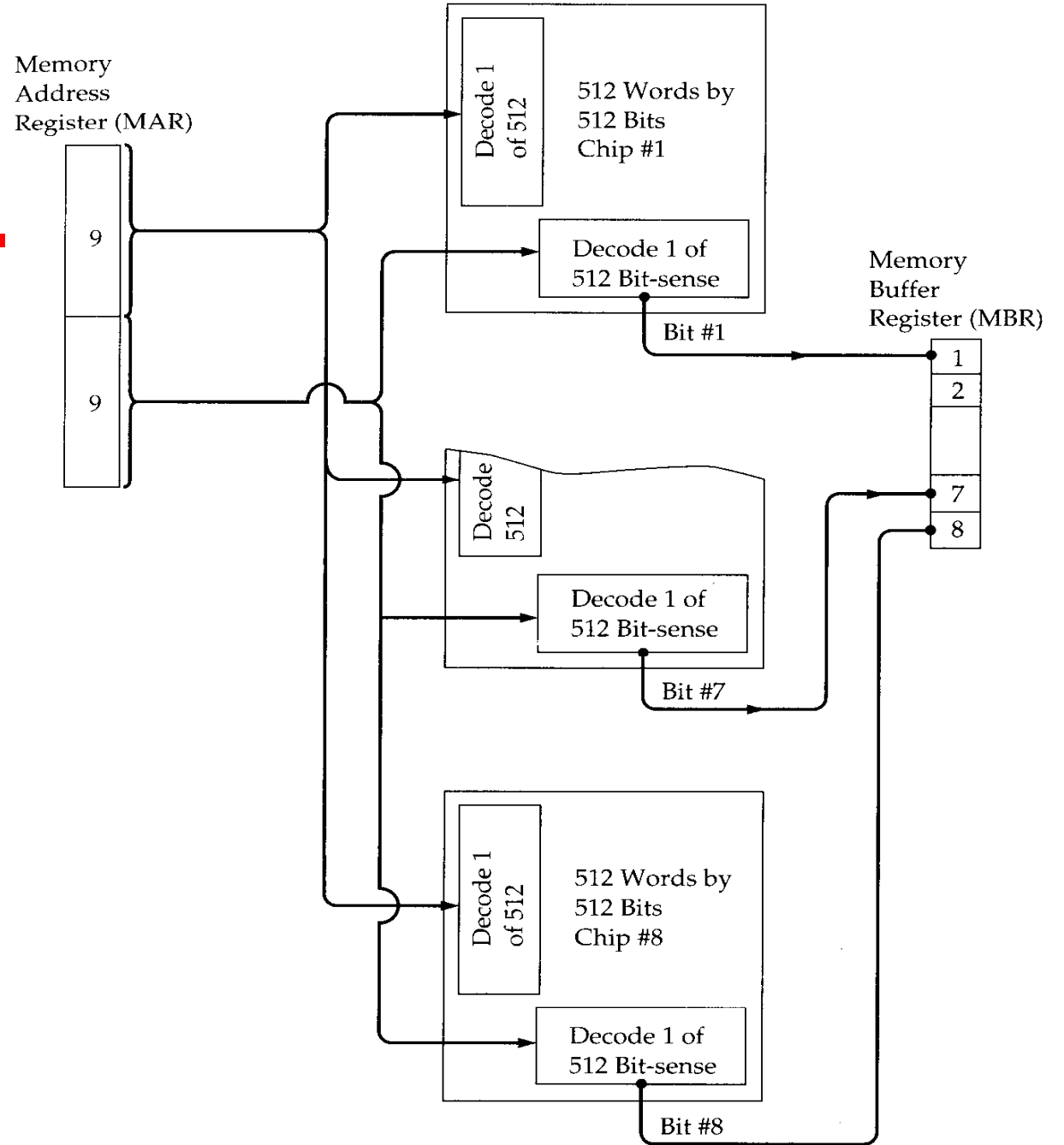
(a) 8 Mbit EPROM



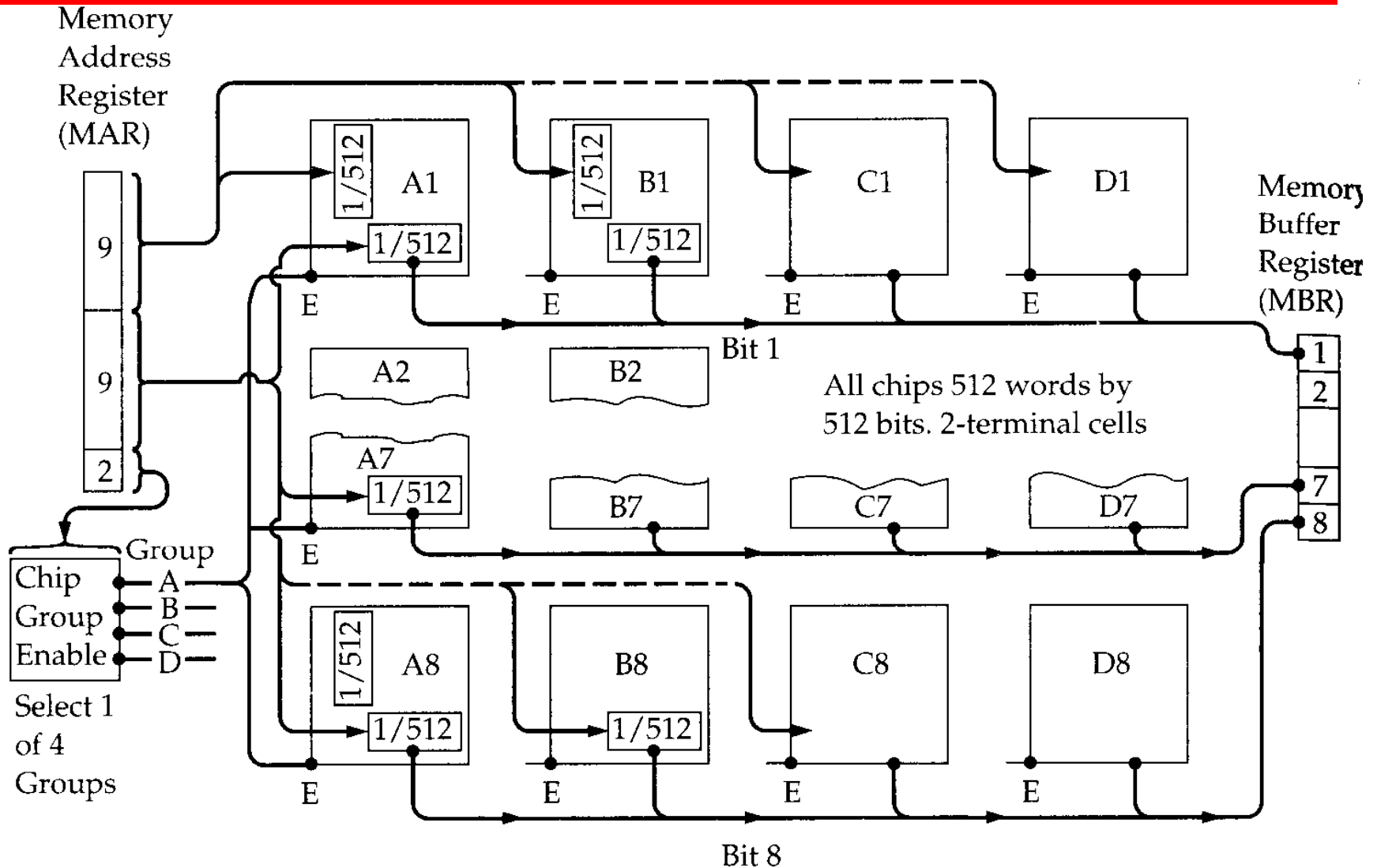
(b) 16 Mbit DRAM

Organisasi Memori

356 Kbyte



Organisasi Memori 1 Mbyte (2)



