Organisasi dan Arsitektur Komputer

Pertemuan 6a: Cache Memori Mapping

Mapping *Cache-Main memory*

(a)Direct Mapping ✓

(b) *Associative Mapping*

(c) *Set Associative Mapping*

Direct Mapping - Kelebihan/kekurangan

(+) Sederhana

(+) Mudah diimplementasikan

(+) Tidak mahal

(-) Lokasi mapping setiap blok sudah tertentu (tidak fleksibel)

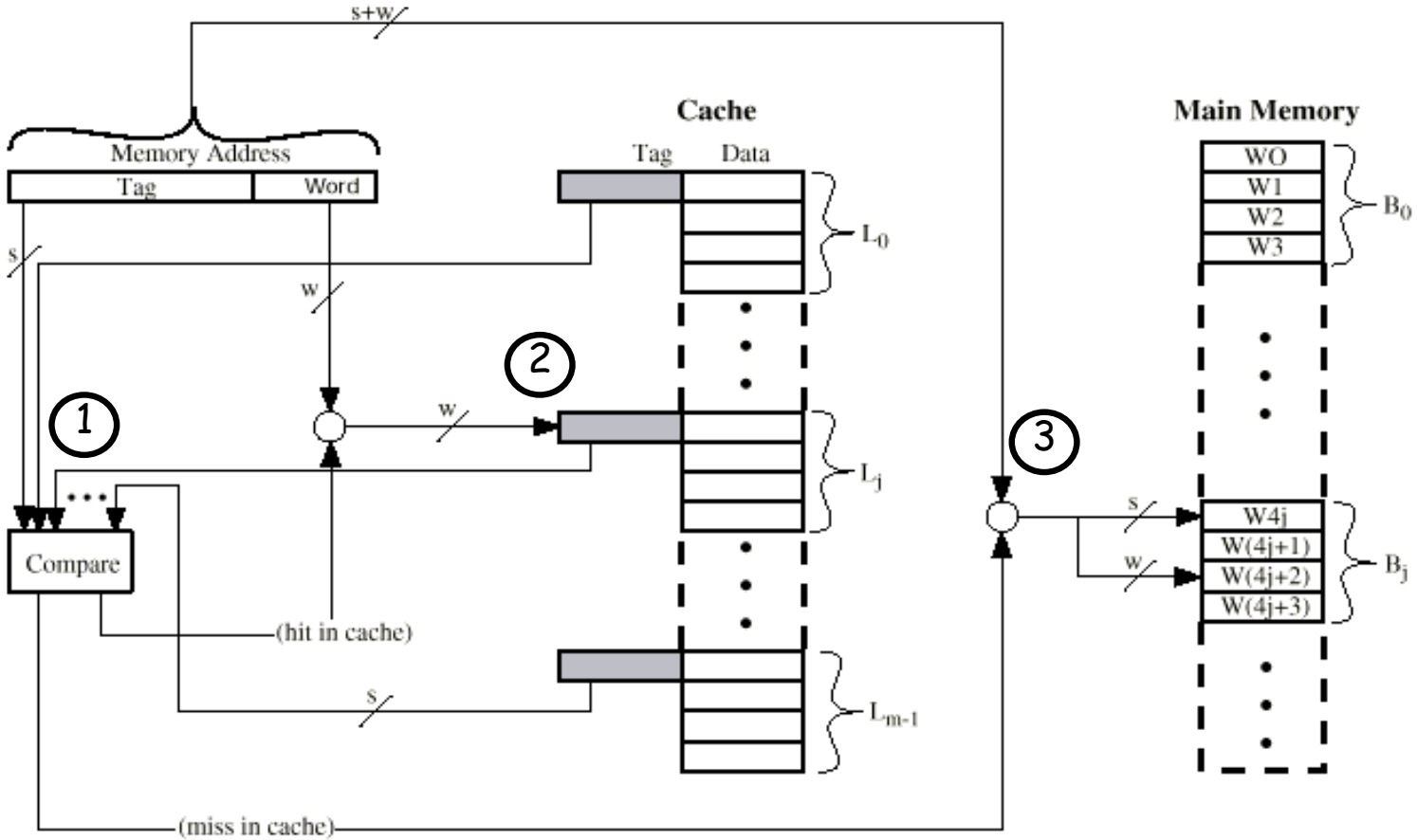
(-) Dapat terjadi *thrashing* bila program mengakses 2 blok yang terletak pada line cache yang sama secara berulang-ulang → terjadi proses swap memori berkali-kali → hit ratio menjadi rendah

Associative Mapping

- Format alamat memori: (dari sisi *cache*)

