BAB

4

Memori

Memori adalah bagian dari komputer tempat program – program dan data – data disimpan. Bebarapa pakar komputer (terutama dari Inggris) menggunakan istilah *store* atau *storage* untuk memori, meskipun kata *storage* sering digunakan untuk menunjuk ke penyimpanan disket. Tanpa sebuah memori sebagai tempat untuk mendapatkan informasi guna dibaca dan ditulis oleh prosesor maka tidak akan ada komputer – komputer digital dengan sistem penyimpanan program.

Walaupun konsepnya sederhana, memori komputer memiliki aneka ragam jenis, teknologi, organisasi, unjuk kerja dan harganya. Dalam bab ini akan dibahas mengenai memori internal dan bab selanjutnya membahas memori eksternal. Perlu dijelaskan sebelumnya perbedaan keduanya yang sebenarnya fungsinya sama untuk penyimpanan program maupun data. *Memori internal* adalah memori yang dapat diakses langsung oleh prosesor. Sebenarnya terdapat beberapa macam memori internal, yaitu register yang terdapat di dalam prosesor, cache memori dan memori utama berada di luar prosesor. Sedangkan *memori eksternal* adalah memori yang diakses prosesor melalui piranti I/O, seperti disket dan hardisk.

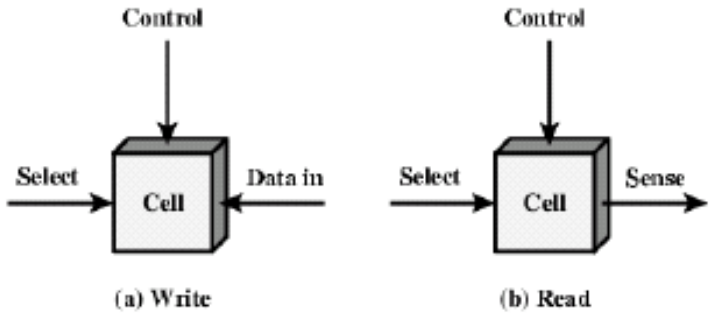
4.1 Operasi Sel Memori

Elemen dasar memori adalah sel memori. Walaupun digunakan digunakan sejumlah teknologi elektronik, seluruh sel memori memiliki sifat – sifat tertentu :

- Sel memori memiliki dua keadaan stabil (atau semi-stabil), yang dapat digunakan untuk merepresentasikan bilangan biner 1 atau 0.
- Sel memori mempunyai kemampuan untuk ditulisi (sedikitnya satu kali).
- Sel memori mempunyai kemampuan untuk dibaca.

Gambar 4.1 menjelaskan operasi sel memori. Umumnya sel memori mempunyai tiga terminal fungsi yang mampu membawa sinyal listrik. Terminal *select* berfungsi memilih operasi tulis atau baca. Untuk penulisan, terminal lainnya menyediakan sinyal listrik yang men-set

keadaan sel brnilai 1 atau 0, sedangkan untuk operasi pembacaan, terminal ini digunakan sebagai keluaran.



Gambar 4.1 Operasi sel memori

4.2 Karakteristik Sistem Memori

Untuk mempelajari sistem memori secara keseluruhan, harus mengetahui karakteristik – karakteristik kuncinya. Karakteristik penting sistem memori disajikan dalam tabel 4.1 berikut :

Tabel 4.1 Karakteristik penting sistem memori komputer

Karakteristik	Macam/ Keterangan
Lokasi	1. CPU 2. Internal (main) 3. External (secondary)
Kapasitas	1. Ukuran word 2. Jumlah word
Satuan transfer	1. Word 2. Block
Metode akses	1. Sequential access 2. Direct access 3. Random access 4. Associative access
Kinerja	1. Access time 2. Cycle time 3. Transfer rate
Tipe fisik	1. Semikonduktor 2. Magnetik
Karakteristik fisik	1. Volatile/nonvolatile 2. Erasable/nonerasable

Dilihat dari *lokasi*, memori dibedakan menjadi beberapa jenis, yaitu register, memori internal dan memori eksternal. Register berada di dalam chip prosesor, memori ini diakses

langsung oleh prosesor dalam menjalankan operasinya. Register digunakan sebagai memori sementara dalam perhitungan maupun pengolahan data dalam prosesor. Memori internal adalah memori yang berada diluar chip prosesor namun mengaksesannya langsung oleh prosesor. Memori internal dibedakan menjadi memori utama dan cache memori. Memori eksternal dapat diakses oleh prosesor melalui piranti I/O, memori ini dapat berupa disk maupun pita.

Karakteristik lainnya adalah *kapasitas*. Kapasitas memori internal maupun eksternal biasanya dinyatakan dalam mentuk *byte* (1 byte = 8 bit) atau *word*. Panjang word umumnya 8, 16, 32 bit. Memori eksternal biasanya lebih besar kapasitasnya daripada memori internal, hal ini disebabkan karena teknologi dan sifat penggunaannya yang berbeda.

Karakteristik berikutnya adalah *satuan tranfer*. Bagi memori internal, satuan tranfer sama dengan jumlah saluran data yang masuk ke dan keluar dari modul memori. Jumlah saluran ini sering kali sama dengan panjang word, tapi dimungkinkan juga tidak sama. Tiga konsep yang berhubungan dengan satuan transfer :

- *Word*, merupakan satuan "alami" organisasi memori. Ukuran word biasanya sama dengan jumlah bit yang digunakan untuk representasi bilangan dan panjang instruksi.
- *Addressable units,* pada sejumlah sistem, adressable units adalah word. Namun terdapat sistem dengan pengalamatan pada tingkatan byte. Pada semua kasus hubungan antara panjang A suatu alamat dan jumlah N adressable unit adalah 2A =N.
- *Unit of tranfer,* adalah jumlah bit yang dibaca atau dituliskan ke dalam memori pada suatu saat. Pada memori eksternal, tranfer data biasanya lebih besar dari suatu word, yang disebut dengan *block*.

Perbedaan tajam yang terdapat pada sejumlah jenis memori adalah *metode access*-nya. Terdapat empat macam metode :

- *Sequential access*, memori diorganisasi menjadi unit – unit data yang disebut *record*. Akses harus dibuat dalam bentuk urutan linier yang spesifik. Informasi mengalamatan yang disimpan dipakai untuk memisahkan record – record dan untuk membantu proses pencarian. Terdapat *shared read/write mechanism* untuk penulisan/pembacaan memorinya. Pita magnetik merupakan memori yang menggunakan metode sequential access.
- *Direct access,* sama sequential access terdapat *shared read/write mechanism*. Setiap blok dan record memiliki alamat unik berdasarkan lokasi fisiknya. Akses dilakukan langsung pada alamat memori. Disk adalah memori direct access.
- *Random access,* setiap lokasi memori dipilih secara random dan diakses serta dialamati secara langsung. Contohnya adalah memori utama.

- *Associative access,* merupakan jenis random akses yang memungkinkan pembandingan lokasi bit yang diinginkan untuk pencocokan. Jadi data dicari berdasarkan isinya bukan alamatnya dalam memori. Contoh memori ini adalah cache memori yang akan dibahas di akhir bab ini.