Koreksi kesalahan

⌘ Hard Failure (kegagalan keras)

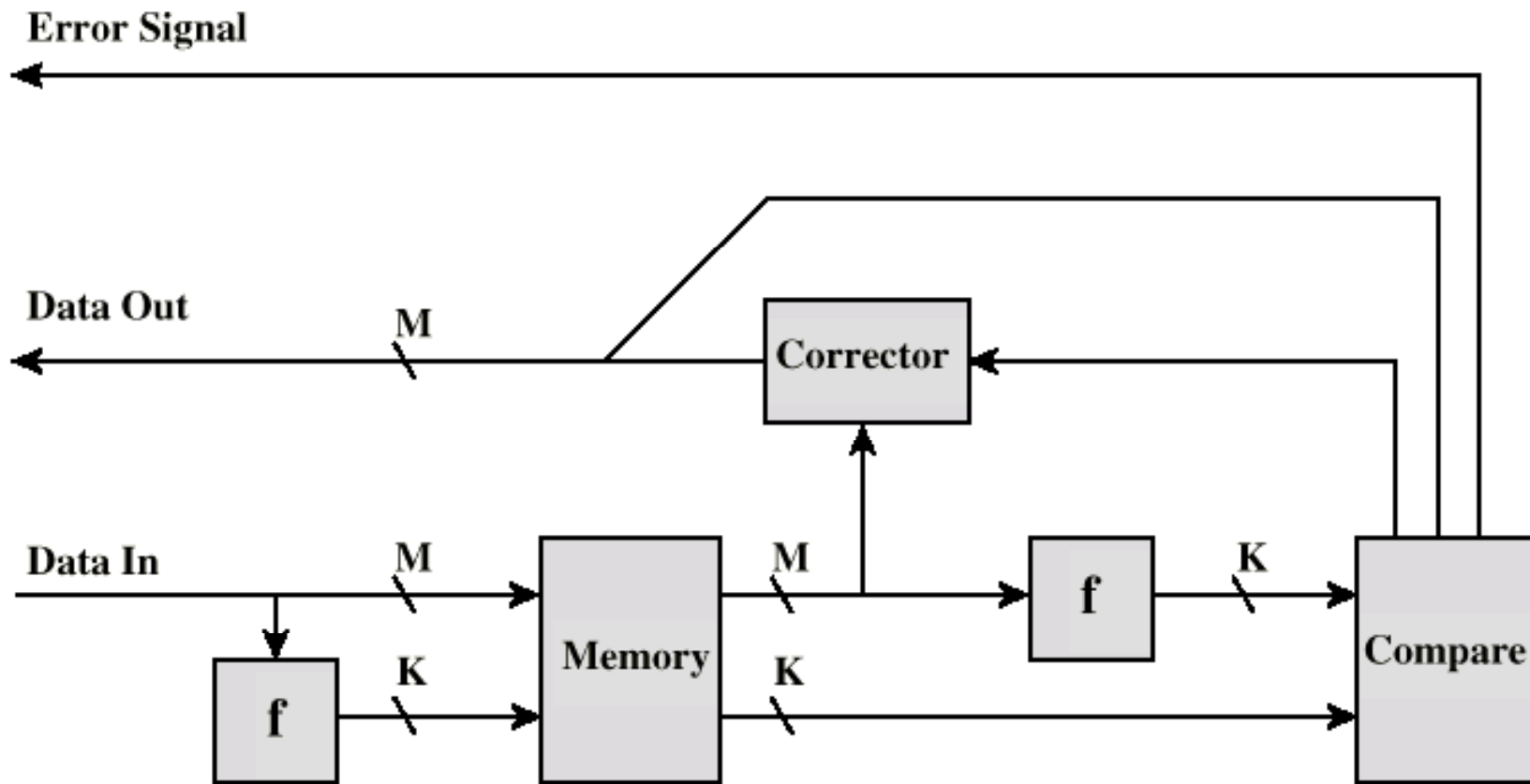
- ☑ Kerusakan fisik permanen, shg sel memori tak dapat digunakan

