BAB IV

PENJADWALAN MEMORI

MATERI

1. Virtual Memori
2. Algortima Penggantian Page
3. Isu Desain Sitem Paging
4. Segmentasi

STANDAR KOMPETENSI

Mengetahui tentang penjadwalan memori dan segmentasi

CAPAIAN PEMBELAJARAN

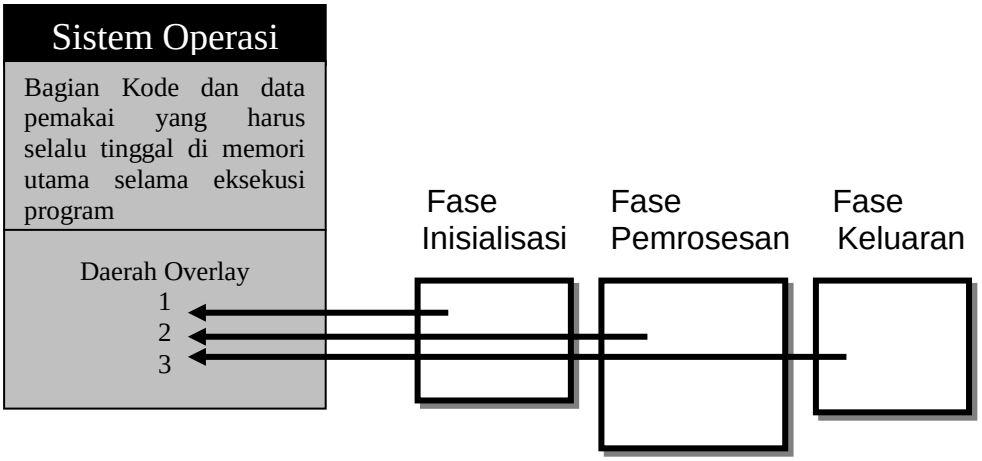
1. Mahasiswa mengetahui tentang virtual memori
2. Mahasiswa mengerti tentang Algortima Penggantian Page
3. Mahasiswa mengetahui Isu Desain Sitem Paging
4. Mahasiswa mengerti tentang Segmentasi

1. Virtual Memori

- **Overlay :**

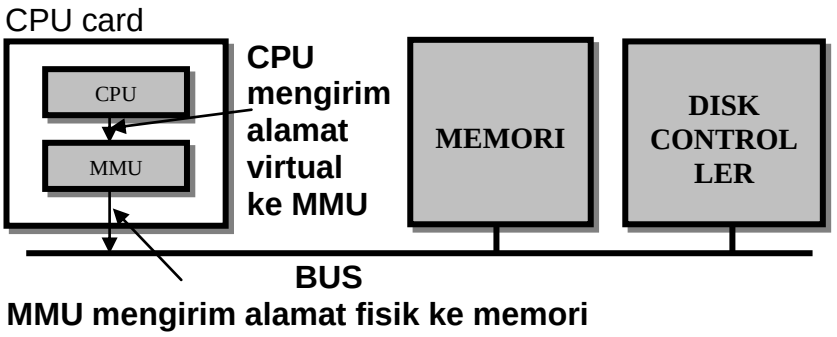
Program dipecah menjadi bagian-bagian yang dapat dimuat memori, jika memori terlalu kecil untuk menampung seluruhnya sekaligus.

Overlay disimpan pada disk dan di-keluar-masukkan dari dan ke memori oleh sistem operasi. Pembagian dilakukan oleh programmer.

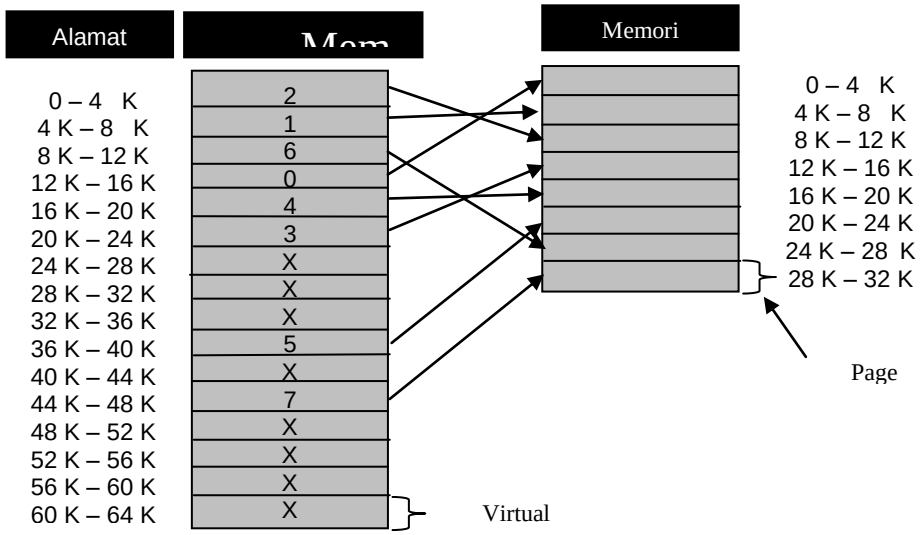


Struktur Umum Overlay

- ***Paging***



Posisi dan fungsi MMU



Relasi Antara Alamat Virtual dan Alamat Fisik

- **Virtual memory (Memori maya) :**

sistem operasi menyimpan bagian-bagian proses yang sedang digunakan di memori utama dan sisanya di disk.