Berdasarkan karakteristik unjuk kerja, memiliki tiga parameter utama pengukuran unjuk kerja, yaitu :

- *Access time*, bagi random access memory, waktu akses adalah waktu yang dibutuhkan untuk melakukan operasi baca atau tulis. Sedangkan untuk memori non-random akses merupakan waktu yang dibutuhkan dalam melakukan mekanisme baca atau tulis pada lokasi tertentu.
- *Memory cycle time,* konsep ini digunakan pada random access memory dan terdiri dari access time ditambah dengan waktu yang diperlukan transient agar hilang pada saluran sinyal.
- *Transfer rate,* adalah kecepatan data transfer ke unit memori atau dari unit memori. Pada random access memory sama dengan 1/(cycle time). Sedangkan untuk non-random access memory dengan perumusan :

$$T_N = T_A + \frac{N}{R}$$

TN = waktu rata – rata untuk membaca atau menulis N bit

TA = waktu akses rata – rata

N = jumlah bit

R = kecepatan transfer dalam bit per detik (bps)

Jenis *tipe fisik* memori yang digunakan saat ini adalah memori semikonduktor dengan teknologi VLSI dan memori permukaan magnetik seperti yang digunakan pada disk dan pita magnetik.

Berdasarkan *karakteristik fisik*, media penyimpanan dibedakan menjadi *volatile* dan *non-volatile*, serta *erasable* dan *nonerasable*. Pada *volatile memory*, informasi akan hilang apabila daya listriknya dimatikan, sedangkan *non-volatile memory* tidak hilang walau daya listriknya hilang. Memori permukaan magnetik adalah contoh *no-nvolatile memory*, sedangkan semikonduktor ada yang *volatile* dan *non-volatile*. Ada jenis memori semikonduktor yang tidak bisa dihapus kecuali dengan menghancurkan unit *storage*-nya, memori ini dikenal dengan ROM (*Read Only Memory*).

4.3 Keandalan Memori

Untuk memperoleh keandalan sistem ada tiga pertanyaan yang diajukan: Berapa banyak ? Berapa cepat? Berapa mahal?

Pertanyaan berapa banyak adalah sesuatu yang sulit dijawab, karena berapapun kapasitas memori tentu aplikasi akan menggunakannya. Jawaban pertanyaan berapa cepat adalah memori harus mempu mengikuti kecepatan CPU sehingga terjadi sinkronisasi kerja antar CPU dan memori tanpa adanya waktu tunggu karena komponen lain belum selesai prosesnya. Mengenai harga, sangatlah relatif. Bagi produsen selalu mencari harga produksi paling murah tanpa mengorbankan kualitasnya untuk memiliki daya saing di pasaran.

Hubungan harga, kapasitas dan waktu akses adalah :

- Semakin kecil waktu akses, semakin besar harga per bitnya.
- Semakin besar kapasitas, semakin kecil harga per bitnya.
- Semakin besar kapasitas, semakin besar waktu aksesnya.

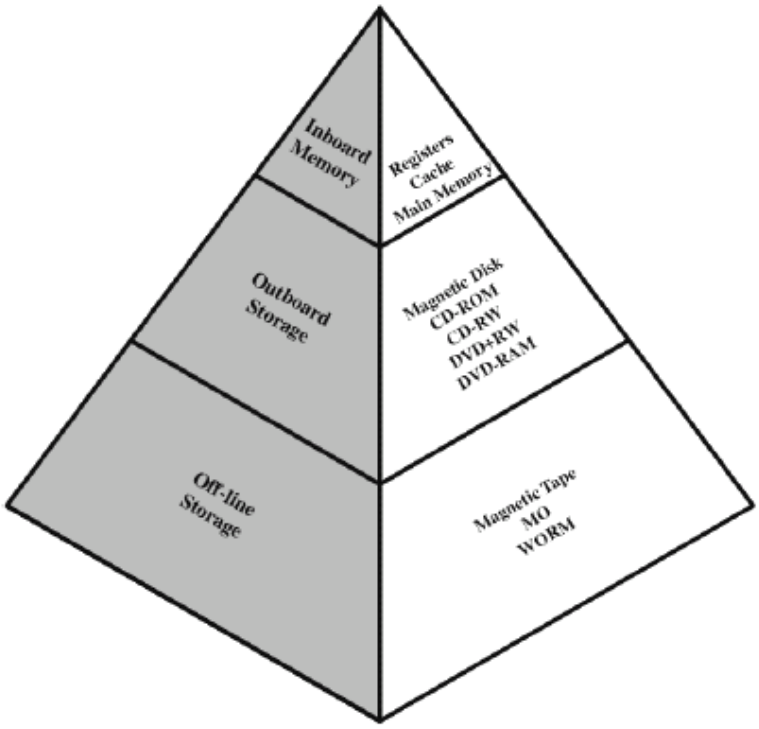
Dilema yang dihadapi para perancang adalah keinginan menerapkan teknologi untuk kapasitas memori yang besar karena harga per bit yang murah namun hal itu dibatasi oleh teknologi dalam memperoleh waktu akses yang cepat. Salah satu pengorganisasian masalah ini adalah menggunakan *hirarki memori*. Seperti terlihat pada gambar 4.2, bahwa semakin menurunnya hirarki maka hal berikut akan terjadi :

- Penurunan harga/bit
- Peningkatan kapasitas
- Peningkatan waktu akses
- Penurunan frekuensi akses memori oleh CPU.

Kunci keberhasilan hirarki ini pada penurunan frekuensi aksesnya. Semakin lambat memori maka keperluan CPU untuk mengaksesnya semakin sedikit. Secara keseluruhan sistem komputer akan tetap cepat namun kebutuhan kapasitas memori besar terpenuhi.

Tabel 4.2 Tabel spesifikasi memori

Tipe memori	Teknologi	Ukuran	Waktu akses
Cache Memory	semikonduktor RAM	128 – 512 KB	10 ns
Memori Utama	semikonduktor RAM	4 – 128 MB	50 ns
Disk magnetik	Hard Disk	Gigabyte	10 ms, 10MB/det
Disk Optik	CD-ROM	Gigabyte	300ms, 600KB/det
Pita magnetik	Tape	100 MB	Det -mnt, 10MB/mnt



Gambar 4.2 Hirarki memori

4.4 Satuan Memori

Satuan pokok memori adalah digit biner, yang disebut *bit*. Suatu bit dapat berisi sebuah angka 0 atau 1. Ini adalah satuan yang paling sederhana. Memori juga dinyatakan dalam byte (1 byte = 8 bit). Kumpulan byte dinyatakan dalam *word*. Panjang word yang umum adalah 8, 16, dan 32 bit.

Tabel 4.3 Tingkatan satuan memori

Symbol		Number of bytes	
Kilobytes	Kb	2e10	1024
Megabyte	Mb	2e20	1,048,576
Gigabyte	Gb	2e30	1,073,741,824
Terabyte	Tb	2e40	1,099,511,627,776

4.5 Memori Utama Semikonduktor

Pada komputer lama, bentuk umum random access memory untuk memori utama adalah sebuah piringan ferromagnetik berlubang yang dikenal sebagai *core*, istilah yang tetap dipertahankan hingga saat ini.

4.5.1 Jenis Memori Random Akses

Semua jenis memori yang dibahas pada bagian ini adalah berjenis random akses, yaitu data secara langsung diakses melalui logik pengalamatan *wired-in*. Tabel 4.4 adalah daftar jenis memori semikonduktor utama.

Hal yang membedakan karakteristik RAM (*Random Access Memory*) adalah dimungkinkannya pembacaan dan penulisan data ke memori secara cepat dan mudah. Aspek lain adalah RAM bersifat *volatile*, sehingga RAM hanya menyimpan data sementara. Teknologi yang berkembang saat ini adalah statik dan dinamik. *RAM dinamik* disusun oleh sel – sel yang menyimpan data sebagai muatan listrik pada kapasitor. Karena kapasitor memiliki kecenderungan alami untuk mengosongkan muatan, maka RAM dinamik memerlukan pengisian muatan listrik secara periodik untuk memelihara penyimpanan data. Pada *RAM statik*, nilai biner disimpan dengan menggunakan konfigurasi gate logika flipflop tradisional. RAM statik akan menyimpan data selama ada daya listriknya.