⌘ Soft Error (kesalahan lunak)

- ☑ acak, non-destructive, mengubah isi satu atau lebih sel memori
- ☑ Tidak merusak memori

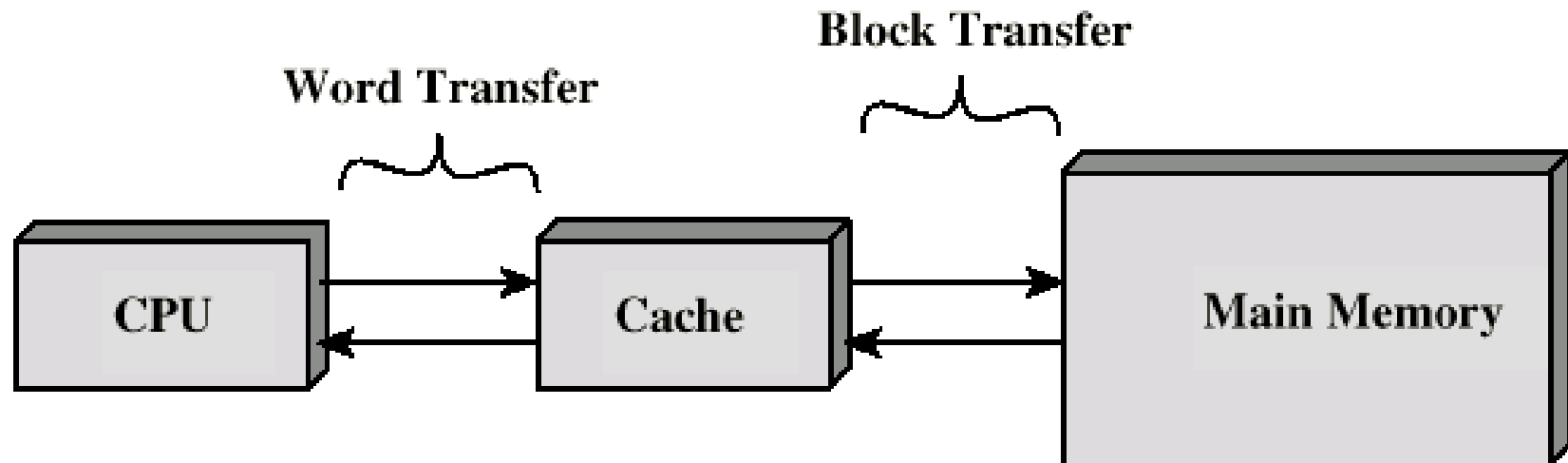
⌘ Mendeteksi kesalahan dengan Kode Pengkoreksi Kesalahan Hamming

Fungsi kode Pengkoreksi kesalahan



Cache

- ⌘ Kecil, cepat, kapasitas besar
- ⌘ Berada antara CPU dan memori utama
- ⌘ Ditempatkan di chip CPU atau modul



Cara kerja Cache

- ⌘ CPU meminta isi dari lokasi memori
- ⌘ Cache memeriksa data ini
- ⌘ Jika ada, diambil dari cache
- ⌘ Jika tidak, akan membaca block dari memori utama untuk cache
- ⌘ Kemudian dikirim dari cache ke CPU
- ⌘ Cache termasuk *tag* untuk mengidentifikasi block mana pada memori utama yang disimpan

Desain Cache

- ⌘ Ukuran cache
- ⌘ Fungsi pemetaan
- ⌘ Algoritma penempatan
- ⌘ Ketentuan menulis
- ⌘ Ukuran block
- ⌘ Jumlah cache