- Alamat memori diinterpretasikan sebagai tag dan word
- Tag merupakan identitas blok memori
- Setiap satu baris cache mempunyai satu tag
- Tag menjadi kata kunci dalam setiap pencarian data

Associative Mapping - Baca data (1)



Associative Mapping - Baca data (2)

1. CPU membandingkan nomor tag yang akan dibaca dengan semua nomor tag yang ada di *cache* secara bersamaan

2. Bila nomor tag tsb ada di *cache* (*cache hit*) → pilih word yang diinginkan yang terletak pada nomor tag yang diinginkan

3. Bila nomor tag tsb tidak ada di *cache* (*cache miss*) → ambil (*fetch*) satu blok data sesuai dengan nomor tag yang diinginkan

Associative Mapping - Contoh (1)

Diketahui:

- *Main memory* berukuran 16 MByte
- *Cache* berukuran 64 kByte
- 1 alamat = 1 byte
- 1x transfer data = 1 blok memori = 1 *line cache* = 4 byte = 4 alamat
- Sebutkan jumlah bit untuk tag dan word (w) !
- Gambarkan mappingnya !
- Berikan contohnya !

Maka:

- Memori 16 Mbyte = 16 M alamat = $2^4 . 2^{20} = 2^{24}$ → Jumlah bit alamat yang diperlukan = 24 bit
- Jumlah bit identitas word (w) = 2 bit (1 blok = 4 alamat = 2^2)
- Jumlah bit tag = 24 – 2 = 22 bit

Format alamat memori: (dari sisi *cache*)

Associative Mapping - Contoh (2)

Jumlah **line** cache = 64 kbyte/4 byte = 16 k line

16 k = 2^4 . 2^{10} = 2^{14} → Jumlah bit line = 14 bit

Jumlah blok memori = 16 Mbyte/4 byte = 4 M blok = jumlah tag (2^{22})

1 *line cache* ≈ 1 blok memori (tag), maka:

1 line cache mempunyai kemungkinan ditempati oleh **4 M blok** yang berbeda

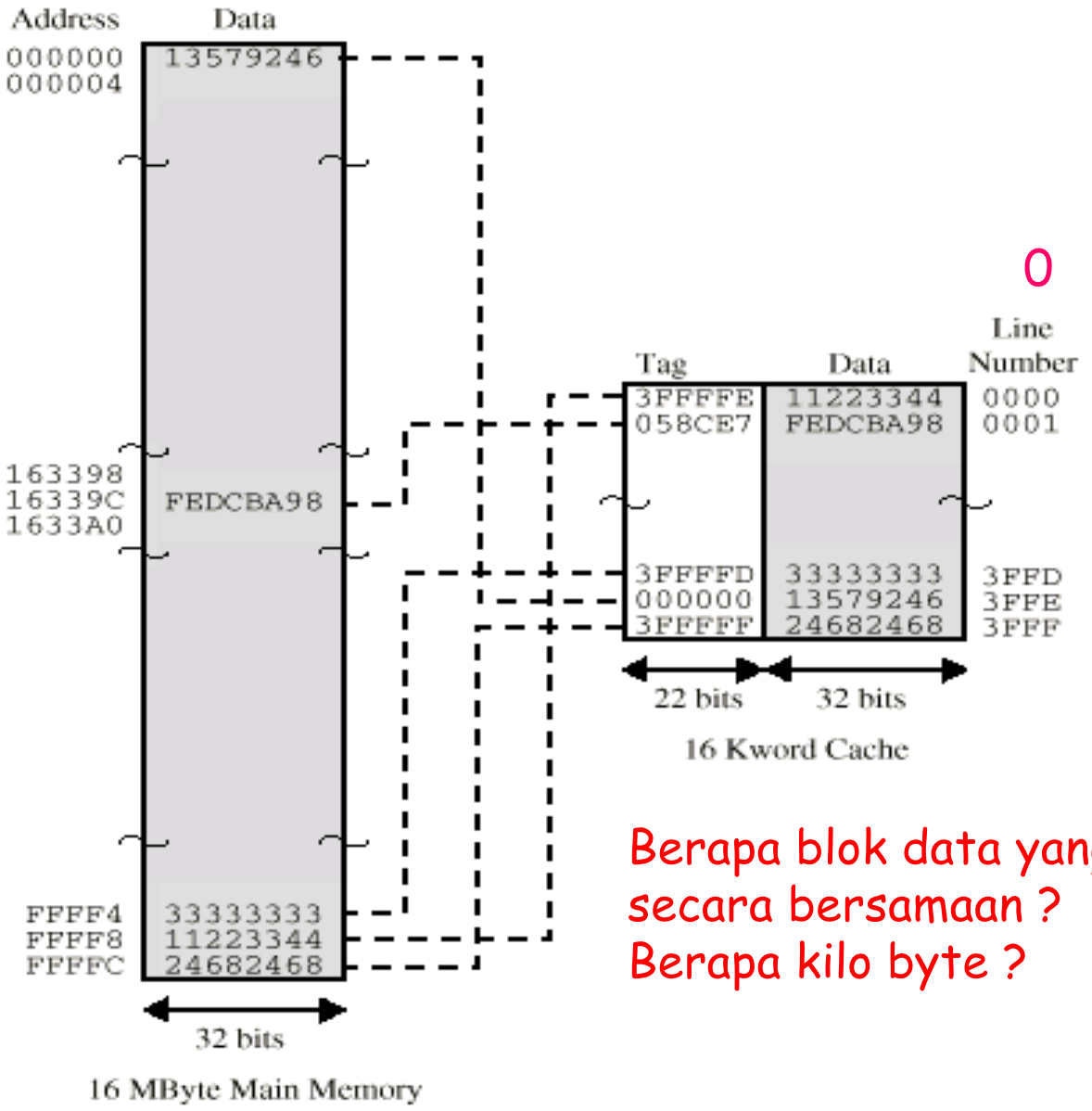
Setiap tag dapat menempati nomor line yang mana saja (tidak berhubungan dengan nomor baris)