Virtual memory dapat diimplementasikan dengan tiga cara, yaitu:

– Paging
– Segmentasi
– Kombinasi paging dan segmentasi

Sistem paging mengimplementasikan ruang alamat besar pada memori kecil menggunakan index register, base register, segment register, dll.

Istilah pada sistem paging:

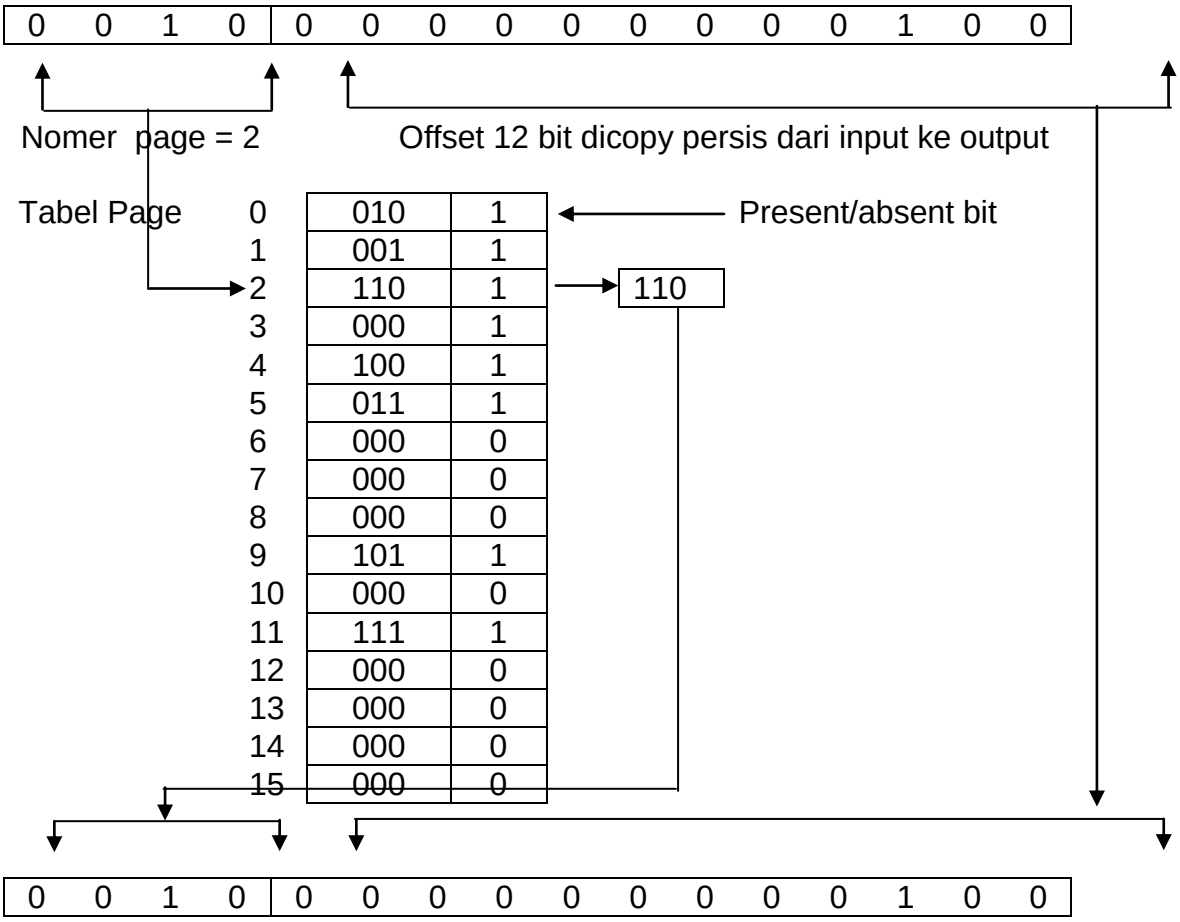
- Alamat virtual = V; Alamat yg dihasilkan dgn perhitungan menggunakan index register, base register, segment reg dsb.
- Alamat nyata (real address = R); Alamat yang tesedia di memori utama fisik.
- Page; Unit terkecil virtual address space.
- Page frame; Unit terkecil memori fisik.
- Page fault; Permintaan alokasi page ke memori yang belum dipetakan.
- MMU (Memory Management Unit); Chip atau kumpulan chip yang memetakan alamat maya ke alamat fisik.