Ukuran cache

⌘ Harga

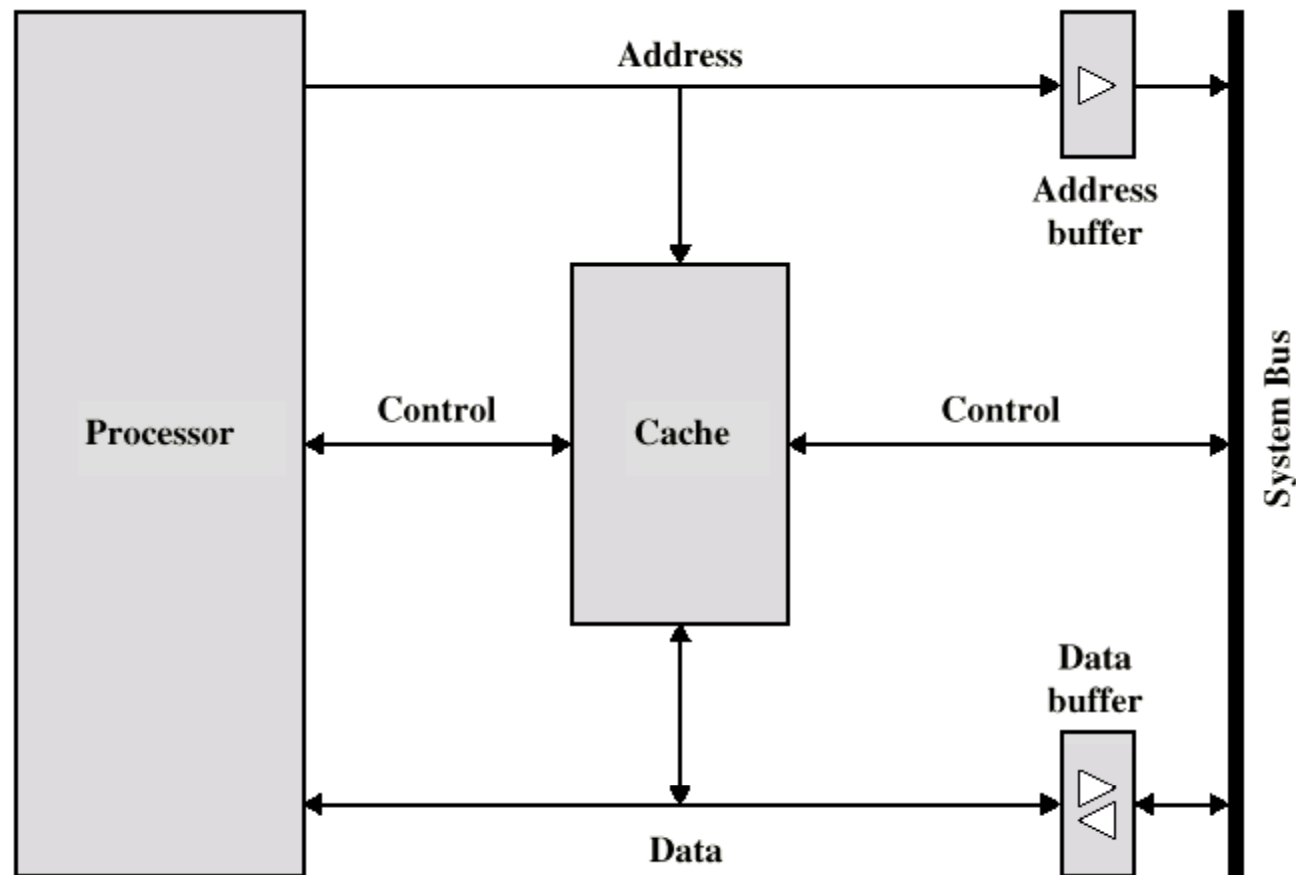
- ☑ Kebanyakan cukup mahal

⌘ Speed

- ☑ Lebih cepat, semakin besar ukuran semakin lambat

- ☑ Memeriksa data pada cache membutuhkan waktu

Organisasi Cache umum



Fungsi pemetaan

- ⌘ Cache dapat menampung 64 Kbyte
- ⌘ Setiap block cache terdiri dari 4 byte, artinya data ditransfer ke memori utama dlm block-block
 - ☑ i.e. cache, $16k = 2^{14}$ baris yang masing-masing besarnya 4 bytes
- ⌘ Memori utama terdiri dari 16 Mbyte
- ⌘ Setiap byte dapat dialamati 24 bit
 - ☑ i.e. memori utama, $2^{24} = 16M$, terdiri dari 4 M block berukuran 4 byte

Pemetaan langsung

- ⌘ Setiap block memori utama memetakan ke satu baris cache saja
 - ☑ i.e. jika block itu adalah cache, harus berada pada satu tempat yang spesifik
 - ⌘ Alamat terdiri dari 2 bagian
 - ⌘ w bit kurang penting, mengidentifikasi word atau byte dlm memori utama
 - ⌘ s bit sisanya, menspesifikasi salah satu dari 2^s block memori utama
- Tag ($s-r$) bit, bagian bit yg paling penting
 r adalah bidang baris

Pemetaan Langsung (2)

Ringkasan sbb:

⌘ Panjang alamat = $(s + w)$ bit

⌘ Jml unit yg dapat dialamati = 2^{s+w} word atau byte

⌘ Ukuran block = ukuran baris = 2^w word atau byte

⌘ Jumlah word pada memori utama = $\frac{2^{s+w}}{2^w}$

⌘ Jumlah baris pada cache = $m = 2^r$

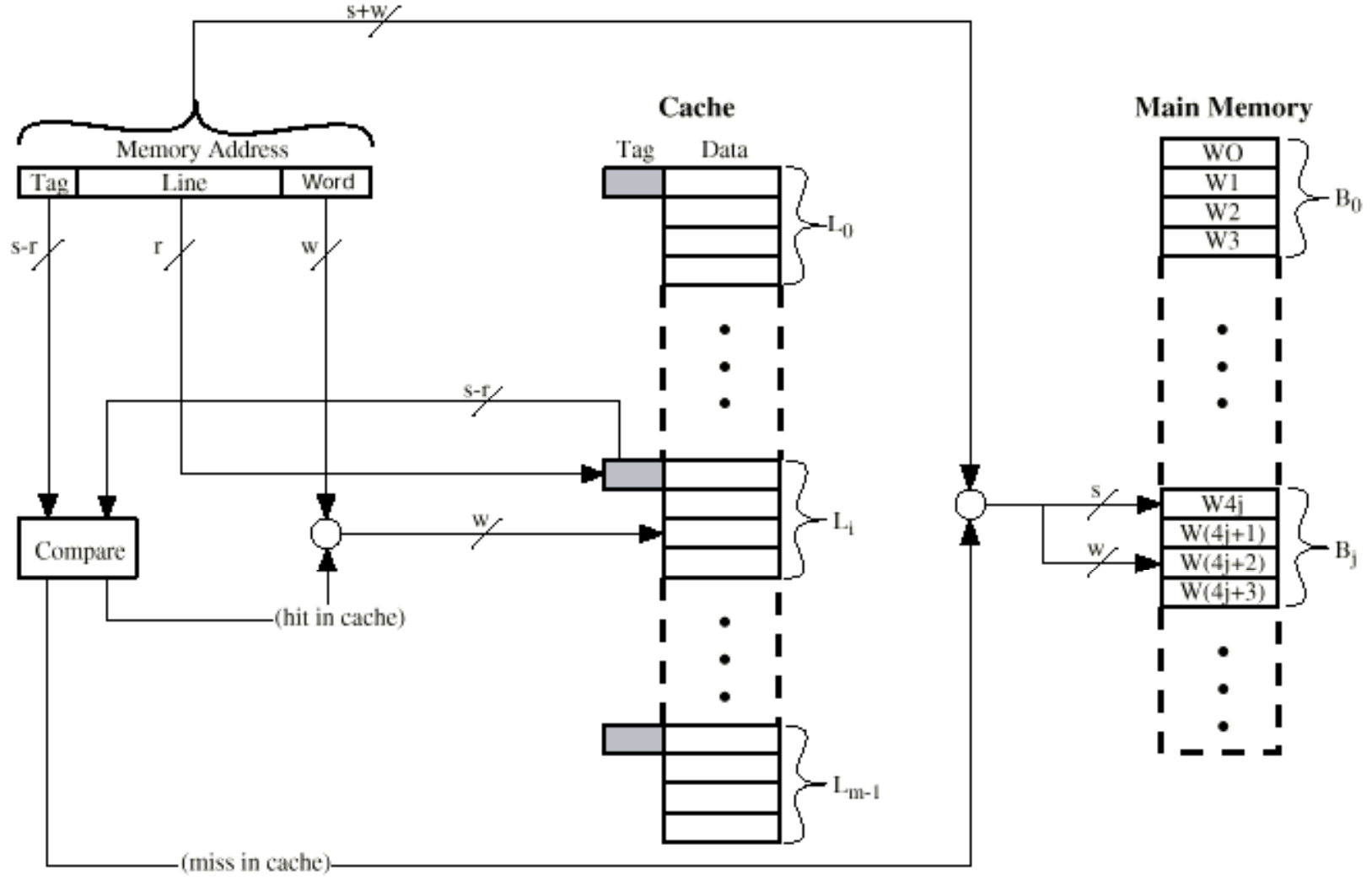
⌘ Ukuran tag = $(s - r)$ bit

Pemetaan Langsung

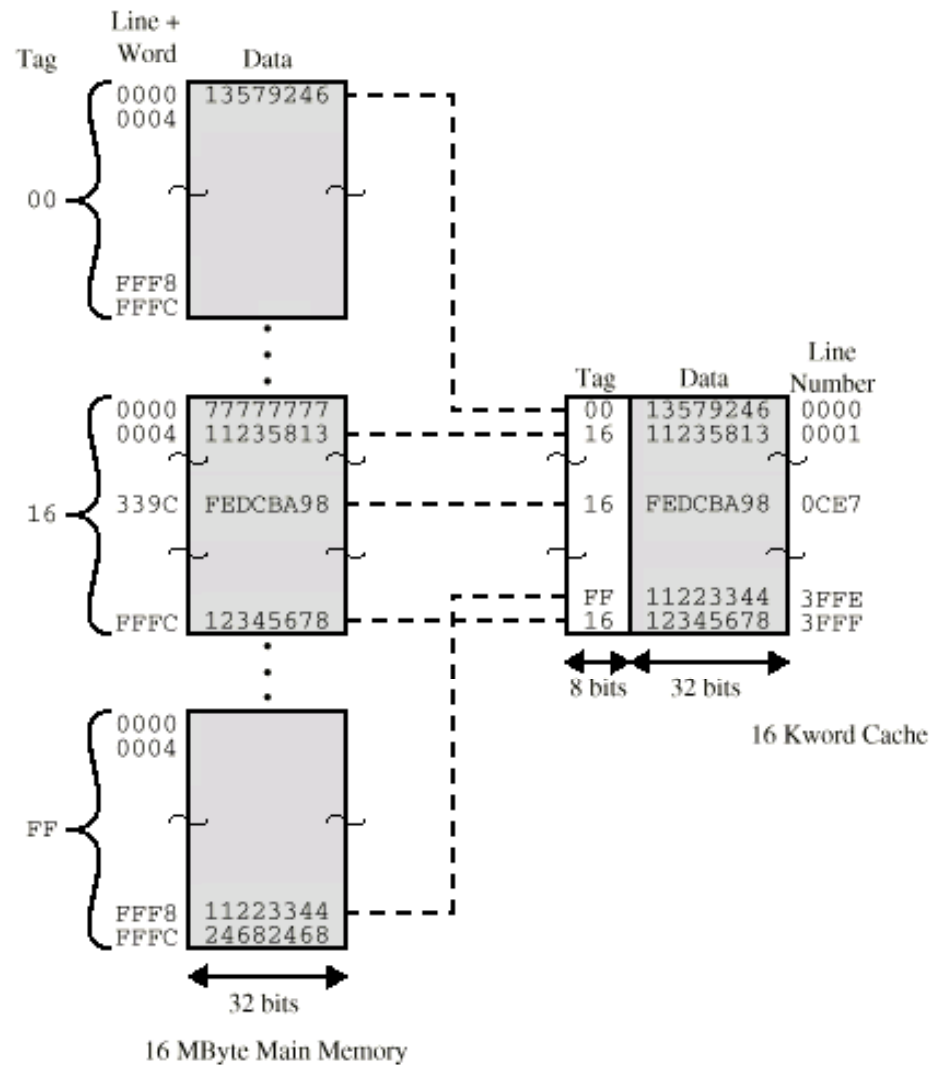
Tabel baris cache

⌘ Baris cache	block Memori utama yang diberikan
⌘ 0	$0, m, 2m, 3m \dots 2^s - m$
⌘ 1	$1, m+1, 2m+1 \dots 2^s - m + 1$
⌘ $m-1$	$m-1, 2m-1, 3m-1 \dots 2^s - 1$