RAM statik maupun dinamik adalah *volatile*, tetapi RAM dinamik lebih sederhana dan rapat sehingga lebih murah. RAM dinamik lebih cocok untuk kapasitas memori besar, namun RAM statik umumnya lebih cepat.

Read only memory (ROM) sangat berbeda dengan RAM, seperti namanya, ROM berisi pola data permanen yang tidak dapat diubah. Data yang tidak bisa diubah menimbulkan keuntungan dan juga kerugian. Keuntungannya untuk data yang permanen dan sering digunakan pada sistem operasi maupun sistem perangkat keras akan aman diletakkan dalam ROM. Kerugiaannya apabila ada kesalahan data atau adanya perubahan data sehingga perlu penyisipan – penyisipan.

Kerugian tersebut bisa diantisipasi dengan jenis *programmable ROM*, disingkat PROM. ROM dan PROM bersifat *non-volatile*. Proses penulisan PROm secara elektris dengan peralatan khusus.

Variasi ROM lainnya adalah *read mostly memory*, yang sangat berguna untuk aplikasi operasi pembacaan jauh lebih sering daripada operasi penulisan. Terdapat tiga macam jenis, yaitu: EPROM, EEPROM dan *flash memory*.

EEPROM (*electrically erasable programmable read only memory*) merupakan memori yang dapat ditulisi kapan saja tanpa menghapus isi sebelumnya. EEPROM menggabungkan kelebihan *non-volatile* dengan fleksibilitas dapat di-update.

Bentuk memori semikonduktor terbaru adalah *flash memory*. Memori ini dikenalkan tahun 1980-an dengan keunggulan pada kecepatan penulisan programnya. *Flash memory* menggunakan teknologi penghapusan dan penulisan elektrik. Seperti halnya EPROM, *flash*

memory hanya membutuhkan sebuah transistor per byte sehingga dapat diperoleh kepadatan tinggi.

Tabel 4.4 Tipe – tipe memori semikonduktor

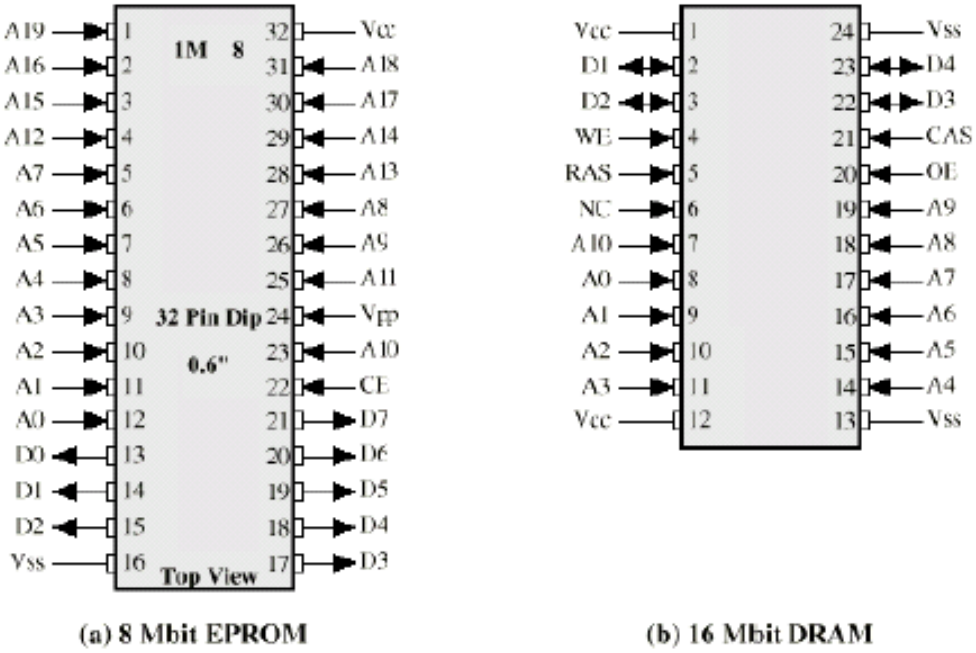
<table>
<tr><th>Tipe memori</th><th>Katagori</th><th>Penghapusan</th><th>Penulisan</th><th>Volatilitas</th></tr>
<tr><td>RAM</td><td>Read – write
Read only</td><td>Electrically
byte level</td><td>Electrically</td><td>Volatile</td></tr>
<tr><td>ROM</td><td rowspan="2">Read only</td><td rowspan="2">Tidak bisa
dihapus</td><td>Mask</td><td rowspan="5">Non-volatile</td></tr>
<tr><td>PROM</td><td rowspan="4">Electrically</td></tr>
<tr><td>EPROM</td><td rowspan="3">Read mostly</td><td>UV light
chip level</td></tr>
<tr><td>Flash Memory</td><td>Electrically
block level</td></tr>
<tr><td>EEPROM</td><td>Electrically
Byte level</td></tr>
</table>

4.5.2 Pengemasan (Packging)

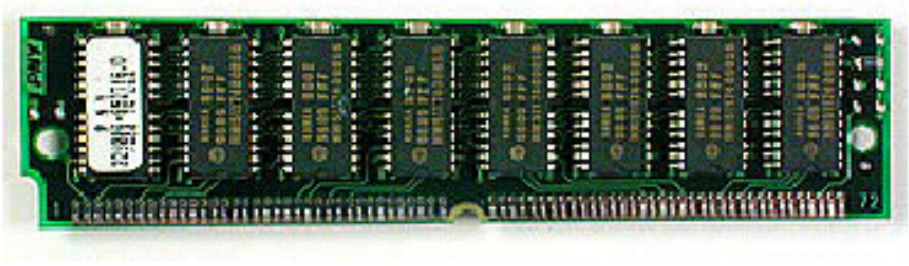
Gambar 4.3a menunjukkan sebuah contoh kemasan EPROM, yang merupakan keping 8 Mbit yang diorganisasi sebagai 1Mx8. Dalam kasus ini, organisasi dianggap sebagai kemasan satu word per keping. Kemasan terdiri dari 32 pin, yang merupakan salah satu ukuran kemasan keping standar. Pin – pin tersebut mendukung saluran – saluran sinyal beikut ini :

- Alamat word yang sedang diakses. Untuk 1M word, diperlukan sejumlah 20 buah (220 = 1M).
- Data yang akan dibaca, terdiri dari 8 saluran (D0 –D7)
- Catu daya keping adalah Vcc
- Pin grounding Vss
- Pin chip enable (CE). Karena mungkin terdapat lebih dari satu keping memori yang terhubung pada bus yang sama maka pin CE digunakan untuk mengindikasikan valid atau tidaknya pin ini. Pin CE diaktifkan oleh logik yang terhubung dengan bit berorde tinggi bus alamat (diatas A19)
- Tegangan program (Vpp).

Konfigurasi pin DRAM yang umum ditunjukkan gambar 4.3b, untuk keping 16 Mbit yang diorganisasikan sebagai 4M x 4. Terdapat sejumlah perbedaan dengan keping ROM, karena ada operasi tulis maka pin – pin data merupakan input/output yang dikendalikan oleh WE (*write enable*) dan OE (*output enable*).