- ***Tabel Page***

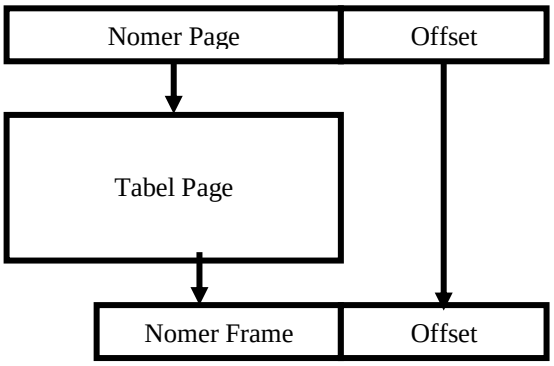
Alamat virtual dibagi menjadi dua bagian:

- Nomer Page (bit-bit awal)
- Offset (bit-bit akhir)

Secara metematis: tabel page merupakan fungsi dgn nomer page sebagai argumen dan nomer frame sebagai hasil.



Cara Kerja Pemetaan oleh MMU



- ***Memori Asosiatif***

  Tabel Page biasanya diletakkan di memori, dengan demikian diperlukan dua kali referensi ke memori : sekali untuk mencari page, dan sekali untuk mencari data yang akan diproses.

  Solusi:

  Komputer dilengkapi dengan komponen hardware kecil untuk pemetaan alamat virtual ke alamat fisik tanpa menelusuri seluruh tabel page.

  Komponen ini disebut ***memori asosiatif*** atau ***translation lookaside buffer***, yang biasanya berada di dalam MMU, dan berisi beberapa entri.

Valid entry ↓	No. page	Modified	Protection	No. frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Memori asosiatif untuk mempercepat paging

Bagian referensi memori yang dapat dipenuhi dari memori asosiatif disebbut ***hit ratio***. Makin tinggi hit ratio, makin baik performance manajemen memori khususnya, dan komputer umumnya.

2. Algoritma Penggantian Page

Saat terjadi ***fault*** berarti harus diputuskan page frame yang harus diganti.

- ***Algoritma penggantian page acak:***

  Page yg dikeluarkan untuk memberi tempat ke yang baru ditentukan secara acak tanpa kriteria tertentu.

- ***Algoritma penggantian page optimal****:*

  Setiap page diberi label untuk menandai berapa instruksi lagi baru dia digunakan. Page dengan label tertinggi (waktu dari sekarang sampai pemakaian berikutnya paling lama) yang akan dikeluarkan.

  Algoritma Penggantian Page Optimal

String Pengacuan		2	3	2	1	5	2	4	5	3	2	5	2	
		2	2	2	2	2	2	4	4	4	2	2	2	
			3	3	3	3	3	3	3	3	3	3	3	
					1	5	5	5	5	5	5	5	5	
Fault		F	F		F	F		F			F			6 Fault

- ***Algoritma penggantian page NRU (not recently used)****:*

  Setiap page diberi status bit R (referenced) dan M (modified).
  Bit bernilai 0 jika page belum direferensi/dimodifikasi, dan 1 jika sebaliknya. Dari nilai desimalnya didapat 4 kelas:

R	M	Kelas	Keterangan	
0	0	0	not referenced,	not modified
0	1	1	not referenced,	modified
1	0	2	referenced,	not modified
1	1	3	referenced,	modified

  Page dengan kelas terkecillah yang akan dikeluarkan

  Algoritma Penggantian Page FIFO

String Pengacuan		2	3	2	1	5	2	4	5	3	2	5	2	
		2	2	2	2	2	2	4	4	4	2	2	2	
			3	3	3	3	3	3	3	3	3	3	3	
					1	5	5	5	5	5	5	5	5	
Fault		F	F			F	F	F		F		F	F	8 Fault

- ***Algoritma penggantian page FIFO (First In First Out)**:*

  Page yang paling dulu masuk ke memori dari semua page yang ada dikeluarkan.

  Anomali pada FIFO (Belady's Anomaly)

String Pengacuan		0	1	2	3	0	1	4	0	1	2	3	4	
Page Termuda		0	1	2	3	0	1	4	4	4	2	3	3	
			0	1	2	3	0	1	1	1	4	2	2	
Page Tertua				0	1	2	3	0	0	0	1	4	4	
Fault		F	F			F	F	F		F		F	F	9 Fault

(a)

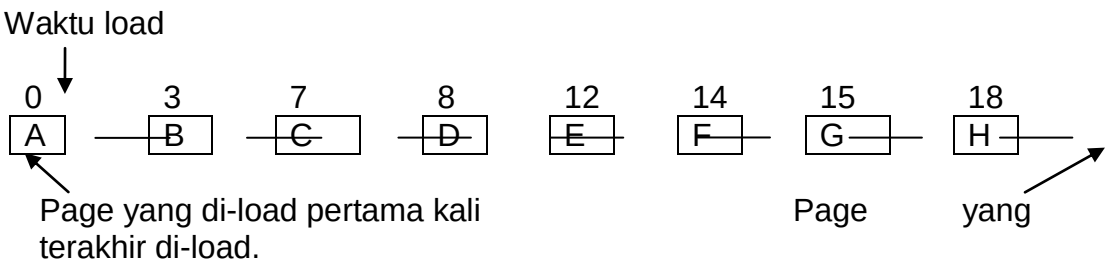
String Pengacuan		0	1	2	3	0	1	4	0	1	2	3	4	
Page Termuda		0	1	2	3	3	3	4	0	1	2	3	4	
			0	1	2	2	2	3	4	0	1	2	3	
				0	1	1	1	2	3	4	0	1	2	
Page Tertua					0	0	0	1	2	3	4	0	1	
Fault		F	F			F	F	F		F		F	F	10 Fault