Direct Mapping Cache Organization



Direct Mapping Example



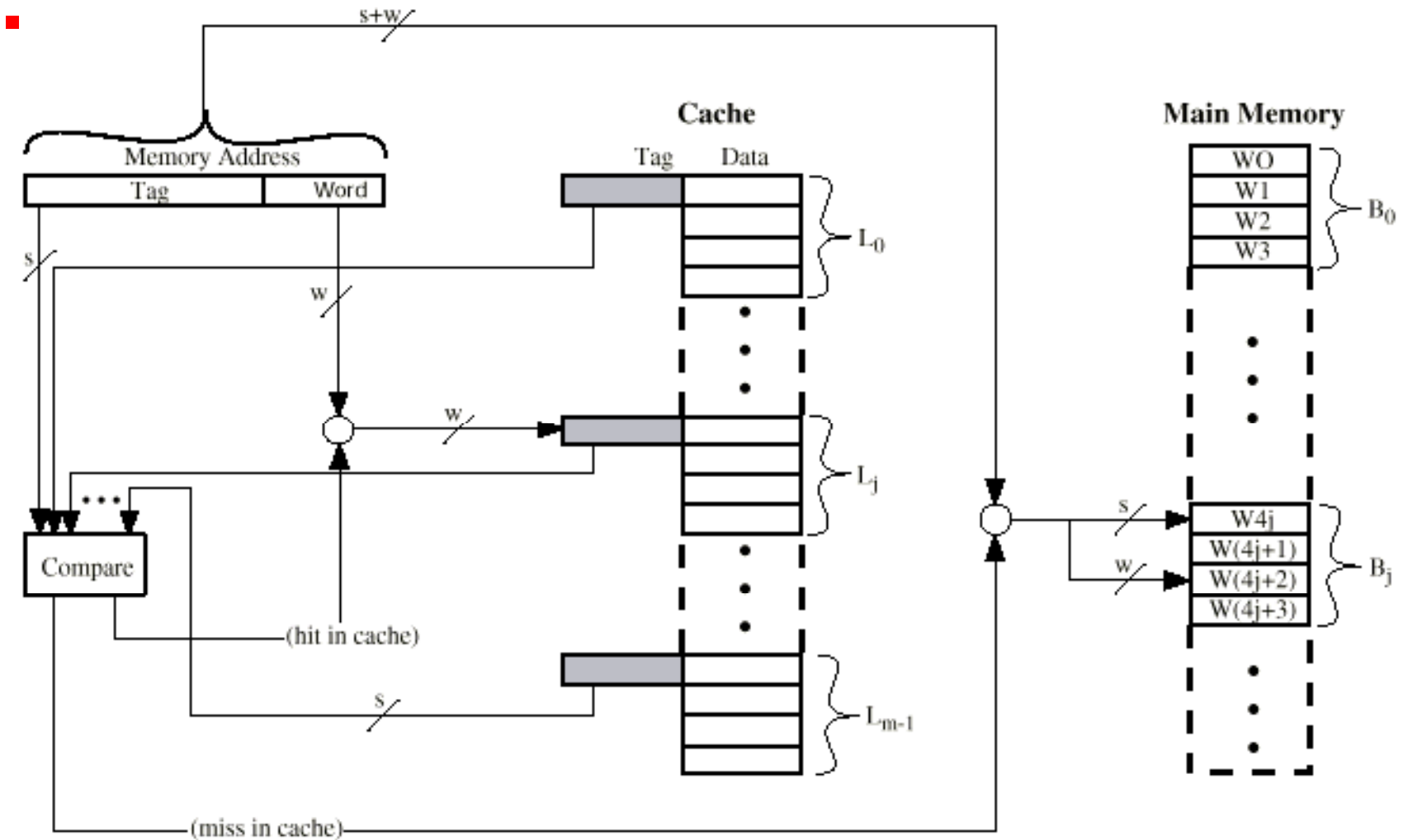
Untung rugi Pemetaan Langsung

- ⌘ Sederhana
- ⌘ Murah
- ⌘ Terdapat lokasi cache yang tetap untuk sebagian block manapun yang ditentukan (rugi)
 - ☒ Jika sebuah program secara berulang mengakses dari dua block yang berbeda memetakan ke baris yang sama, maka block-block secara terus menerus akan ditukar ke cache, maka kerugian cache akan tinggi (fenomena *trashing*)

