ORGANISASI CACHE MEMORY

Arie S. M. Lumenta.
Jurusan Teknik Elektro-FT UNSRAT, Manado-95115, Email: arie.lumenta@unsrat.ac.id

Abstrak

Cache memory dirancang secara umum untuk meningkatkan kinerja suatu sistem komputer. Secara logika cache memory berada diantara prosessor dan memory utama yang bertujuan untuk menyediakan data atau instruksi yang akan diolah oleh prosessor. Hal ini menyebabkan waktu yang dibutuhkan menjadi lebih singkat karena prosessor tidak perlu mengambilnya dari memori utama. Terdapat beberapa metode pemetaan memory utama kepada cache memory yakni pemetaan langsung, cache assosiative, set assosiative dan sector caches. Dalam makalah ini akan dibahas tentang metode pemetaan tersebut. Juga akan dibahas tentang parameter-parameter disain suatu cache memory serta tentang performance suatu cache memory.

Kata kunci : *cache memory, block, direct mapping, assosiative cache, hit ratio, cycle counts*

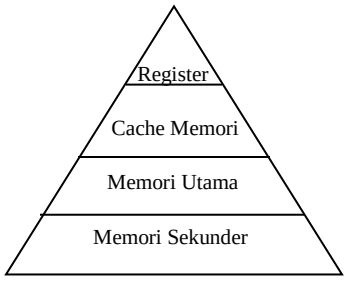
1. Pendahuluan

Idealnya sebuah prosessor tidak membuang waktu untuk mengakses instruksi dan data dari memori utama. Sistem memori utama dituntut untuk secepat mungkin menyediakan sebuah *operand* bagi prosessor disertai dengan kemampuan menyediakan sejumlah data sebanyak yang dapat disediakan per satuan waktu. Semakin cepat prosessor mengolah data, maka semakin sulit bagi sistem memori utama untuk mensuplai operand dalam satu atau dua siklus. Oleh sebab itu diperlukan suatu cara untuk mempercepat waktu akses prosessor ke memori utama

Untuk meningkatkan kecepatan dalam hal menyediakan data dan intstruksi bagi prosessor, memori utama dibatasasi oleh batasan rancangan elektronik dan masalah *packaging*. Karena itu perlu dicari solusi lain untuk menangani masalah tersebut. Salah satu cara yang digunakan adalah dengan membuat suatu memori *cache* yang menyimpan data atau instruksi terbaru dari memori utama dalam bentuk *word-word* yang dapat mempercepat akses prosessor ke memori utama. Selanjutnya akan dibahas mengenai organisasi *cache memory* untuk meningkatkan kinerja suatu sistem komputer secara keseluruhan, terutama memperpendek waktu akses prosessor ke *memory* utama. Termasuk didalamnya pembahasan tentang beberapa aspek penting dalam pengorganisasian suatu *cache* dan beberapa hal penting dalam disain rancangan *cache memory* serta hal-hal yang mempengaruhi performa suatu *cache memory*.

2. Definisi Cache Memori

Dalam hirarki memori posisi *cache memory* adalah satu level dibawah register seperti tampak pada Gambar 1 berikut.



Gambar 1. Hirarki Memori

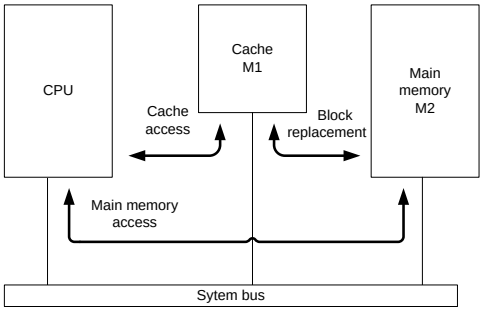
PengGambaran tersebut menunjukkan bahwa semakin keatas ukuran semakin kecil dan kecepatan semakin cepat serta harga semakin mahal. Sebaliknya, dari atas kebawah ukuran semakin besar dan kecepatan makin lambat serta harga makin murah.