(b)

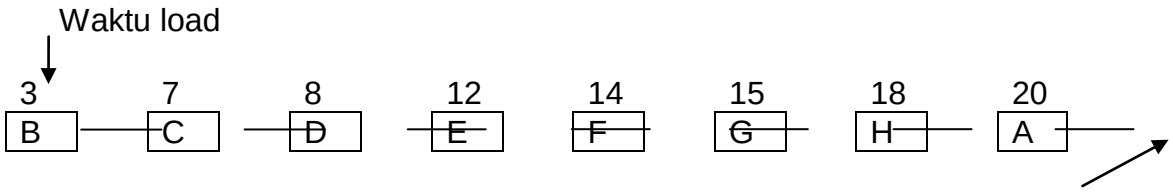
- ***Algoritma penggantian page Modifikasi FIFO (Second Chance):***

  Mencari page yang berada di memori paling lama, tetapi juga tidak dipakai.

  Jika sebuah page dipakai (direferensi) bit R diset. Jika sistem menemukan bahwa bit R page yang paling lama ter-set, page tersebut tidak jadi dikeluarkan, tetapi bit R-nya di-reset.



(a) Page dalam urutan FIFO



(b) Daftar page setelah page fault pada waktu 20
dan bit R page A dalam keadaan set.

Pada algoritma ini, daftar page bisa juga dibuat berbentuk jam (clock page replacement algorithm)

Algoritma penggantian page clock

String Pengacuan		2	3	2	1	5	2	4	5	3	2	5	2	
	>	2	2	2	>2*	2*	2*	2*	>2*	>2	>2*	>2*	>2*	
		>	3	3	3	5	5	5	5*	5	5	5*	5*	
			>	>	1	>1	>1	4	4	3	3	3	3	
Fault		F	F		F	F		F		F				6 Fault

Keterangan : * adalah diacu dan > adalah ditunjuk pointer

- ***Algoritma penggantian page LRU (Least Recently Used):***

  Yang dikeluarkan ialah page yang sudah tidak terpakai dalam waktu paling lama.

  Algoritma Penggantian Page LRU

String Pengacuan		2	3	2	1	5	2	4	5	3	2	5	2	
		2	2	2	2	2	2	4	4	3	3	3	3	
			3	3	3	5	5	5	5	5	5	5	5	
					1	1	1	4	4	4	2	2	2	
Fault		F	F		F	F		F		F	F			7 Fault

3. Isu Desain Sistem Paging

- **Model Working Set**
  - Dalam bentuk paging murni, proses dimulai dengan memori kosong, dan page-page dimasukkan ke dalamnya setelah diminta. Cara ini disebut **demand paging**.
  - **Locality of reference**: Kecenderungan proses untuk memakai satu set page yang sama selama beberapa saat.
  - Satu set page tersebut di atas membentuk **working set**. Dalam hal ini, yang diusahakan oleh sistem operasi adalah agar working set berada utuh di memori pada saat eksekusinya.
  - Jika ukuran memori terlalu kecil untuk working set, akan seringkali terjadi page fault. Hal ini disebut **thrashing**.
  - Banyak sistem paging yang mengusahakan agar working set sudah ada di memori *sebelum* proses dimulai. Pendekatan ini disebut **model working set**. Tujuannya adalah untuk memperkecil jumlah terjadinya page fault (page yang diminta tidak ada di memori).
  - Memasukkan page ke memori sebelum proses dimulai juga disebut **prepaging**.
  - Untuk pertama kali menentukan working set, dipakai sistem **aging** untuk mengetahui berapa kali jumlah pemakaian setiap page.

- **Alokasi Global dan Lokal**
  - Pada sistem timesharing, isi memori bisa seperti pada Gambar a.
  - Misalkan diminta page A6. Jika dikeluarkan A5 untuk memberi tempat ke A6, berarti dilakukan alokasi **lokal**. Bila yang dikeluarkan adalah B3, dilakukan alokasi **global**.

- Algoritma lokal berhubungan dengan pemberian jumlah frame yang sama untuk setiap proses, sementara algoritma global secara dinamis mengalokasikan frame untuk proses yang berjalan.

	Age
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

(a)

A0
A1
A2
A3
A4
A5 -> A6
B0
B1
B2
B3
B4
B5
B6
C1
C2
C3

(b)

A0
A1
A2
A3
A4
A5
B0
B1
B2
B3 -> A6
B4
B5
B6
C1
C2
C3

(c)

Penggantian page global vs. lokal. (a) Konfigurasi awal. (b) Penggantian page lokal. (c) Penggantian page global.