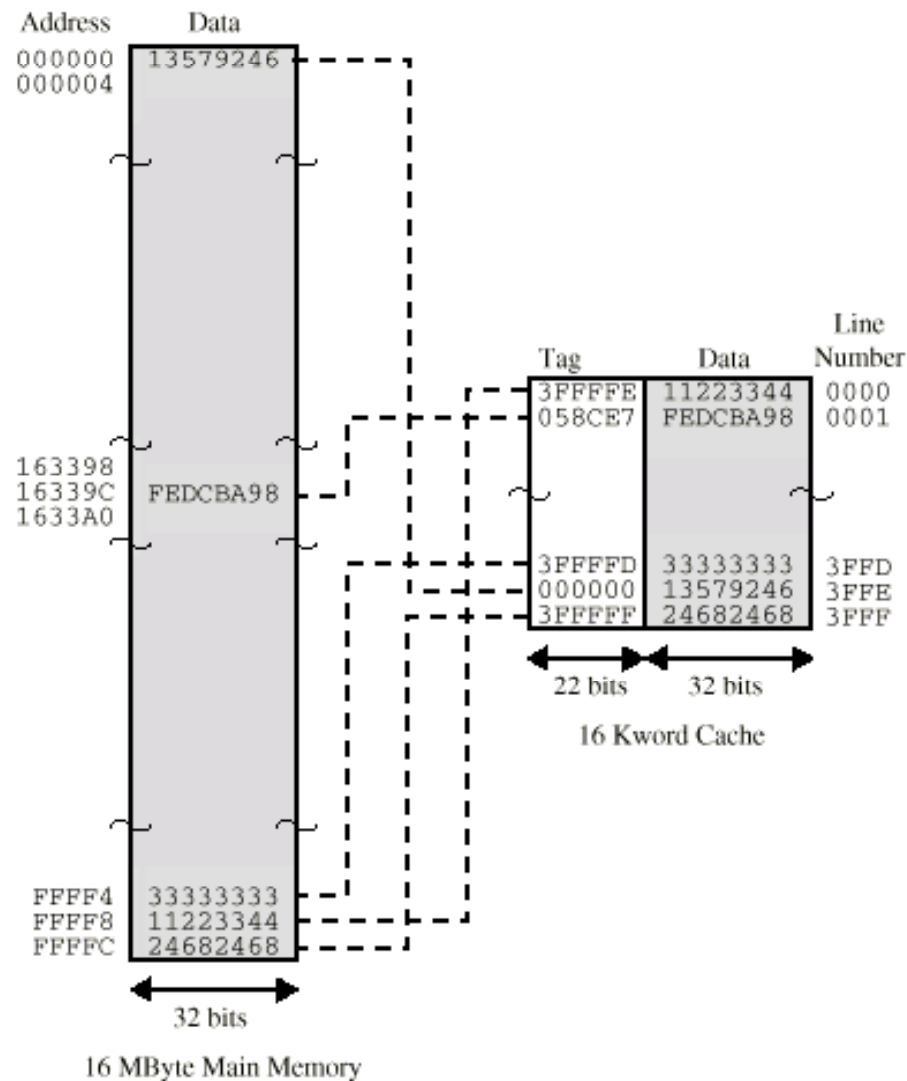
Pemetaan asosiatif

- ⌘ Setiap block memori utama bisa menempati baris cache manapun
- ⌘ Logika kontrol cache menginterpretasikan alamat memori hanya sebagai tag dan word
- ⌘ Tag secara unik mengidentifikasi block memori utama
- ⌘ Setiap baris tag diperiksa apakah suatu block berada pada cache atau tidak

Organisasi cache asosiatif secara penuh



Contoh pemetaan asosiatif



Struktur Pengalamatan Pemetaan Asosiatif

Tag 22 bit	Word 2bit
------------	-----------

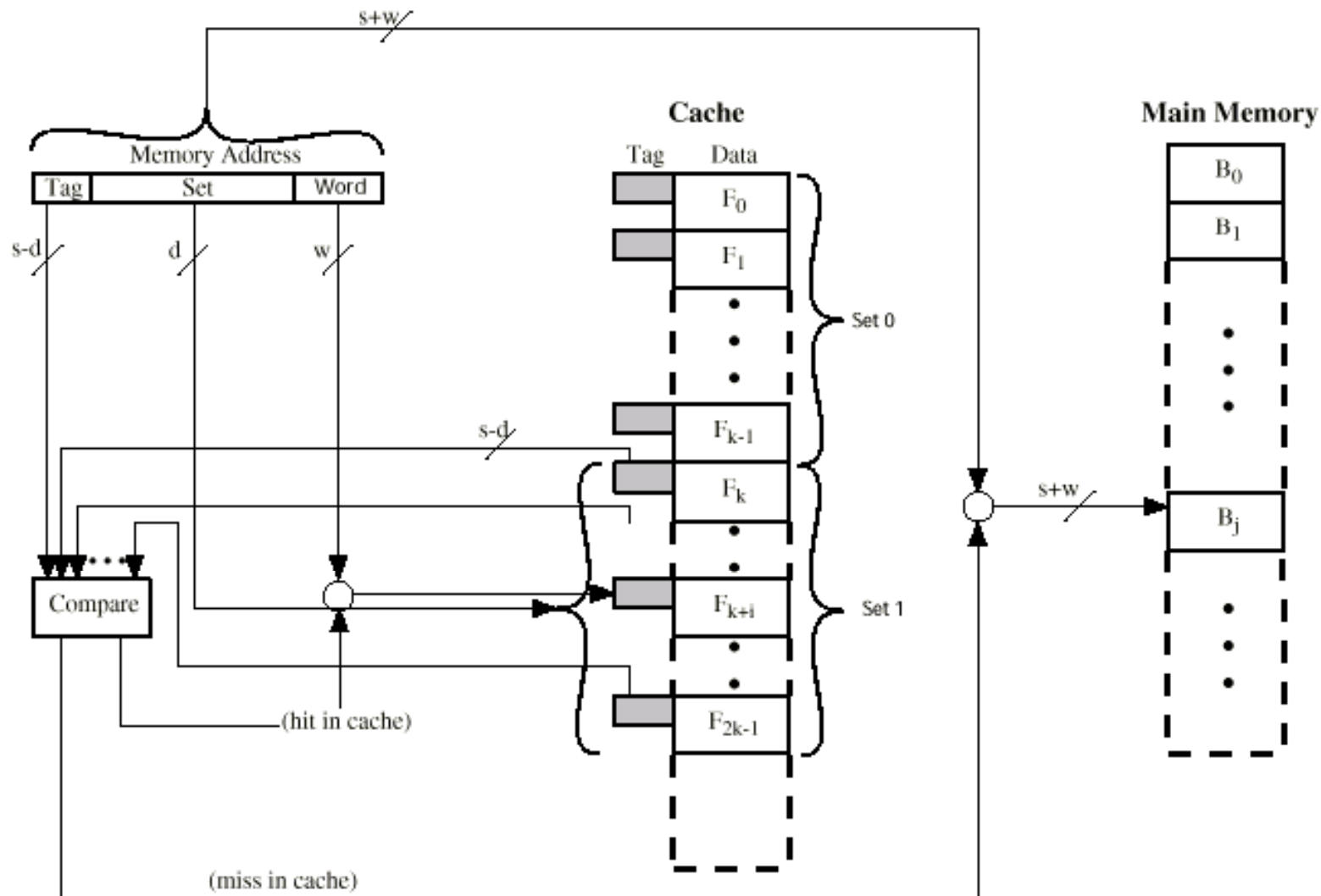
- ⌘ Tag 22 bit disimpan dlm blok data 32 bit pd setiap baris cache
- ⌘ Bandingkan tag field dng tag entry pd cache
- ⌘ 2 bit yg paling signifikan mengidentifikasi address, dmn 16 bit word terdiri dari 32 bit data block
- ⌘ e.g.