Gambar 4.3 Pin dan sinyal kemasan memori



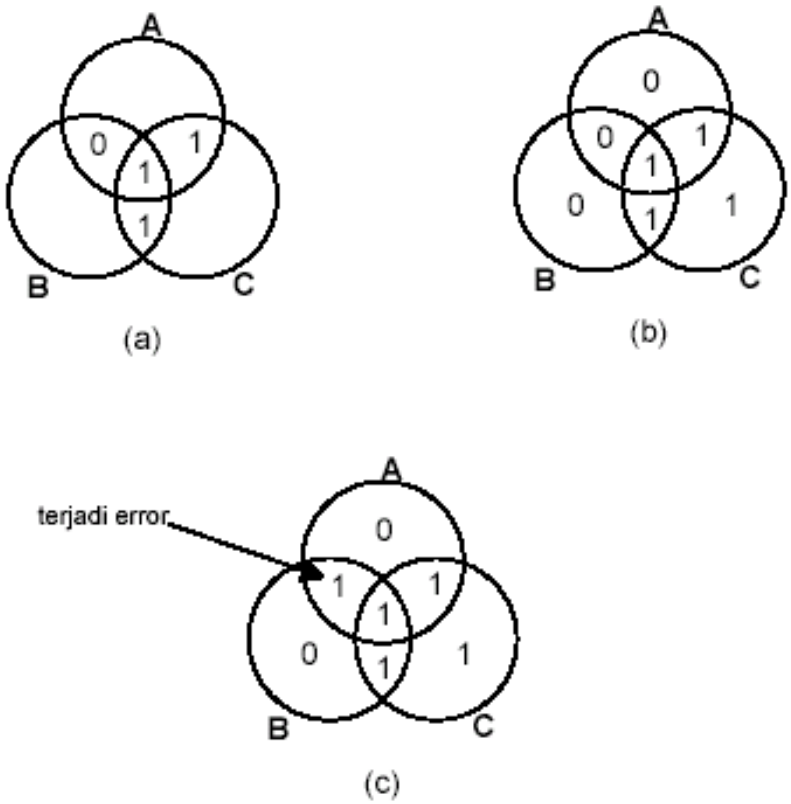
Gambar 4.4 Packging SIMM

4.5.3 Koreksi Error

Dalam melaksanakan fungsi penyimpanan, memori semikonduktor dimungkinkan mengalami kesalahan. Baik kesalahan berat yang biasanya merupakan kerusakan fisik memori maupun kesalahan ringan yang berhubungan data yang disimpan. Kesalahan ringan dapat dikoreksi kembali. Untuk mengadakan koreksi kesalahan data yang disimpan diperlukan dua mekanisme, yaitu *mekanisme pendeteksian kesalahan* dan *mekanisme perbaikan kesalahan*.

Mekanisme pendeteksian kesalahan dengan menambahkan data word (D) dengan suatu kode, biasanya bit cek paritas (C). Sehingga data yang disimpan memiliki panjang D + C. Kesalahan akan diketahui dengan menganalisa data dan bit paritas tersebut. Mekanisme perbaikan

kesalahan yang paling sederhana adalah *kode Hamming*. Metode ini diciptakan Richard Hamming di Bell Lab pada tahun 1950.



Gambar 4.5 Koreksi kesalahan dengan kode Hamming

Perhatikan gambar 4.5, disajikan tiga lingkaran Venn (A, B, C) saling berpotongan sehingga terdapat 7 ruang. Metode diatas adalah koreksi kesalahan untuk word data 4 bit (D =4). Gambar 4.5a adalah data aslinya. Kemudian setiap lingkaran harus diset bit logika 1 berjumlah genap sehingga harus ditambah bit – bit paritas pada ruang yang kosong seperti gambar 4.5b. Apabila ada kesalahan penulisan bit pada data seperti gambar 4.5c akan dapat diketahui karena lingkaran A dan B memiliki logika 1 berjumlah ganjil.

Lalu bagaimana dengan word lebih dari 4 bit ? Ada cara yang mudah yang akan diterangkan berikut. Sebelumnya perlu diketahui jumlah bit paritas yang harus ditambahkan untuk sejumlah bit word. Contoh sebelumnya adalah koreksi kesalahan untuk kesalahan tunggal yang sering disebut *single error correcting* (SEC). Jumlah bit paritas yang harus ditambahkan lain pada *double error correcting* (DEC). Tabel 4.5 menyajikan jumlah bit paritas yang harus ditambahkan dalam sistem kode Hamming.

Tabel 4.5 Penambahan bit cek paritas untuk koreksi kode Hamming

# Data Bits	# Bit Paritas SEC	# Bit Paritas DEC
8	4	5
16	5	6
32	6	7
64	7	8
128	8	9
512	9	10

Contoh koreksi kode Hamming 8 bit data :

Dari tabel 4.5 untuk 8 bit data diperlukan 4 bit tambahan sehingga panjang seluruhnya adalah 12 bit. Layout bit disajikan dibawah ini :

Posisi Bit	Posisi Angka				Cek Bit	Data Bit
12	1	1	0	0		D8
11	1	0	1	1		D7
10	1	0	1	0		D6
9	1	0	0	1		D5
8	1	0	0	0	C4	
7	0	1	1	1		D4
6	0	1	1	0		D3
5	0	1	0	1		D2
4	0	1	0	0	C3	
3	0	0	1	1		D1
2	0	0	1	0	C2	
1	0	0	0	1	C1	

Bit cek paritas ditempatkan dengan perumusan 2N dimana N = 0,1,2, ……, sedangkan bit data adalah sisanya. Kemudian dengan exclusive-OR dijumlahkan ebagai berikut :

$$C1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7$$

$$C2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$$

$$C4 = D2 \oplus D3 \oplus D4 \oplus D8$$

$$C8 = D5 \oplus D6 \oplus D7 \oplus D8$$

Setiap cek bit (C) beroperasi pada setiap posisi bit data yang nomor posisinya berisi bilangan 1 pada kolomnya.

Sekarang ambil contoh suatu data, misalnya masukkan data : 00111001 kemudian ganti bit data ke 3 dari 0 menjadi 1 sebagai error-nya. Bagaimanakah cara mendapatkan bit data ke 3 sebagai bit yang terdapat error?

Jawab :

Masukkan data pada perumusan cek bit paritas :

$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$

$C2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$

$C4 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$

$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$

Sekarang bit 3 mengalami kesalahan sehingga data menjadi: 00111**1**01

$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$

$C2 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$

$C4 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$

$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$

Apabila bit – bit cek dibandingkan antara yang lama dan baru maka terbentuk *syndrom word* :

	C8	C4	C2	C1	
	0	1	1	1	
⊕	0	0	0	1	
	0	1	1	0	= 6

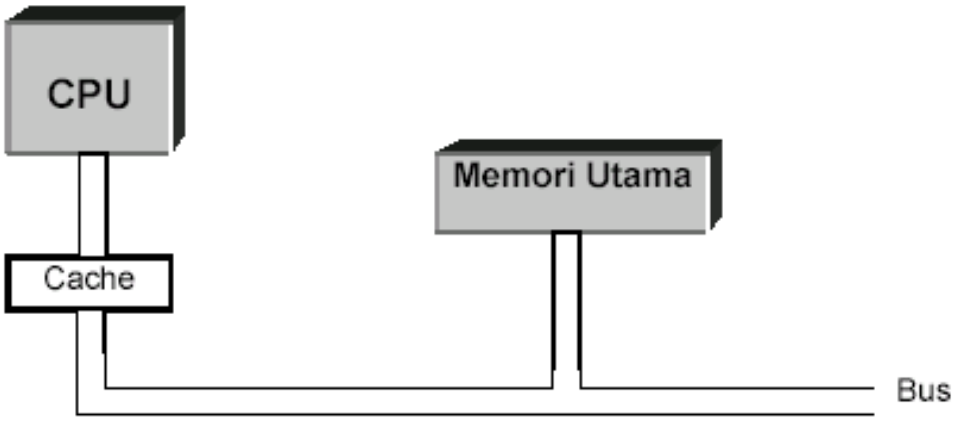
Sekarang kita lihat posisi bit ke-6 adalah data ke-3.

