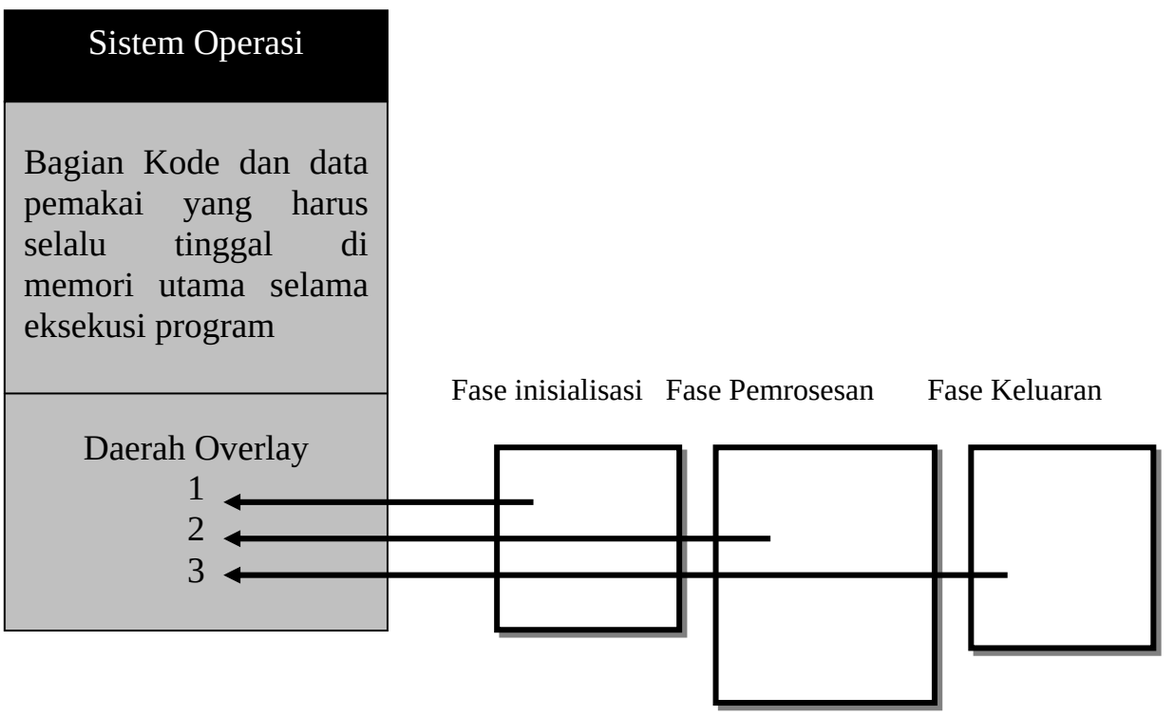
VIRTUAL MEMORY

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.



Gambar 1. Struktur Umum Overlay

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:

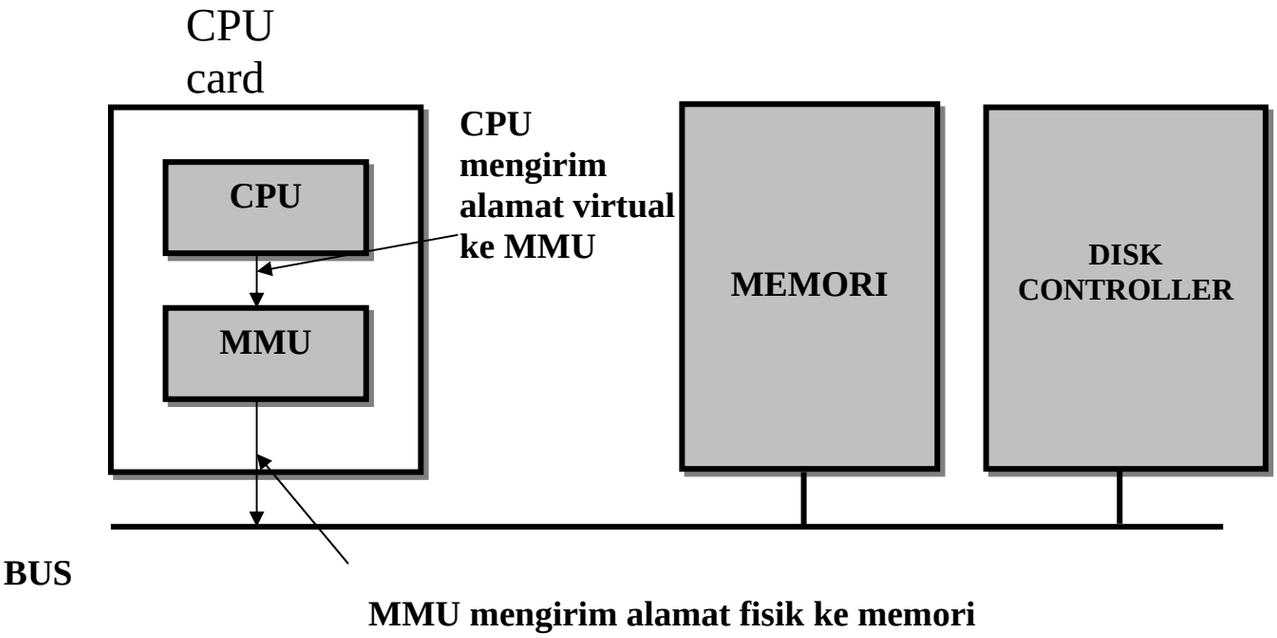
- Paging
- Segmentasi
- Kombinasi paging dan segmentasi