Cache memory adalah memori berukuran kecil, buffer memori berkecepatan tinggi yang digunakan komputer untuk menampung sebagian isi memori utama yang sedang digunakan [Smith, 1982].

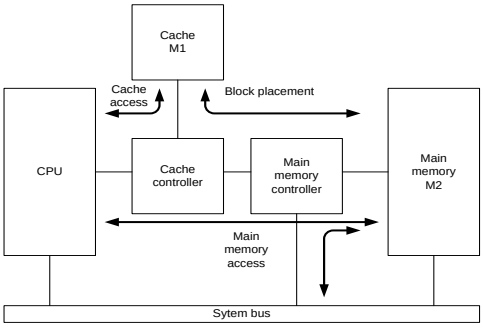
3. Organisasi Cache Memory

Secara logika *cache memory* berada diantara CPU dan memori utama. Ada dua sistem organisasi untuk cache memori seperti ditunjukkan pada Gambar 2.

Secara prinsip komponen sebuah cache ditunjukkan seperti pada Gambar 3. *Memory word* disimpan didalam sebuah *cache data memory* dan dikelompokkan menjadi halaman-halaman kecil yang disebut blok-blok *cache* atau *lines*. Isi dari *cache data memory* adalah *copy* dari satu set blok *memory* utama.



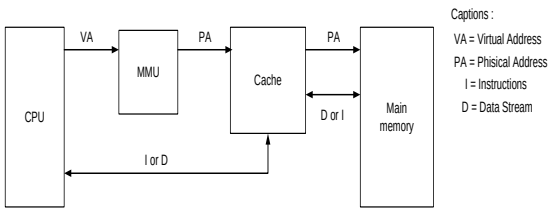
(a) look-aside



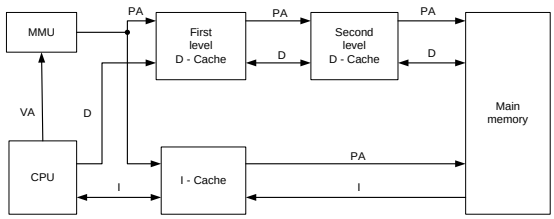
(b) look-through

Gambar 2. Dua sistem organisasi cache memori (Hayes, 1998)

Cache memory memiliki dua jenis pengalamatan yaitu secara fisik dan secara virtual. Pengalamatan secara fisik adalah ketika sebuah cache diakses dengan sebuah alamat fisik memory, seperti pada Gambar 3 dibawah:



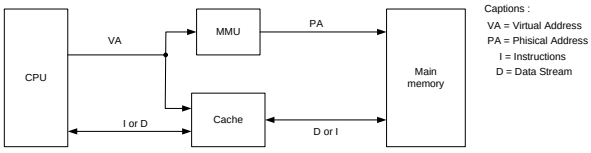
(a) Cache yang diakses dengan alamat fisik.



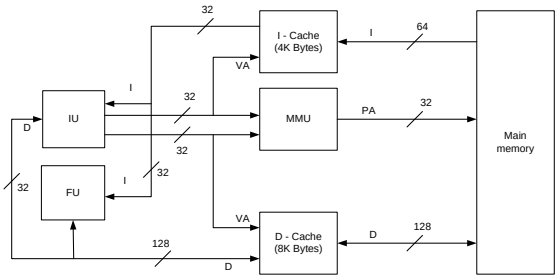
(b) Cache terpisah yang diakses dengan alamat fisik

Gamabar 3. Model pengalamatan fisik sebuah cache memory (Hwang, 1993)

sedangkan pengalamatan virtual adalah ketika sebuah *cache* di-index dengan sebuah alamat virtual seperti Gambar 4:



(a) Cache yang diakses dengan alamat virtual.



(b) Sebuah cache terpisah yang diakses oleeh alamat virtual

Gambar 4. Model pengalamatan virtual sebuah cache.