Mekanisme koreksi kesalahan akan meningkatkan realibitas bagi memori tetapi resikonya adalah menambah kompleksitas pengolahan data. Disamping itu mekanisme koreksi kesalahan akan menambah kapasitas memori karena adanya penambahan bit – bit cek paritas. Jadi ukuran memori akan lebih besar beberapa persen atau dengan kata lain kapasitas penyimpanan akan berkurang karena beberapa lokasi digunakan untuk mekanisme koreksi kesalahan.

4.6 Cache Memori

Cache memori difungsikan mempercepat kerja memori sehingga mendekati kecepatan prosesor. Konsepnya dijelaskan pada gambar 4.6 dan gambar 4.7. Dalam organisasi komputer, memori utama lebih besar kapasitasnya namun lambat operasinya, sedangkan cache memori berukuran kecil namun lebih cepat. Cache memori berisi salinan memori utama.

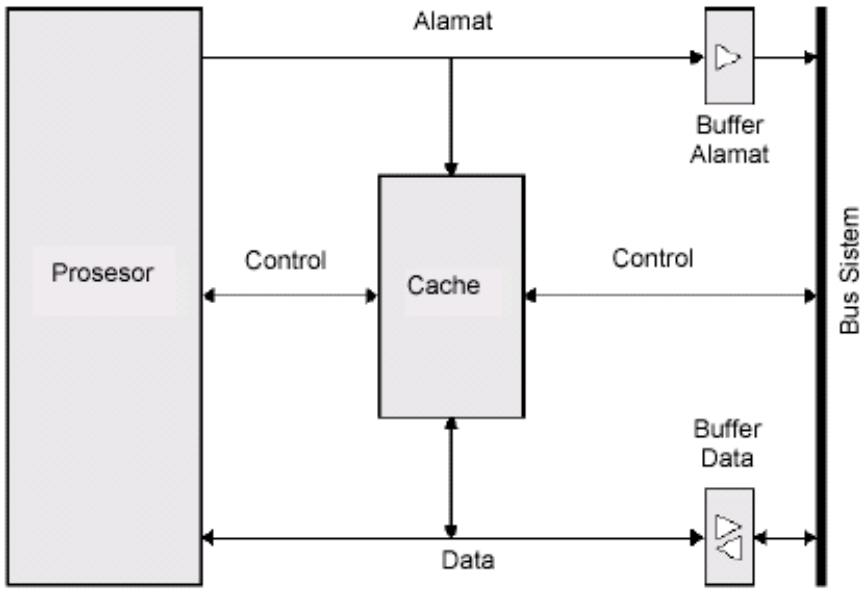
Pada saat CPU membaca sebuah word memori, maka dilakukan pemeriksaan untuk mengetahui apakah word tersebut berada dalam cache memori. Bila ada dalam cache memori maka dilakukan pengiriman ke CPU, bila tidak dijumpai maka dicari dalam memori utama, selanjutnya blok yang berisi sejumlah word tersebut dikirim ke cache memori dan word yang diminta CPU dikirimkan ke CPU dari cache memori. Karena fenomena lokalitas referensi, ketika blok data diberikan ke dalam cache memori, terdapat kemungkinan bahwa word-word berikutnya yang berada dalam satu blok akan diakses oleh CPU. Konsep ini yang menjadikan kinerja memori lebih baik.



Gambar 4.6 Hubungan cache memori

Sehingga dapat disimpulkan bahwa kerja cache adalah antisipasi terhadap permintaan data memori yang akan digunakan CPU. Apabila data diambil langsung dari memori utama bahkan memori eksternal akan memakan waktu lama yang menyebabkan status tunggu pada prosesor.

Ukuran cache memori adalah kecil, semakin besar kapasitasnya maka akan memperlambat proses operasi cache memori itu sendiri, disamping harga cache memori yang sangat mahal.



Gambar 4.7 Organisasi cache memori

4.7 Elemen Rancangan

Walaupun terdapat banyak implementasi cache, namun dari sisi organisasi maupun arsitekturnya tidak banyak macamnya.

Tabel 4.6 Unsur – unsur rancangan cache memori

Unsur	Macam
Kapasitas	-
Ukuran blok	-
Mapping	*1. Direct Mapping* *2. Assosiative Mapping* *3. Set Assosiative Mapping*
Algoritma pengganti	1. *Least recently used* (LRU) 2. *First in first out* (FIFO) 3. *Least frequently used* (LFU) *4. Random*
Write Policy	1. Write Througth 2. Write Back 3. Write Once
Jumlah Cache	1. Singe atau dua level 2. Unified atau split

4.7.1 Kapasitas Cache

Menentukan ukuran memori cache sangatlah penting untuk mendongkrak kinerja komputer. Dari segi harga cache sangatlah mahal tidak seperti memori utama. Semakin besar kapasitas cache tidak berarti semakin cepat prosesnya, dengan ukuran besar akan terlalu banya gate pengalamatannya sehingga akan memperlambat proses.

Kita bisa melihat beberapa merek prosesor di pasaran beberapa waktu lalu. AMD mengeluarkan prosesor K5 dan K6 dengan cache yang besar (1MB) tetapi kinerjanya tidak bagus. Kemudian Intel pernah mengeluarkan prosesor tanpa cache untuk alasan harga yang murah, yaitu seri Intel Celeron pada tahun 1998-an hasil kinerjanya sangat buruk terutama untuk operasi data besar, floating point, 3D. Intel Celeron versi berikutnya sudah ditambah cache sekitar 128KB.

Lalu berapa idealnya kapasitas cache? Sejumlah penelitian telah menganjurkan bahwa ukuran cache antara 1KB dan 512KB akan lebih optimum [STA96].

4.7.2 Ukuran Blok

Elemen rancangan yang harus diperhatikan lagi adalah ukuran blok. Telah dijelaskan adanya sifat lokalitas referensi maka nilai ukuran blok sangatlah penting. Apabila blok berukuran besar ditransfer ke cache akan menyebabkan hit ratio mengalami penurunan karena banyaknya data yang dikirim disekitar referensi. Tetapi apabila terlalu kecil, dimungkinkan memori yang akan dibutuhkan CPU tidak tercakup. Apabila blok berukuran besar ditransfer ke cache, maka akan terjadi :

1. Blok – blok yang berukuran lebih besar mengurangi jumlah blok yang menempati cache. Karena isi cache sebelumnya akan ditindih.
2. Dengan meningkatnya ukuran blok maka jarak setiap word tambahan menjadi lebih jauh dari word yang diminta, sehingga menjadi lebih kecil kemungkinannya digunakan cepat.