1. Paging

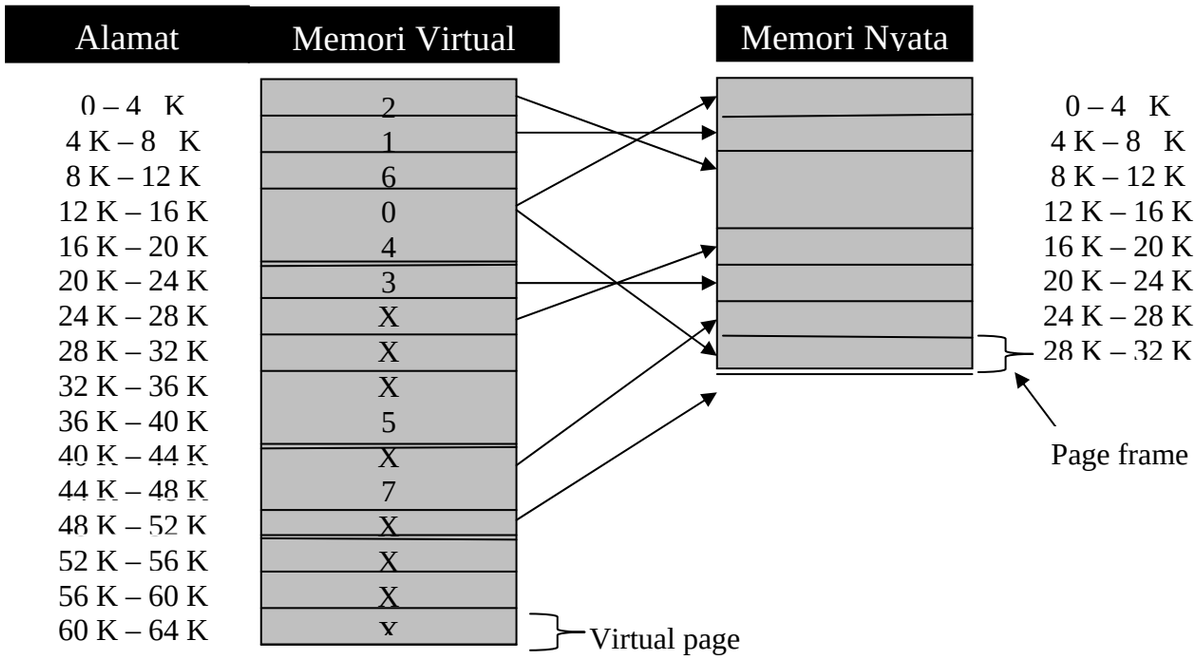
Sistem paging mengimplementasikan ruang alamat besar pada memori kecil menggunakan index register, base register, segment register, dll.

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.



Gambar 2. Posisi dan fungsi MMU
(Tanenbaum, Bab 3, Hlm. 90)



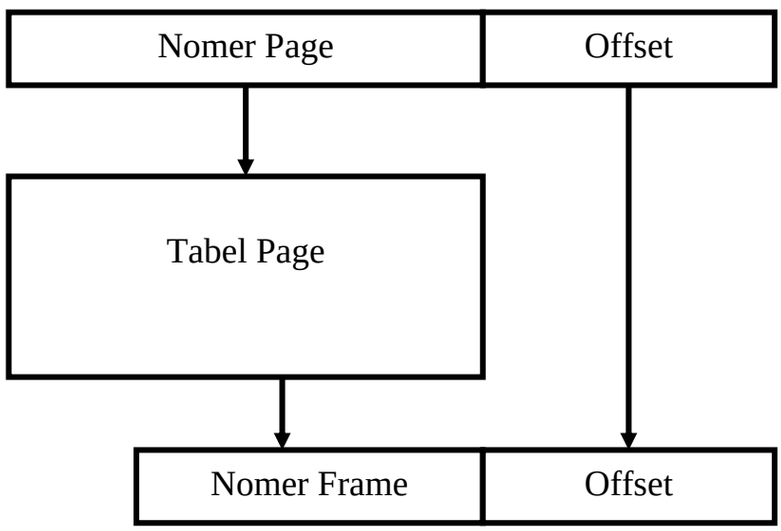
Gambar 3. Relasi Antara Alamat Virtual dan Alamat Fisik
(Tanenbaum, Bab 3, Hlm. 91)