- **Ukuran Page**
  - Ukuran page merupakan salah satu parameter yang ditentukan oleh perancang sistem operasi.
  - Penentuan ukuran page yang optimum harus menyeimbangkan beberapa faktor.
  - Rata-rata, page terakhir hanya akan terisi setengah (fragmentasi internal), berarti page sebaiknya kecil. Tetapi page yang kecil akan menghasilkan tabel page yang panjang.
  - s (byte) = ukuran proses rata-rata
  - p (byte) = ukuran page
  - e (byte) = ukuran setiap page entry

- s/p = perkiraan jumlah page yang dibutuhkan per-proses
- se/p (byte) = ruang untuk tabel page
- p/2 = memori yang terbuang karena fragmentasi
- overhead = memori yang terpakai untuk tabel page dan fragmen internal.

  overhead = se/p + p/2
- Ukuran tabel page besar jika ukuran page kecil. Fragmen internal besar jika ukuranb page besar. Optimum harus ada di antaranya. Dengan mengambil penurunan pertama terhadap p dan menyemakan dengan nol:

  $-se/p^2 + ½ = 0$
- Dari persamaan ini, ukuran page optimum adalah: **$p = \sqrt{(2se)}$**
- Sebagian besar komputer komersial menggunakan ukuran page antara 512 byte – 8K.

- **Isu Implementasi**

- ***Instruction backup***

  Instruksi yang menyebabkan referensi ke page yang belum ada di memori (menyebabkan page fault) harus diulang ketika page tersebut telah tersedia. Beberapa sistem operasi meng-copy setiap instruksi sebelum dilaksanakan sehingga hal ini tidakmakan waktu terlalu lama.
- ***Locking pages in memory***

  Pada saat satu proses menjalani tahap I/O, proses lain bisa dijalankan. Yang mungkin terjadi ialah page proses I/O tersebut digantikan oleh proses yang kedua ini (jika dipakai alokasi global). Jalan keluarnya ialah dengan me-lock page-page proses I/O.
- ***Shared pages***

  Dua atau lebih proses bisa memakai bersama page-page yang berasal dari editor yang mereka pakai. Penutupan salah satu proses ini tanpa disengaja bisa mengosongkan juga page yang dipakai bersama tersebut.

Diperlukan suatu struktur data khusus untuk memantau page-page yang dipakai bersama ini.

- ***Backing Store***

  Pada disk, disediakan area untuk menampung page yang dikeluarkan dari memori (paged out) yang disebut swap area. Setiap proses memiliki swap area di disk. Swap area ada yang statis ada juga yang dinamis.

- ***Paging Daemon***

  Untuk meyakinkan tersedianya frame bebas yang cukup banyak, banyak sistem paging yang menggunakan proses background yang disebut paging daemon. Jika jumlah frame bebas terlalu sedikit, paging daemon akan mengosongkan beberapa page setelah menulisnya ke disk jika pernah dimodifikasi.

- ***Penanganan Page Fault***

- **Urutan langkah-langkah penanganan adalah sebagai berikut:**

- Hardware melakukan trap ke kernel, program counter di-save ke stack. Pada banyak mesin, beberapa informasi tentang status instruksi saat itu di-save di register-register khusus CPU.
- Rutin kode assembly dimulai untuk men-save register-register umum dan informasi lain yang bisa berubah, agar sistem operasi tidak merusaknya. Rutin ini memanggil sistem operasi sebagai suatu prosedur.
- Sistem operasi menemukan bahwa terjadi page fault, dan mencoba menemukan page virtual mana yang diperlukan. Seringkali salah satu register hardware berisi informasi ini. Jika tidak, sistem operasi harus menarik program counter, mengambil instruksi, dan melakukan parsing pada software untuk mengetahui apa yang dilakukan sebelum terjadi fault.
- Begitu alamat virtual yang menyebabkan fault diketahui, sistem operasi memeriksa apakah alamat ini valid dan proteksinya konsisten dengan

akses. Jika tidak, proses dikirimi sinyal atau ditutup. Jika alamat valid dan tidak ada pelanggaran proteksi, sistem berusaha untuk mendapatkan frame page dari daftar frame bebas. Jika tidak ada frame yang bebas, dijalankan algoritma penggantian page untuk mencari yang bisa ditukar.