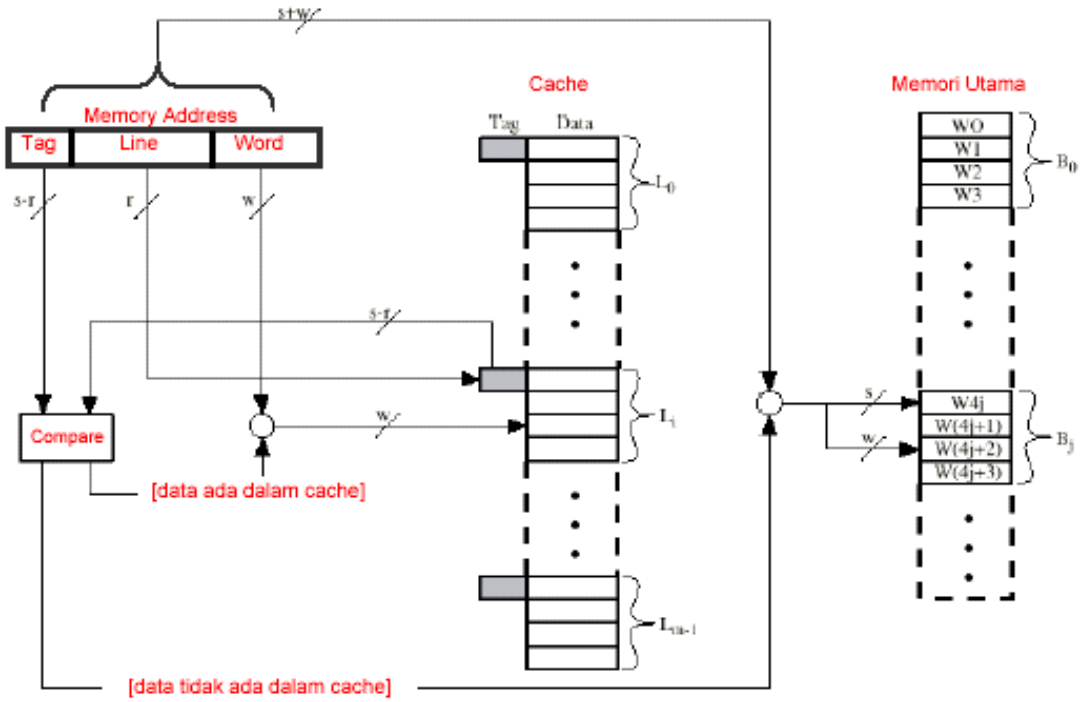
Hubungan antara ukuran blok dan hit ratio sangat rumit untuk dirumuskan, tergantung pada karakteristik lokalitas programnya dan tidak terdapat nilai optimum yang pasti telah ditemukan. Ukuran antara 4 hingga 8 satuan yang dapat dialamati (word atau byte) cukup beralasan untuk mendekati nilai optimum [STA96].

4.7.3 Fungsi Pemetaan (Mapping)

Telah kita ketahui bahwa cache mempunyai kapasitas yang kecil dibandingkan memori utama. Sehingga diperlukan aturan blok – blok mana yang diletakkan dalam cache. Terdapat tiga metode, yaitu pemetaan langsung, pemetaan asosiatif, dan pemetaan asosiatif set.

Pemetaan Langsung

Pemetaan langsung adalah teknik yang paling sederhana, yaitu teknik ini memetakan blok memori utama hanya ke sebuah saluran cache saja. Gambar 4.8 menjelaskan mekanisme pemetaan langsung.



Gambar 4.8 Organisasi cache pemetaan langsung

i = j modulus m dan **m = 2r**

dimana :

i = nomer saluran cache

j = nomer blok memori utama

m = jumlah saluran yang terdapat dalam cache

Fungsi pemetaan diimplementasikan dengan menggunakan alamat, yang terdiri dari tiga field (tag, line, word), lihat gambar 4.8.

w = word, adalah bit paling kurang berarti yang mengidentifikasikan word atau byte unik dalam blok memori utama.

s = byte sisa word yang menspesifikasi salah satu dari 2S blok memori utama. Cache logik menginterpretasikan bit – bit S sebagai suatu tag s – r bit (bagian paling berarti dalam alamat) dan field saluran r bit.

Efek pemetaan tersebut adalah blok – blok memori utama diberikan ke saluran cache seperti berikut ini:

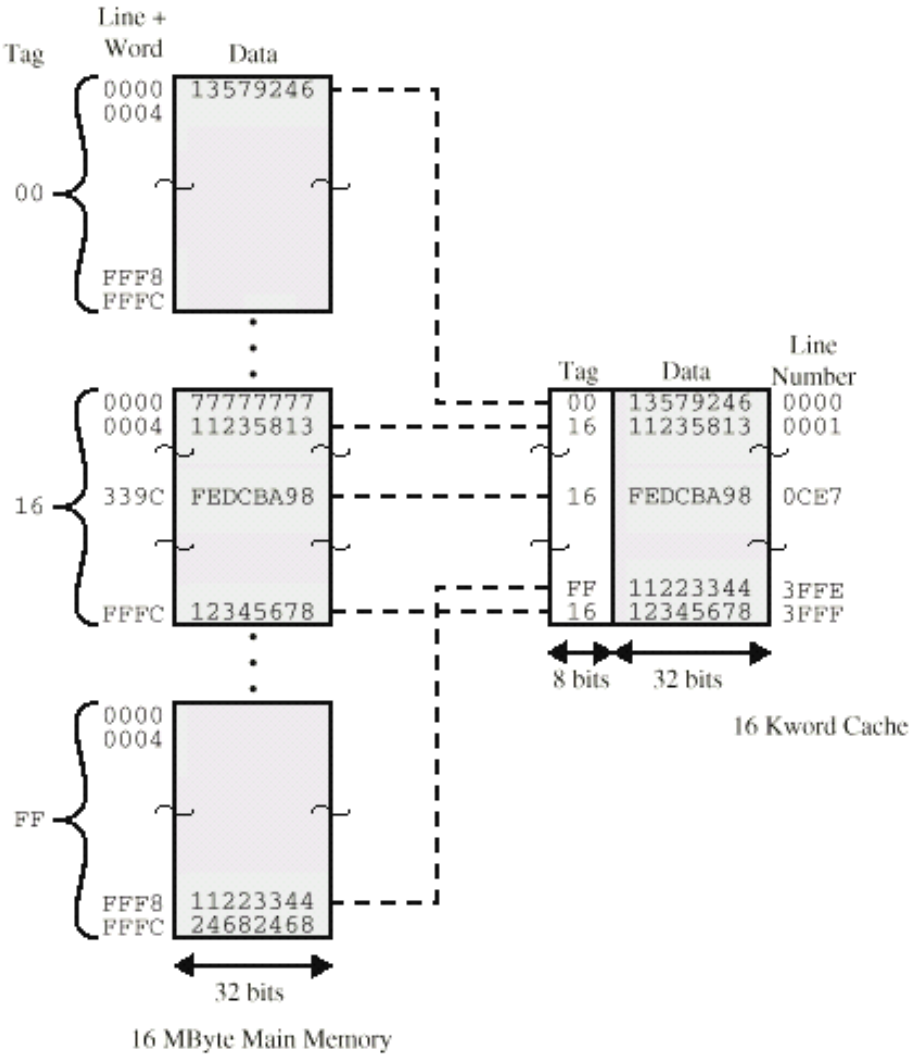
Saluran cache	Blok – blok memori utama
0	0, m, ……., 2^s – m
1	1, (m+1), ……., 2^s – (m+1)
.	.
.	.
.	.
m - 1	(m-1), (2m-1), ……., 2^s – 1

Jadi dalam metode ini pemetaan adalah bagian alamat blok memori utama sebagai nomer saluran cache. Ketika suatu blok data sedang diakses atau dibaca terhadap saluran yang diberikan, maka perlu memberikan tag bagi data untuk membedakannya dengan blok – blok lain yang dapat sesuai dengan saluran tersebut.

Pada gambar 4.9 disajikan contoh pemetaan langsung dengan m = 16K, maka pemetaannya :

Saluran cache	Blok – blok memori utama
0	000000, 010000, FF0000
1	000001, 010001, FF0001
.	.
.	.
.	.
3FFF	00FFFC, 01FFFC, FFFFFC

Perlu diketahui bahwa tidak ada dua buah blok yang dipetakan ke nomer saluran uang sama memiliki tag sama. Sehingga 000000, 010000, …., FF0000 masing – masing memiliki tag 00, 01, …., FF.