4. Direct Mapping dan Associative Caches

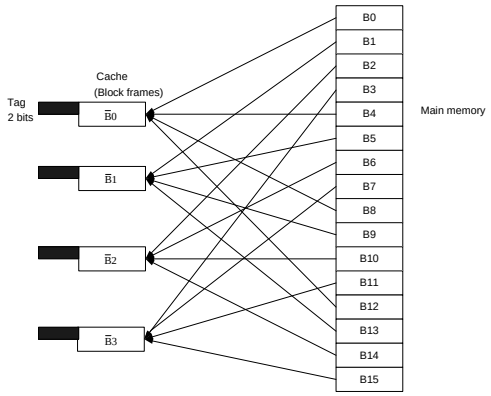
Perpindahan informasi dari memori utama ke cache memory diatur dalam unit-unit blok *cache* atau jalur *cache* [Hwang, 1993]. Direct mapping atau pemetaan langsung adalah teknik pemetaan alamat cache memory yang *relative* sederhana. Blok dalam *cache* yang disebut *block frames* dihubungkan dengan *block* dalam memory utama. Dimisalkan *block frames* adalah $\overline{B}_i$ untuk i=0,1,2,3,…,m dan block dalam memory utama adalah B_j dengan j=0,1,2,3,…n. Dengan *direct mapping* tiap block $\{B_j\}$ dipetakan

langsung dengan *block frame* $\{\bar{B}_i\}$. Diasumsikan pula bahwa n>>m, $n=2^s$ dan $m=2^r$. Tiap *block* atau *block frame* diasumsikan memiliki b words, dimana $b=2^w$ mengakibatkan *cache* berisi $m.b=2^{r+w}$ *words* dan *main memory* memiliki $n.b=2^{s+w}$ *words* yang dialamati oleh (s+w) bits. Ketika block frame dibagi menjadi $v=2^t$ himpunan, maka *block* untuk tiap himpunan adalah $k=m/v=2^{r-t}$.

Organisasi *direct mapping cache* seperti yang diGambarkan pada Gambar 5, didasarkan pada sebuah pemetaan langsung oleh $n/m=2^{s-r}$ block memory yang dipisahkan dengan jarak yang sama ke satu *block frame* dalam *cache*. Pemetaan *block* B_j ke *block frame* $\bar{B}_i$ mengikuti aturan:

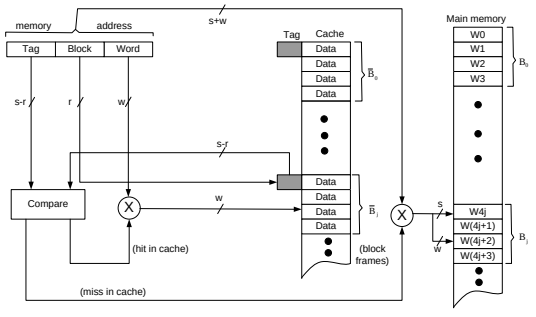
$$B_j \rightarrow \bar{B}_i \text{ jika } i=j \text{ (modulo m)}$$

Organisasi direct mapping ini sangat kaku tetapi merupakan organisasi *cache memory* yang paling sederhana yang dapat diimplementasikan.



Gambar 5. Organisasi cache dengan deirct mapping.

Untuk *direct mapping* dengan kasus dimana tiap *block* mengandung 2 *words* (w=2 bits), pengalamatan memorynya tampak pada Gambar 6 dibawah :