- Jika frame page yang dipilih telah dimodifikasi, page dijadwalkan untuk ditransfer ke disk, dan terjadi pertukaran proses, menghentikan sementara proses yang fault dan membiarkan yang lainnya berjalan hingga transfer disk selesai. Frame ditandai terpakai untuk mencegah dipakai untuk tujuan lain.
- Begitu frame page bersih (apakah langsung atau setelah disave ke disk), sistem operasi menelusuri alamat disk di mana page diperlukan, dan menjadwalkan operasi disk untuk memasukkannya. Sementara page dimasukkan, proses yang mengalami fault dihentikan sejenak dan yang lainnya dijalankan, jika ada.
- Ketika disk interrrupt menandai bahwa page telah ada, tabel page di-update untuk menunjukkan posisinya, dan frame ditandai berada dalam status normal.
- Instruksi yang menyebabkan fault di-back-up ke status mulainya dan program counter di-reset untuk menunjuk ke instruksi tersebut.
- Proses yang fault tersebut dijadwalkan, dan sistem operasi kembali ke rutin bahasa assembly yang memanggilnya.
- Rutin ini mengembalikan register dan informasi lainnya ke keadaan semula , dan kembali ke user untuk melanjutkan eksekusi, seakan-akan tidak ada fault yang terjadi.

4. Segmentasi

Compiler bisa memiliki beberapa tabel dengan alamat virtual yang terpisah, misalnya terdiri dari tabel-tabel untuk:

✓ Source text,

- ✓ Tabel simbol,
- ✓ Tabel untuk semua konstanta integer dan floating point,
- ✓ Parse tree, berisi analisis sintaksis program, dan
- ✓ Stack yang digunakan untuk pemanggilan prosedur.

Tabel 1 s/d 4 bisa bertambah pada saat kompilasi berjalan, sehingga dengan sistem paging yang berukuran tetap, batas satu page bisa terlampaui. Dengan alasan ini dipakai bagian-bagian dengan alamat yang relatif independen, yang disebut **segmen**. Setiap segmen mempunyai ukuran yang berbeda dengan yang lain. Panjang segmen juga bisa berubah selama eksekusi.

Program harus menyediakan alamat yang terdiri dari dua bagian:

- nomer segmen
- alamat di dalam segmen

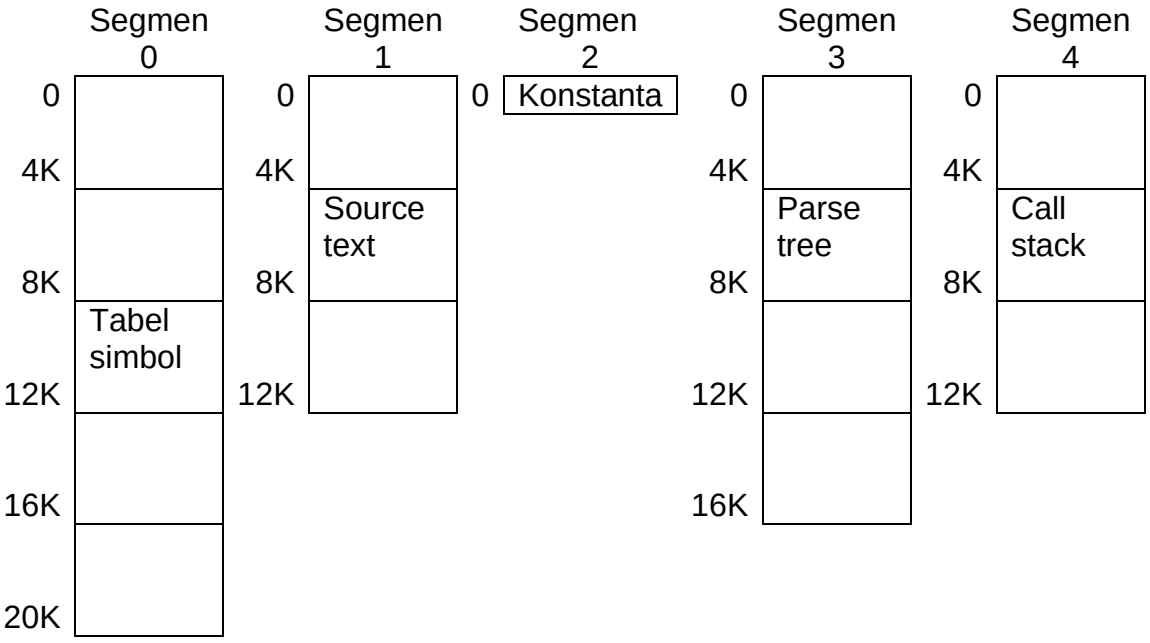
Segmentasi juga memberikan fasilitas pemakaian bersama prosedur atau data antar beberapa proses. Contoh umumnya adalah **shared library**.

Memori yang tersegmentasi memungkinkan setiap tabel bertambah atau berkurang.