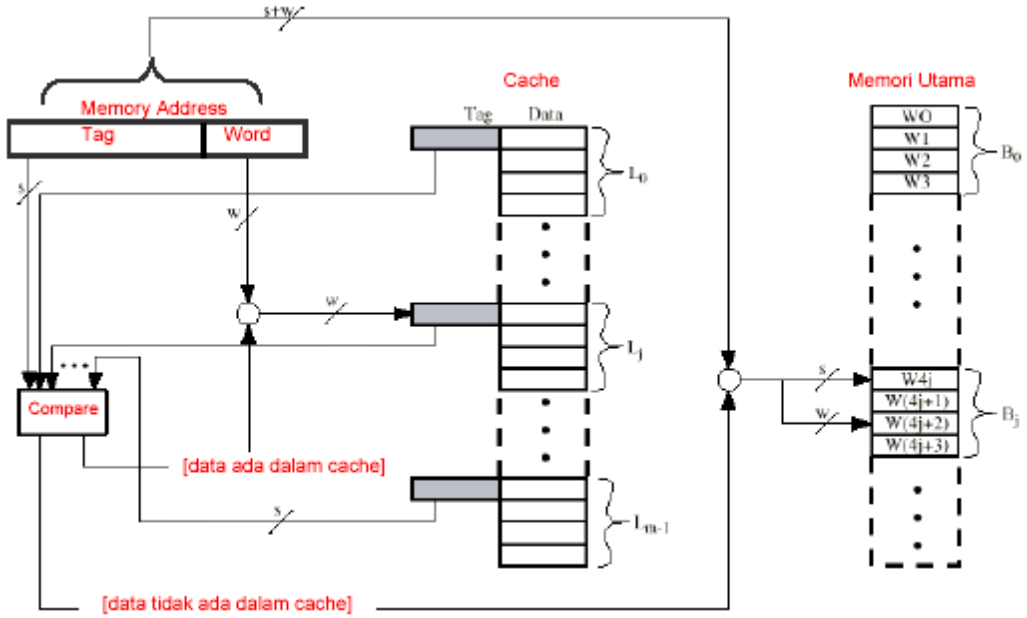
Gambar 4.9 Contoh pemetaan langsung

Teknik pemetaan ini sederhana dan mudah diimplementasikan, namun kelemahannya adalah terdapat lokasi cache yang tetap bagi sembarang blok – blok yang diketahui. Dengan demikian, apabila suatu program berulang – ulang melakukan word referensi dari dua blok yang berbeda memetakan saluran yang sama maka blok – blok itu secara terus – menerus akan di-swap ke dalam cache sehingga hit rasionya akan rendah.

Pemetaan Assosiatif

Pemetaan asosiatif mengatasi kekurangan pemetaan langsung dengan cara setiap blok memori utama dapat dimuat ke sembarang saluran cache. Alamat memori utama diinterpretasikan dalam field tag dan field word oleh kontrol logika cache. Tag secara unik mengidentifikasi sebuah blok memori utama.

Mekanisme untuk mengetahui suatu blok dalam cache dengan memeriksa setiap tag saluran cache oleh kontrol logika cache. Dengan pemetaan ini didapat fleksibilitas dalam penggantian blok baru yang ditempatkan dalam cache. Algoritma penggantian dirancang untuk memaksimalkan hit ratio, yang pada pemetaan langsung terdapat kelemahan dalam bagian ini. Kekurangan pemetaan asosiatif adalah kompleksitas rangkaian sehingga mahal secara ekonomi.



Gambar 4.10 Organisasi cache dengan pemetaan asosiatif

Pemetaan Assosiatif Set

Pemetaan asosiatif set menggabungkan kelebihan yang ada pada pemetaan langsung dan pemetaan asosiatif. Memori cache dibagi dalam bentuk set – set.

Pemetaan asosiatif set prinsipnya adalah penggabungan kedua pemetaan sebelumnya. Alamat memori utama diinterpretasikan dalam tiga field, yaitu: field tag, field set, dan field word. Hal ini mirip dalam pemetaan langsung. Setiap blok memori utama dapat dimuat dalam sembarang saluran cache. Gambar 4.11 menjelaskan organisasi pemetaan asosiatif set.

Dalam pemetaan asosiatif set, cache dibagi dalam *v* buah set, yang masing –masing terdiri dari k saluran. Hubungan yang terjadi adalah :

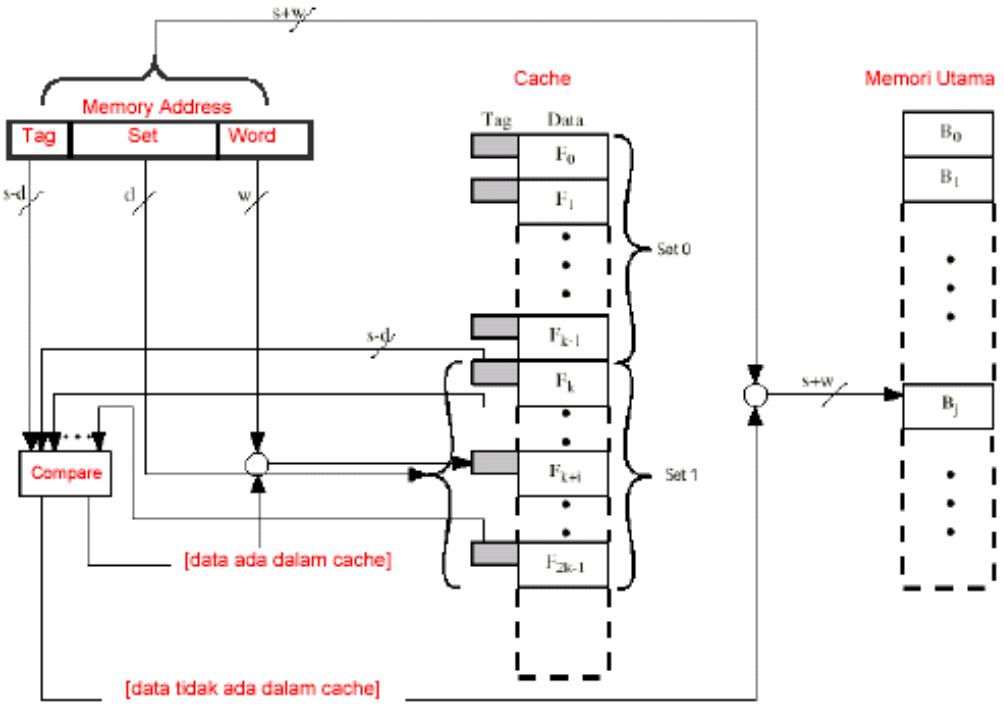
m = v x k

i = j modulus v dan v = 2d dimana :