Setiap blok memori mempunyai satu pasangan nomor

3)

00

0 5 8 C E 7



Berapa blok data yang dapat ditampung di *cache* secara bersamaan ?
Berapa kilo byte ?

Associative Mapping - Kelebihan/kekurangan

(+) Mapping setiap blok dapat dilakukan secara fleksibel (tidak terikat pada nomor line tertentu)

(+) Dapat mengatasi masalah *thrashing*

(-) Diperlukan rangkaian yang lebih rumit untuk membandingkan semua tag secara paralel, karena jumlah tag sangat banyak

Set Associative Mapping (1)

Cache dibagi menjadi sejumlah set

Setiap set terdiri dari sejumlah baris

Setiap blok memori dipasangkan ke nomor set tertentu, tetapi boleh menempati baris mana saja dalam satu set

Misal: jika tiap set terdiri dari 2 baris, maka:

- Model pemetaannya disebut: *2-way set associative mapping*
- Setiap blok memori boleh menempati satu dari dua baris yang tersedia asalkan masih dalam satu set

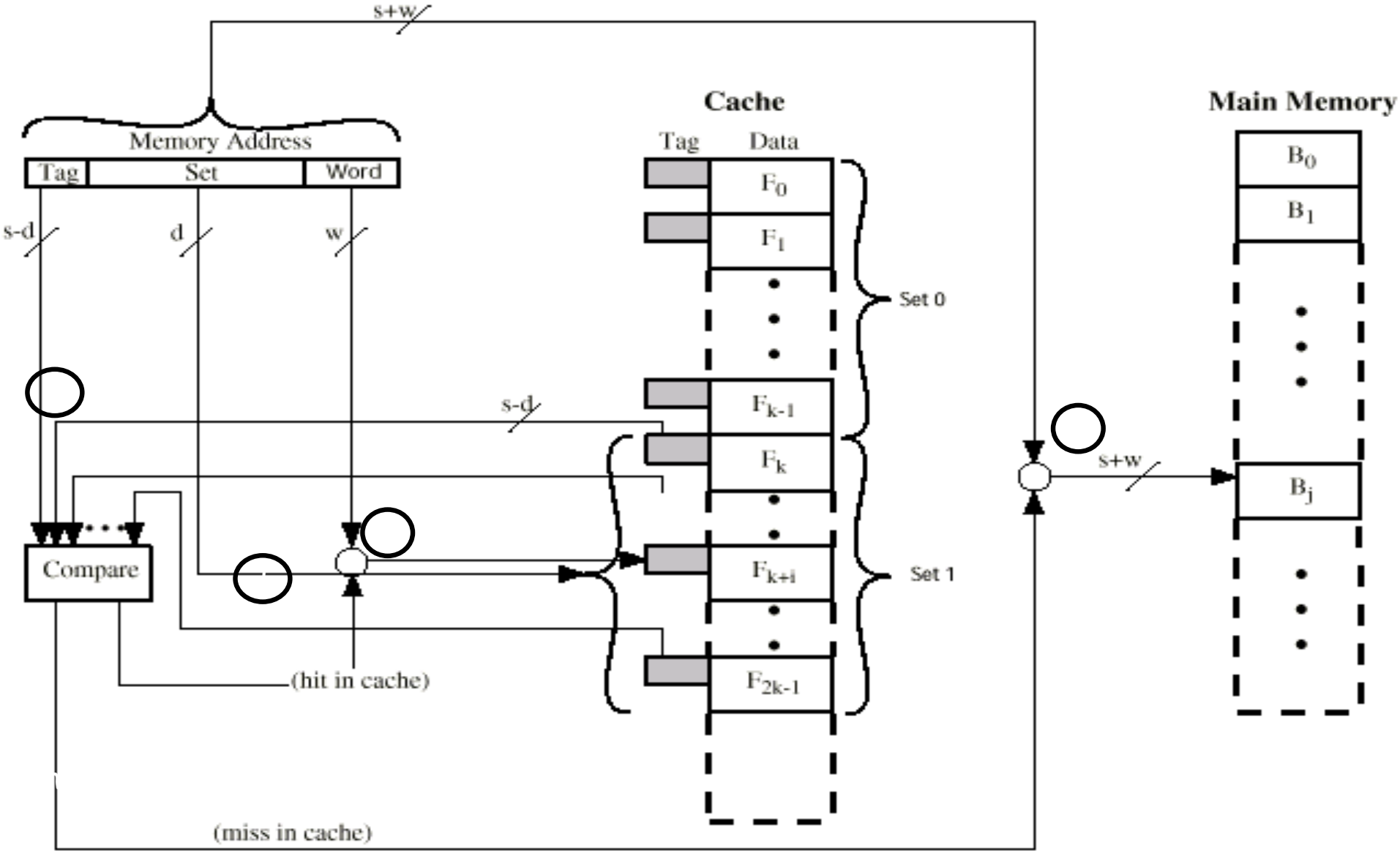
***Set Associative Mapping* (2)**

Format alamat memori: (dari sisi *cache*)

Untuk keperluan akses *cache* → setiap alamat memori dibagi menjadi 2 bagian:

- word(w) = bit-bit identitas word atau byte di dalam blok memori
- s = bit-bit identitas blok memori
  - set field (v): bit-bit nomor set
  - tag (s-v): bit-bit identitas blok data yang ada di memori

Set Associative Mapping - Baca data (1)



Set Associative Mapping - Baca data (2)