Pertimbangan	Paging	Segmentasi
Apakah programmer harus menyadari bahwa teknik ini sedang digunakan?	Tidak	Ya
Berapa banyak ruang alamat linier yang ada?	1	Banyak
Dapatkah ruang alamat total melebihi ukuran memori fisik?	Ya	Ya
Apakah tabel yang ukurannya berubah-ubah dapat diakomodasi?	Tidak	Ya
Dapatkan prosedur dan data dibedakan dan diproteksi secara terpisah?	Tidak	Ya

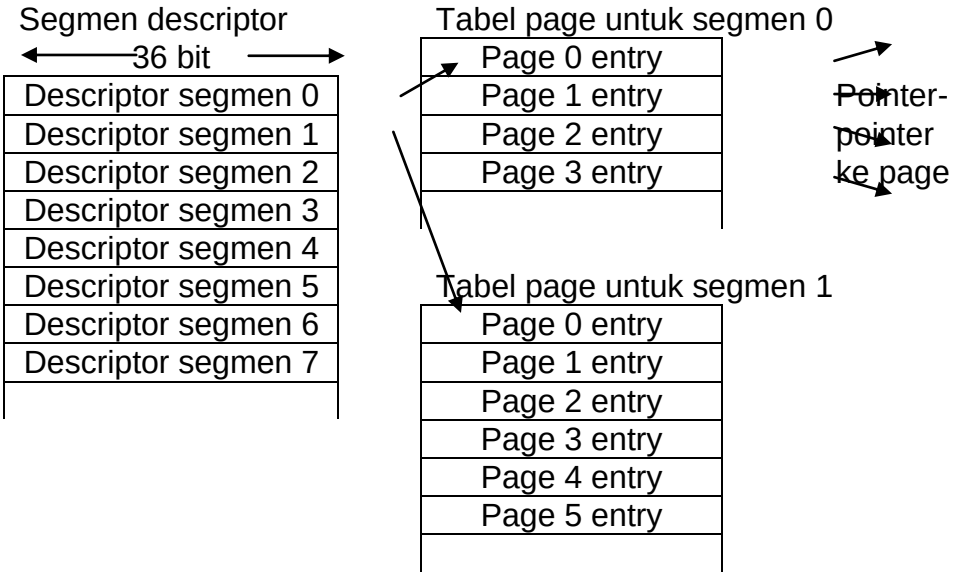
Adakah fasilitas pemakaian bersama prosedur antar user?	Tidak	Ya
Mengapa teknik ini diciptakan?	Untuk mendapatkan ruang alamat linier yang besar tanpa harus membeli memori fisik tambahan	Untuk memungkinkan program dan data dibagi menjadi ruang alamat yang secara logik independen dan untuk membantu pemakaian bersama dan proteksi

Perbandingan paging dan segmentasi.

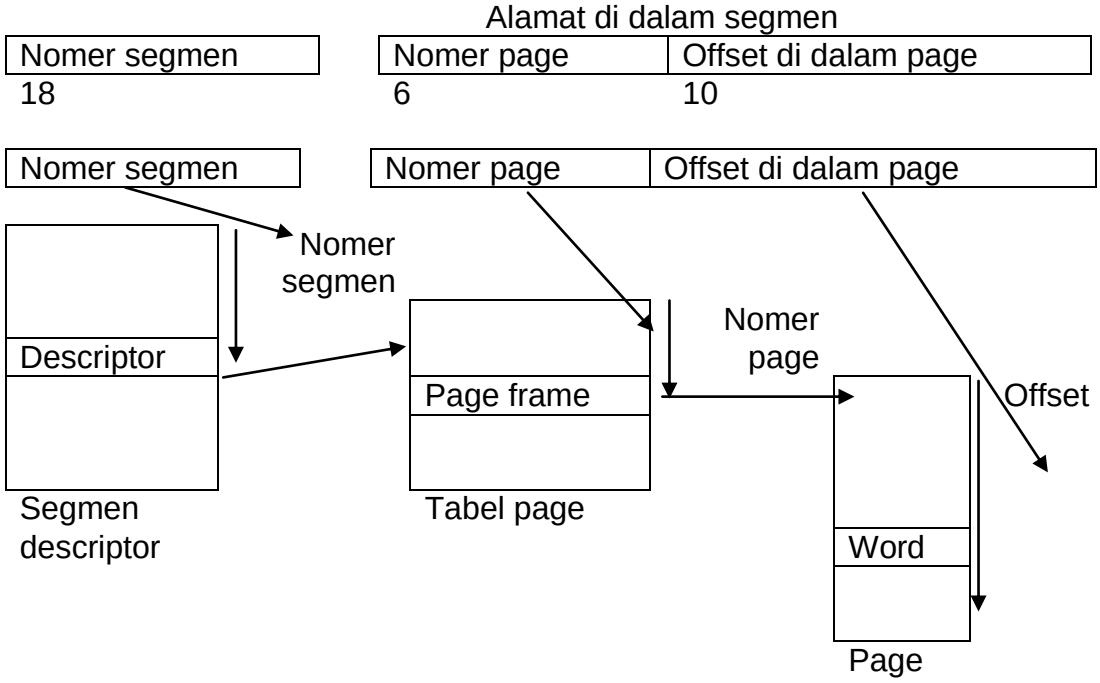


Checkerboarding: Timbulnya blok-blok memori yang kosong (hole) pada saat isi segmen dikeluarkan. Hal ini diatasi dengan pemampatan (compaction).