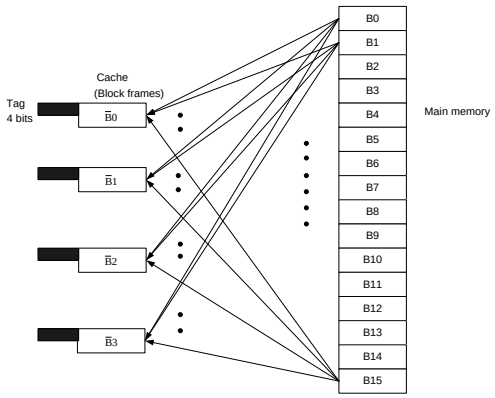


Gambar 6. Pengalamatan cache/memory

Alamat *memory* dibagi menjadi tiga bagian yaitu: *lower* (w) *bits* sebagai *word offset* dalam *block*, *upper* (s) *bits* yang merupakan blok alamat di memory utama dengan *leftmost* (s-r) *bits* sebagai *tag* untuk dicocokan dan *block* (r) bits yang biasanya digunakan untuk mengimplementasikan modulo-m untuk penempatan isi block memory utama ke dalam block cache, dimana $m=2^r$.

Hit cache terjadi ketika dua *tag* sesuai, sebaliknya jika tidak maka yang terjadi adalah *miss cache*. Ketika terjadi *miss* seluruh alamat memori (s+w *bits*) digunakan untuk mengakses memori utama.

Organisasi *assosiative cache* menawarkan fleksibilitas yang tinggi dalam pemetaan *block cache* seperti tampak pada Gambar 7.



Gambar 7. Organisasi assosiative cache

Pada *asssosiative cache* tiap block dalam memory utama dapat ditempatkan di *block frame* mana saja yang tersedia.

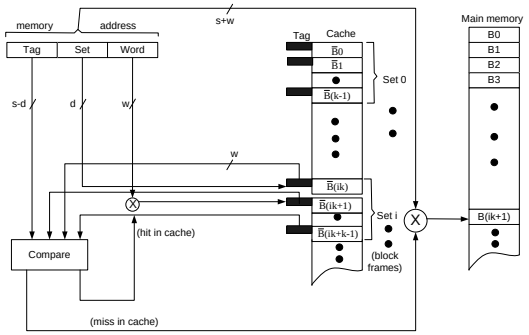
Pemetaan assosiative memberikan kebebasan penuh untuk memilih lokasi *cache* sebagai tempat pemetaan isi blok memori. Hal ini mengakibatkan ruang dalam cache dapat digunakan lebih efisien.

5. *Set-Assosiative* dan *Sector Caches*.

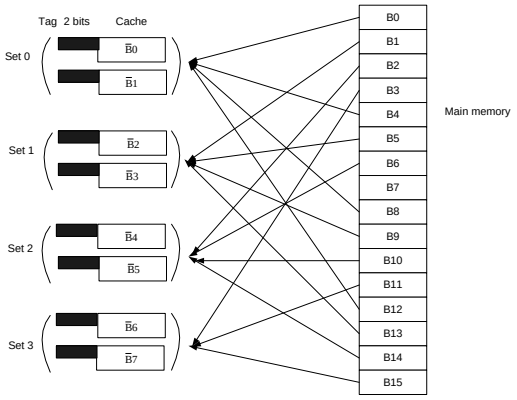
Set assosiative adalah kombinasi teknik *direct mapping* dan *assosiative cache*. Pada metode ini *block cache* dikelompokkan ke dalam *set*, dan aturan pemetaan memungkinkan blok memori utama untuk berada dalam block set tertentu.

Pada k-*way assosiative cache*, m *cache block frames* dibagi menjadi v=m/k *sets*, dimana k adalah *block* per *set*. Tiap *set* diidentifikasi dengan d-*bit set* bilangan, dimana $2^d=v$. *Tag* block cache selanjutnya direduksi menjadi s-d bits. Dalam praktek, ukuran *set* k biasanya

adalah bilangan-bilangan 2, 4, 8, 16, atau 64. Gambar 8 dibawah memperlihatkan metode pemetaan set assosiative.