Address	Tag	Data	Cache line
FFFFFC	FFFFFC	24682468	3FFF

Pemetaan Asosiatif Set

- ⌘ Cache dibagi menjadi beberapa set
- ⌘ Setiap set terdiri dari beberapa baris
- ⌘ Pemetaan blok ke sembarang baris dalam set
 - ☑ e.g. Blok B dapat dipetakan ke sembarang set i
- ⌘ e.g. 2 baris per set
 - ☑ Pemetaan asosiatif set dua arah
 - ☑ Pemetaan dengan dua saluran pada masing-masing set

Asosiatif Set Dua arah Organisasi Cache



Struktur Pengalamatan Pemetaan Asosiatif Set

Tag 9 bit	Set 13 bit	Word 2 bit
-----------	------------	------------

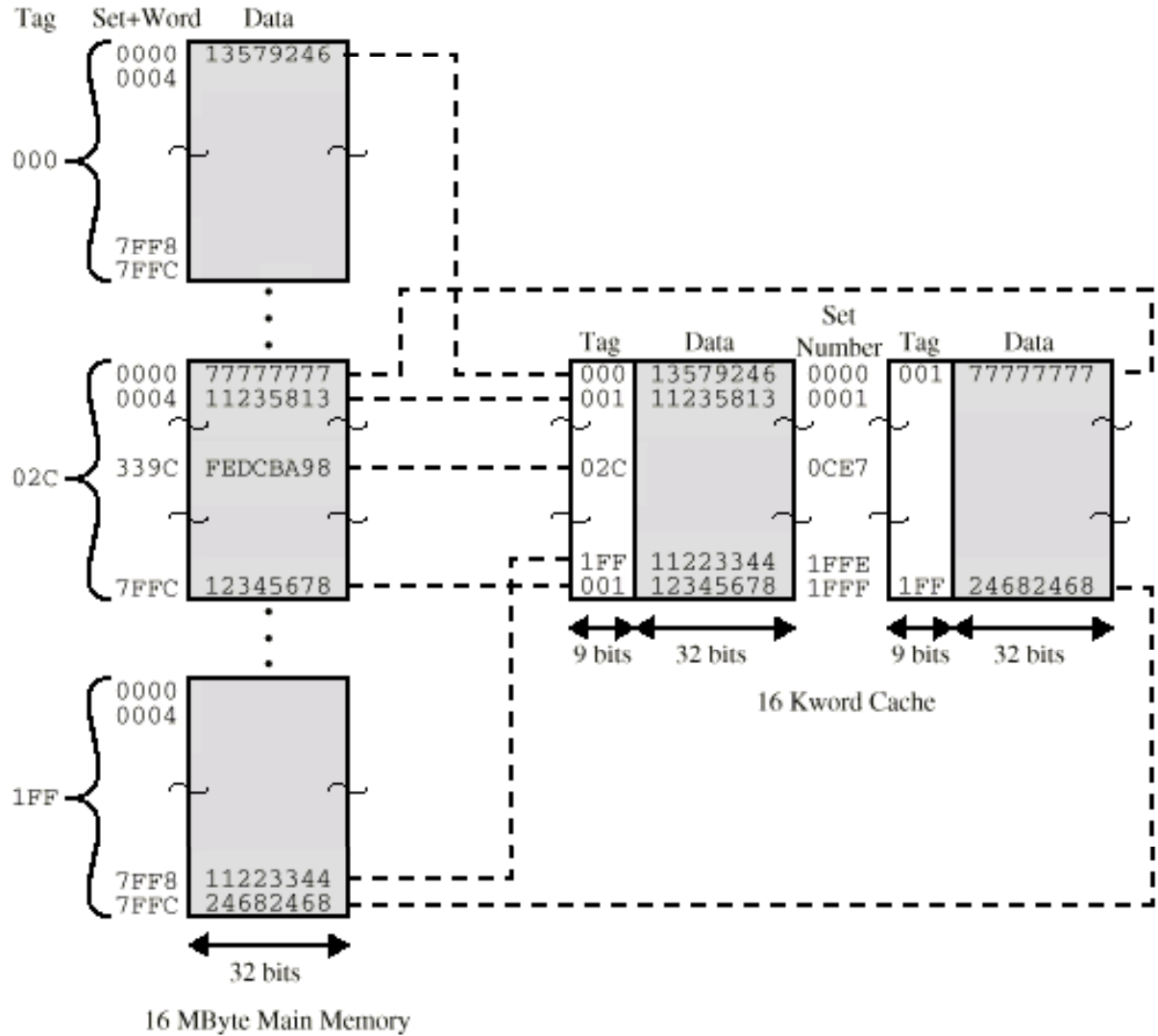
⌘ Gunakan set field untuk menentukan cache set

⌘ Bandingkan tag field

⌘ e.g

Address	Tag	Data	Set number
1FF 7FFC	1FF	12345678	1FFF
001 7FFC	001	11223344	1FFF

Contoh Pemetaan Asosiatif Set Dua Arah



Algoritma Penggantian (1)

Pemetaan langsung

- ⌘ Ketika sebuah blok baru dibawa ke cache, maka salah satu blok harus ada yang diganti
- ⌘ Tidak mempunyai pilihan
- ⌘ Hanya satu baris bagi sembarang blok tertentu
- ⌘ Mengganti baris tersebut

Algoritma Penggantian (2)

Asosiatif dan Asosiatif Set

- ⌘ Implementasi Algoritma Hardware (kecepatan)

- ⌘ Least Recently used (LRU)

- ⌘ e.g. pada 2 saluran asosiasi set

 - ☑ Mana yang termasuk LRU dari 2 blok tersebut?

- ⌘ First in first out (FIFO)

 - ☑ Mengganti block, dari cache yang paling lama digunakan

- ⌘ Least frequently used

 - ☑ Mengganti block dengan penggunaan paling jarang

- ⌘ Acak

Write Policy

- ⌘ Block Cache tidak boleh overwrite kecuali memori yang paling baru
- ⌘ Multiple CPUs boleh mempunyai individual caches
- ⌘ I/O berhubungan langsung dengan address main memory

Tugas :

- ⌘ cari tentang sistem cache Pentium II
- ⌘ Perkembangan terbaru RAM

SDRAM