Segmentasi dengan Paging : Setiap segmen dapat dianggap sebagai satu virtual memori, dan masing-masing dibagi menjadi page-page. Salah satu mesin yang memakai cara ini adalah MULTICS. Setiap program MULTICS memiliki satu tabel segmen, dengan satu descriptor per segmen. **Segmen descriptor** berisi keterangan apakah segmen yang bersangkutan ada di memori atau tidak.



Virtual address MULTICS 34-bit:



Konversi alamat MULTICS menjadi alamat memori utama.

CONTOH SOAL

1. Pada sistem paging alamat logika terdiri dari 2 bagian, yaitu :
    1) Nomer page p : Digunakan sebagai indeks untuk page table yang berisi alamat awal f untuk setiap page pada memori.
    2) Offset page d : offset page ditambahkan pada alamat awal untuk menghasilkan alamat memori sebenarnya.

2. Apabila diketahui page table sebagai berikut :

Nomor page	Frame
0	8
1	2
2	10
3	22
4	12
5	1
:	:
:	:

Dengan asumsi bahwa program membutuhkan page secara berurutan dari 0 sampai n, dimanakah letak alamat fisik dari alamat logika 50, 121, dan 380

1 page = 64 byte.

Menurut page table diatas page 0 akan dipetakan ke frame 8, maka alamat logika 0 akan dipetakan ke alamat fisik (8 * 64) + 0 = 512.

Keadaan memori logika dapat digambarkan sebagai berikut :

Page	Memori Logika
0	0 : 63
1	64 : 127
2	128 : 191
3	192 : 255
4	256 : 319
5	320 : 383
6	384 :
:	:

Dari gambar tersebut dapat dilihat bahwa :

• alamat logika 50 berada di page 0, offset 50 sehingga alamat fisiknya (8 * 64) + 50 = 562

• alamat logika 121 berada di page 1,offset 57 sehingga alamat fisiknya (2 * 64) + 57 = 185

• alamat logika 380 berada di page 5,offset 60 sehingga alamat fisiknya (1 * 64) + 60 = 124

Keterangan :
alamat offset diperoleh dari nilai absolut alamat logika yang ditentukan dikurangi dengan alamat logika awal dari page yang diketahui. Contoh : jika alamat logika 380 berarti alamat offsetnya adalah absolut(380 – 320) = 60

EVALUASI

1. Jelaskan apa yang dimaksud dengan virtual memori
2. Sebutkan fungsi dari manajemen memori tiga cara implementasi memori, berikan contohnya
3. Jelaskan kenapa perlu ada mekanisme pergantian page, dan apa saja keuntungannya
4. Sebutkan macam-macam algoritma penggantian page, berikan contohnya
5. Sebutkan isu apa saja yang terjadi pada desain sistem paging, berikan contohnya
6. Sebutkan tabel apa saja yang bisa dimiliki oleh sebuah Compiler, berikan contohnya

DAFTAR PUSTAKA :

Harvey M Deitel dan Paul J Deitel. 2005. *Java How To Program*. Sixth Edition. Prentice Hall.

Bambang Hariyanto. 1997. *Sistem Operasi* . Buku Teks Ilmu Komputer . Edisi Kedua. Informatika. Bandung.

John L Hennessy dan David A Patterson. 2002. *Computer Architecture* . A Quantitative Approach . Third Edition. Morgan Kaufman. San Francisco.

Randall Hyde. 2003. *The Art of Assembly Language*. First Edition. No Strach Press.

Kenneth H Rosen. 1999. *Discrete Mathematics and Its Application*. McGraw Hill.

Ronald L Krutz dan Russell D Vines. 2001. *The CISSP Prep Guide Mastering the Ten Domains of Computer Security*. John Wiley & Sons.

Sri Kusumadewi. 2000. *Sistem Operasi* . Edisi Dua. Graha Ilmu. Yogyakarta.

Robert Love. 2005. *Linux Kernel Development* . Second Edition. Novell Press.

Larry L Peterson dan Bruce S Davie. 2000. *Computer Networks A Systems Approach*. Second Edition. Morgan Kaufmann.

Riri Fitri Sari dan Yansen. 2005. *Sistem Operasi Modern* . Edisi Pertama. Andi. Yogyakarta.

Betha Sidik. 2004. *Unix dan Linux*. Informatika. Bandung.

Abraham Silberschatz, Peter Galvin, dan Greg Gagne. 2002. *Applied Operating Systems*. Sixth Edition. John Wiley & Sons.

Avi Silberschatz, Peter Galvin, dan Grag Gagne. 2005. *Operating Systems Concepts*. Seventh Edition. John Wiley & Sons.

William Stallings. 2001. *Operating Systems: Internal and Design Principles*. Fourth Edition. Edisi Keempat. Prentice-Hall International. New Jersey.

Andrew S Tanenbaum dan Albert S Woodhull. 1997. *Operating Systems Design and Implementation*. Second Edition. Prentice-Hall.

Andrew S Tanenbaum. 2001. *Modern Operating Systems*. Second Edition. Prentice-Hall.