- CPU membandingkan nomor tag yang akan dibaca dengan semua nomor tag di *cache*. Tag dalam satu set dibandingkan bersama-sama
- Bila nomor tag tsb ada di cache (*cache hit*) → pilih word yang diinginkan yang terletak pada nomor set yang diinginkan
- Bila nomor tag tsb tidak ada di cache (*cache miss*) → ambil (*fetch*) satu blok data sesuai dengan nomor tag dan nomor set yang diinginkan

Set Associative Mapping – Contoh (1)

Diketahui:

Main Memory berukuran 16 MByte

Cache berukuran 64 kByte

1 alamat = 1 byte

1x transfer data = 1 blok memori = 4 byte = 4 alamat

Model mapping = 2 way set associative

Sebutkan jumlah bit untuk tag (s-v), set (v), dan word (w) !

Gambarkan mappingnya !

Berikan contohnya !

Maka:

Memory 16 Mbyte = $2^4 . 2^{20} = 2^{24}$ → Jumlah bit alamat yang diperlukan = 24 bit

Jumlah bit identitas word (w) = 2 bit (1 blok = 4 byte = 2^2)

Jumlah ***line*** *cache* = 64 kbyte/4 byte = 16 k line

1 set = 2 line → jumlah set = 16k/2 = 8 k set

8 k = $2^3 . 2^{10} = 2^{13}$ → Jumlah bit set = 13 bit (0000 - 1FFF)

Jumlah bit tag = 24 – 13 – 2 = 9 bit (range: 000 – 1FF)

Set Associative Mapping – Contoh (2)

Format alamat memori: (dari sisi cache)

Jumlah alamat memori tiap tag = $2^{13+2} = 2^5 \cdot 2^{10}$ = 32 k alamat

Range alamat memori tiap tag (dalam format 24 bit) = 000000 – 007FFF, maka alamat memori : 000000, 008000, 010000, …, FF8000 selalu terletak pada set nomor yang sama (0000)