(a) Pencarian k-way assosiative



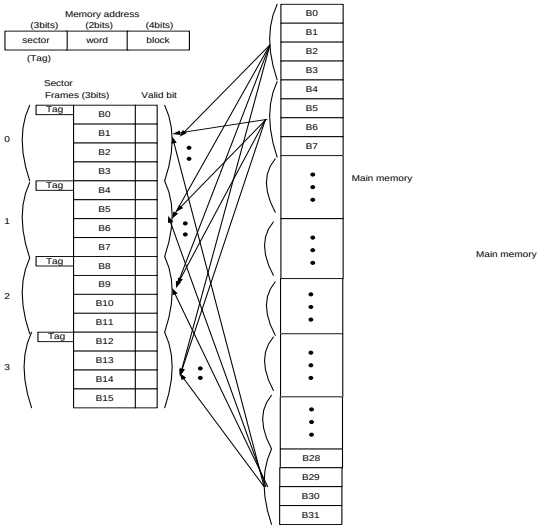
(b) Pemetaan block cache dalam 2-way assosiative.

Gambar 8. Pemetaan Set-Assosiative Cache

Menggunakan 64 *set* berarti *field set* memiliki 6 *bit* alamat untuk menentukan *set cache* mana yang mungkin berisi *block* yang diinginkan. Kemudian *field tag* alamat harus dibandingkan secara *assosiative* terhadap *tag* 2 *block set* untuk memeriksa apakah *block* yang dimaksud berada disana.

Pada *sector mapping cache*, *cache memory* dan memori utama dibagi menjdi sektor-sektor dengan ukuran tetap. Hal ini mengakibatkan tiap sector dapat ditempatkan pada *sector frames* mana saja yang tersedia. Jika terjadi *miss cache* hanya *block* hilang yang diambil kembali dari memori utama dan dibawa ke *block frame* yang sama pada *sector* yang tersedia.

Dibandingkan dengan *assosiative* atau *set assosiative*, *sector mapping cache* menawarkan keuntungan yakni lebih fleksibel untuk diimplementasikan, dapat menggunakan berbagai algoritma perpindahan dan lebih ekonomis untuk jumlah *sector tag* dibandingkan *assosiative*. Contoh *sector mapping* untuk organisasi 4 way diperlihatkan dalam Gambar 9.



Gambar 9. Organisasi cache 4 way sector mapping

6. Hit Rate dan Miss Penalty.

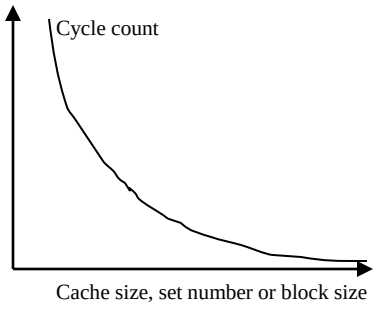
Indikator untuk melihat keefektifan implementasi hirarki memori adalah dengan melihat tingkat keberhasilan mengakses berbagi *level memory*. *Hit* adalah keberhasilan mengakses data dalam suatu *cache*. Jumlah *hit* yang dinyatakan sebagai fraksi semua akses yang dilakukan disebut *hit rate*. Sedangkan *miss rate* adalah jumlah kegagalan akses dilakukan. *Miss penalty* didefinisikan sebagai waktu tambahan yang diperlukan untuk membawa data yang diinginkan ke dalam *cache*. *Penalty* direfleksikan saat prosessor berhenti melakukan proses karena instruksi atau data yang diperlukan tidak tersedia untuk dieksekusi.

Secara umum, *miss penalty* adalah waktu yang diperlukan untuk membawa sutu blok data dari unit yang lebih lambat dalam hirarki memori ke unit yang lebih cepat. *Miss penalty* dapat dikurangi bila diterapkan mekanisme transfer data yang efisien antara berbagi unit hirarki.