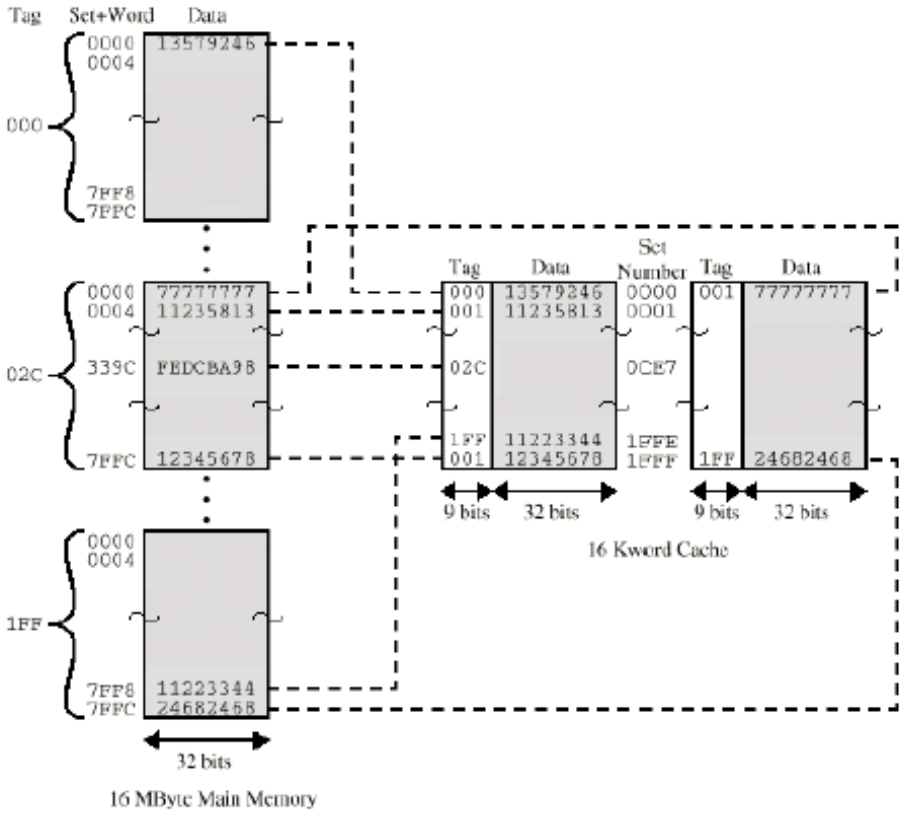
i = nomer set cache

j = nomer blok memori utama

m = jumlah saluran pada cache



Gambar 4.11 Organisasi cache dengan pemetaan asosiatif set



Gambar 4.12 Contoh pemetaan asosiatif set

Gambar 4.12 menjelaskan contoh yang menggunakan pemetaan asosiatif set dengan dua saluran pada masing-masing set, yang dikenal sebagai asosiatif set dua arah. Nomor set mengidentifikasi set unik dua saluran di dalam cache. Nomor set ini juga memberikan jumlah blok di dalam memori utama, modulus 2. Jumlah blok menentukan pemetaan blok terhadap saluran. Sehingga blok-blok 000000, 00A000,…,FF1000 pada memori utama dipetakan terhadap set 0 cache. Sembarang blok tersebut dapat dimuatkan ke salah satu dari kedua saluran di dalam set. Perlu dicatat bahwa tidak terdapat dua blok yang memetakannya terhadap set cache yang sama memiliki nomor tag yang sama. Untuk operasi *read*, nomor set dipakai untuk menentukan set dua saluran yang akan diuji. Kedua saluran di dalam set diuji untuk mendapatkan yang cocok dengan nomor tag alamat yang akan diakses.

Penggunaan dua saluran per set ($v = m/2, k = 2$), merupakan organisasi asosiatif set yang paling umum. Teknik ini sangat meningkatkan hit ratio dibandingkan dengan pemetaan langsung. Asosiatif set empat arah ($v = m/4, k = 4$) memberikan peningkatan tambahan yang layak dengan penambahan harga yang relatif rendah. Peningkatan lebih lanjut jumlah saluran per set hanya memiliki efek yang sedikit.

4.7.4 Algoritma Penggantian

Yang dimaksud Algoritma Penggantian adalah suatu mekanisme pergantian blok – blok dalam memori cache yang lama dengan data baru. Dalam pemetaan langsung tidak diperlukan algoritma ini, namun dalam pemetaan asosiatif dan asosiatif set, algoritma ini mempunyai peranan penting untuk meningkatkan kinerja cache memori.

Banyak algoritma penggantian yang telah dikembangkan, namun dalam buku ini akan dijelaskan algoritma yang umum digunakan saja. Algoritma yang paling efektif adalah *Least Recently Used* (LRU), yaitu mengganti blok data yang terlama berada dalam cache dan tidak memiliki referensi. Algoritma lainnya adalah *First In First Out* (FIFO), yaitu mengganti blok data yang awal masuk. Kemudian *Least Frequently Used* (LFU) adalah mengganti blok data yang mempunyai referensi paling sedikit. Teknik lain adalah algoritma *Random*, yaitu penggantian tidak berdasakan pemakaian datanya, melainkan berdasar slot dari beberapa slot kandidat secara acak.

4.7.5 Write Policy

Apabila suatu data telah diletakkan pada cache maka sebelum ada penggantian harus dicek apakah data tersebut telah mengalami perubahan. Apabila telah berubah maka data pada memori utama harus di-update. Masalah penulisan ini sangat kompleks, apalagi memori utama

dapat diakses langsung oleh modul I/O, yang memungkinkan data pada memori utama berubah, lalu bagaimana dengan data yang telah dikirim pada cache? Tentunya perbedaan ini menjadikan data tidak valid.

Teknik yang dikenalkan diantaranya, *write through*, yaitu operasi penulisan melibatkan data pada memori utama dan sekaligus pada cache memori sehingga data selalu valid. Kekurangan teknik ini adalah menjadikan lalu lintas data ke memori utama dan cache sangat tinggi sehingga mengurangi kinerja sistem, bahkan bisa terjadi hang.

Teknik lainnya adalah *write back*, yaitu teknik meminimasi penulisan dengan cara penulisan pada cache saja. Pada saat akan terjadi penggantian blok data cache maka baru diadakan penulisan pada memori utama. Masalah yang timbul adalah manakala data di memori utama belum di-update telah diakses modul I/O sehingga data di memori utama tidak valid.

Penggunaan multi cache terutama untuk multi prosesor adan menjumpai masalah yang lebih kompleks. Masalah validasi data tidak hanya antara cache dan memori utama saja, namun antar cache juga harus diperhatikan. Pendekatan penyelesaian masalah yang dapat dilakukan adalah dengan :

- *Bus Watching with Write Through*, yaitu setiap cache controller akan memonitoring bus alamat untuk mendeteksi adanya operasi tulis. Apabila ada operasi tulis di alamat yang datanya digunakan bersama maka cache controller akan menginvalidasi data cache-nya.
- *Hardware Transparency,* yaitu adanya perangkat keras tambahan yang menjamin semua updating data memori utama melalui cache direfleksikan pada seluruh cache yang ada.
- *Non Cacheable Memory,* yaitu hanya bagian memori utama tertentu yang digunakan secara bersama. Apabila ada mengaksesan data yang tidak di share merupakan kegagalan cache.

5.2.6 Jumlah Cache

Terdapat dua macam letak cache. Berada dalam keping prosesor yang disebut *on chip cache* atau cache internal. Kemudian berada di luar chip prosesor yang disebut *off chip cache* atau cache eksternal.

Cache internal diletakkan dalam prosesor sehingga tidak memerlukan bus eksternal, akibatnya waktu aksesnya akan cepat sekali, apalagi panjang lintasan internal bus prosesor sangat pendek untuk mengakses cache internal. Cache internal selanjutnya disebut *cache tingkat 1* (L1).

Cache eksternal berada diluar keping chip prosesor yang diakses melalui bus eksternal. Pertanyaannya, apakah masih diperlukan cache eksternal apabila telah ada cache internal? Dari

pengalaman, masih diperlukan untuk mengantisipasi permintaan akses alamat yang belum tercakup dalam cache internal. Cache eksternal selanjutnya disebut *cache tingkat 2* (L2).

Selanjutnya terdapat perkembangan untuk memisah cache data dan cache instruksi yang disebut *unified cache*. Keuntungan *unified cache* adalah :

- Unified cache memiliki hit rate yang tinggi karena telah dibedakan antara informasi data dan informasi instruksi.
- Hanya sebuah cache saja yang perlu dirancang dan diimplementasikan.

Namun terdapat kecenderungan untuk menggunakan *split cache*, terutama pada mesin – mesin superscalar seperti Pentium dan PowerPC yang menekankan pada paralel proses dan perkiraan – perkiraan eksekusi yang akan terjadi. Kelebihan utama *split cache* adalah mengurangi persaingan antara prosesor instruksi dan unit eksekusi untuk mendapatkan cache, yang mana hal ini sangat utama bagi perancangan prosesor – prosesor pipelining.