007FFF = **0000 0000 0**111 1111 1111 1111 + 1 → **0000 0000 1**000 0000 0000 0000 = 008000

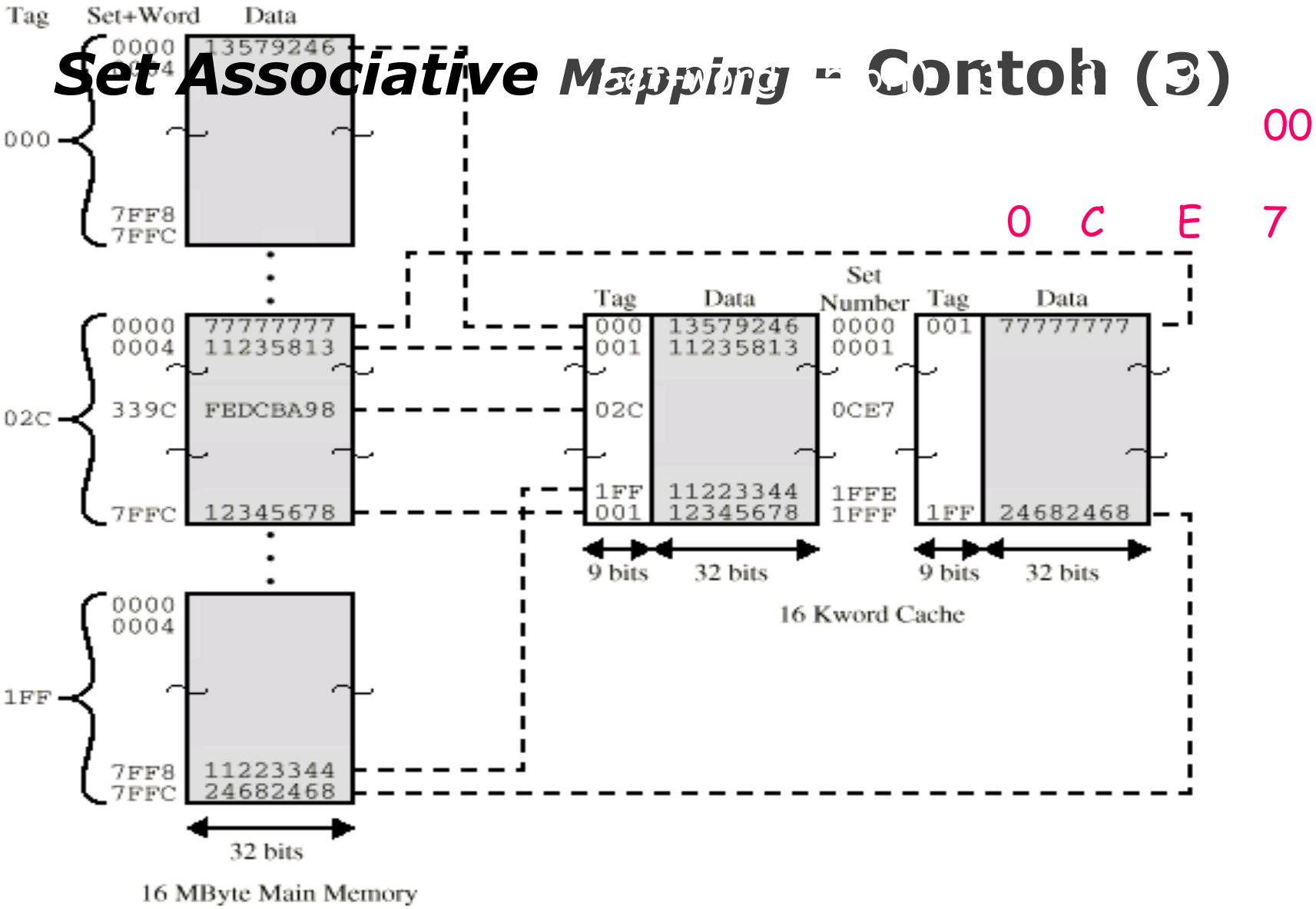
008000 + 007FFF = 00FFFF = **0000 0000 1**111 1111 1111 1111 + 1 → **0000 0001 0**000 0000 0000 0000 = 010000

Setiap alamat memori di atas bebas menempati salah satu line pada nomor set tersebut

Setiap satu nomor blok memori hanya dapat menempati satu nomor set

Satu nomor set dapat dapat ditempati oleh:

Set Associative Mapping - Contoh (3)



Set Associative Mapping - Kelebihan

(+) Setiap blok memori dapat menempati lebih dari satu kemungkinan nomor line (dapat menggunakan line yang kosong), sehingga *thrashing* dapat diperkecil

(+) Jumlah tag lebih sedikit (dibanding model *associative*), sehingga jalur untuk melakukan perbandingan tag lebih sederhana

Referensi

[STA10] Stalling, William. 2016. "*Computer Organization and Architecture: Designing for Performance*". 10th edition