7. Cache Performance

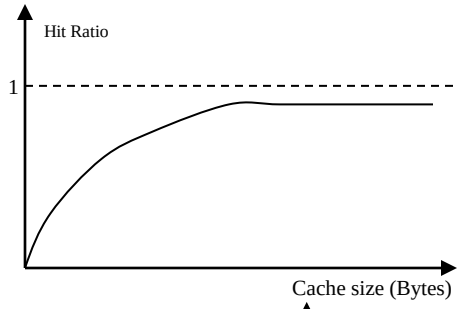
Performa suatu *cache* dapat dilihat dari *cycle counts* dan *hit ratio*. Kecepatan *cache* dipengaruhi oleh teknologi memori utama, organisasi *cache* dan *cache hit ratio*. *Cycle count*

secara langsung berelasi dengan *hit ratio* seperti tampak pada Gambar 10 dibawah. *Write-through* dan *write-back* juga mempengaruhi *cycle counts*. Disamping itu, ukuran *cache*, *block size*, dan jumlah *set*, turut mempengaruhi *cycle count*.



Gambar 10. Total cycle untuk akses cache

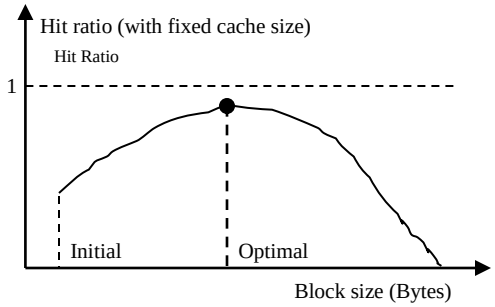
Hit ratio cache dipengaruhi oleh ukuran *cache* dan ukuran *block* seperti tampak pada Gambar 11.



Gambar 11. Hit ratio versus cache size

Gambar 11 diatas menunjukkan bahwa semakin besar ukuran *cache* maka akan mendekati angka *hit ratio* yang ideal. Tetapi memperbesar ukuran *cache* saat performa suatu *cache* telah mendekati angka *hit ratio* yang ideal tidak serta merta akan mencapai *hit ratio* yang ideal. Dari grafik terlihat bahwa pada saat itu memperbesar ukuran *cache* tidak memberikan pengaruh apa-apa, dalam hal ini garis grafik lurus *asimptot* dengan angka *hit ratio* ideal. Kurva pada Gambar 11. dapat didekati dengan persamaan $1 - C^{-0.5}$, dimana C adalah total ukuran *cache*.

Bila ukuran hit ratio tetap maka hubungan antara hit ratio dengan block size menunjukkan bahwa semakin besar ukuran block suatu cache tidak menjamin bahwa performa suatu cache akan meningkat pula. Hal ini dapat dilihat pada Gambar 12.



Gambar 12. Hit Ratio versus block size

Gambar 12 menunjukkan bahwa memperbesar ukuran block suatu cache tidak akan selalu meningkatkan performa suatu cache. Meningkatkan ukuran block cache pada satu titik memang akan meningkatkan performa suatu cache. Hal ini terlihat pada tercapainya titik optimal yang mendekati angka hit ratio ideal. Tetapi jika ukuran block cache terus ditingkatkan hit ratio justru akan semakin turun menuju ke titik 0.

8. Penutup

Dari pembahasan diatas dapat ditarik kesimpulan bahwa cache dapat diorganisasikan dalam beberapa cara pemetaan. Masing-masing teknik memiliki teknik yang berbeda dalam pemindahan data dari memory utama ke dalam cache.

Hit ratio adalah salah satu parameter untuk melihat performa suatu cache. Angka hit ratio yang ideal adalah 1. Meningkatkan ukuran cache dan ukuran block dalam cache tidak serta merta akan meningkatkan performa suatu cache. Dengan kata lain dapat dikatakan bahwa peningkatan ukuran cache dan ukuran block cache tidak linier dengan peningkatan performa suatu cache.

Pustaka

[1] Kai Hwang, **Advanced Computer Architecture: Parallelism, Scalability, Programmability**. McGrawHill, 1993.

[2] Alan J. Smith, **Cache Memories**. ACM Computing Survey, pp. 473-530, Sept. 1982

[3] John P. Hayes, **Computer Architecture and Organization**. 3rd ed. McGrawHill, 1998.