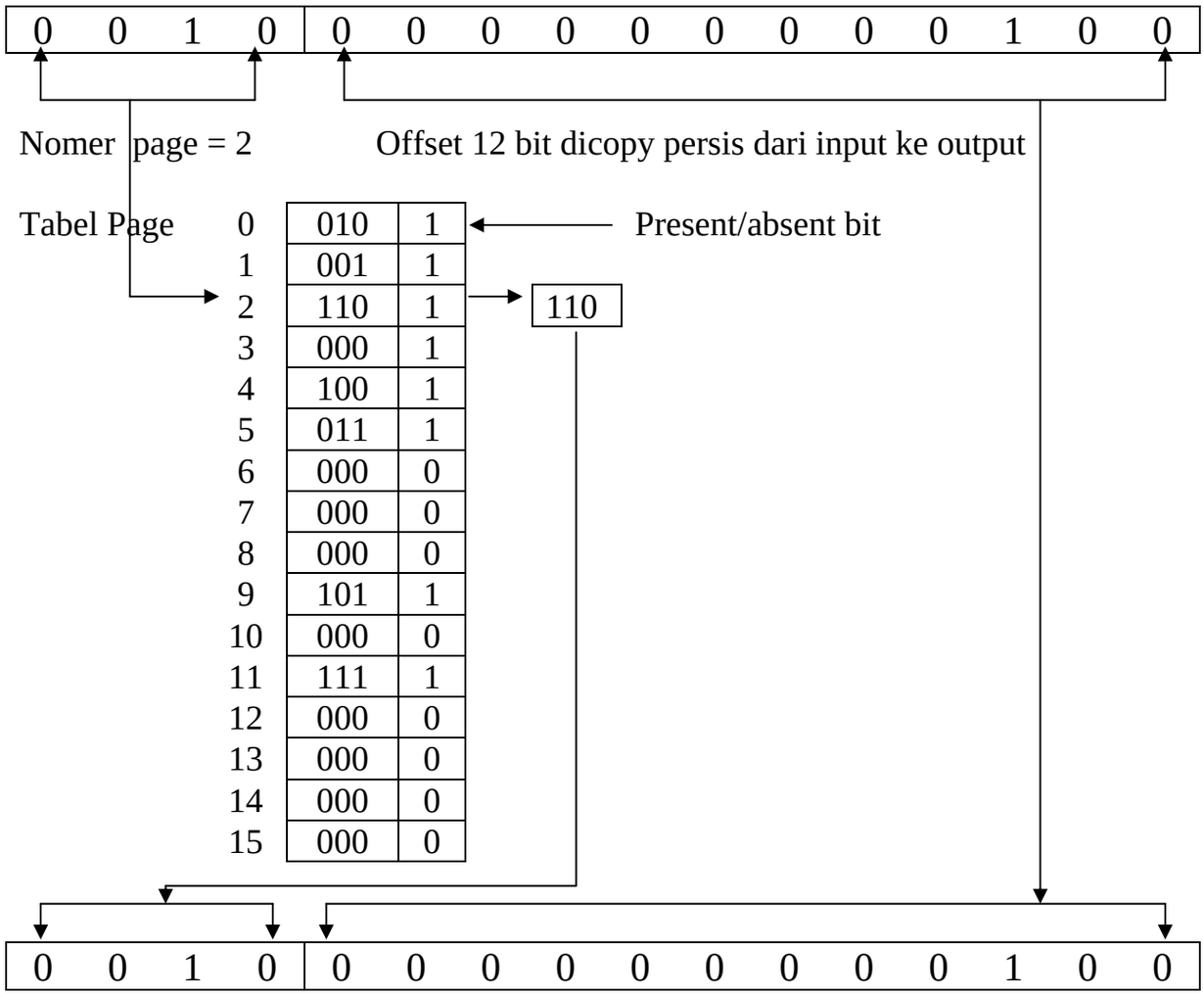
2. Tabel Page

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.





Gambar 4. Cara Kerja Pemetaan oleh MMU
(Tanenbaum, Bab 3, Hlm. 92)

3. 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

Gambar 5. Memori asosiatif untuk mempercepat paging
(Tanenbaum, Bab 3, Hlm. 103)

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.

ALGORITMA PENGGANTIAN PAGE

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

1. **Algoritma penggantian page acak**:
   Page yg dikeluarkan untuk memberi tempat ke yang baru ditentukan secara acak tanpa kriteria tertentu.

2. **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

3. **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.

4. **Algoritma penggantian page FIFO (First In First Out)**:
   Page yang paling dulu masuk ke memori dari semua page yang ada 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

Anomali pada FIFO (Belady's Anomaly)

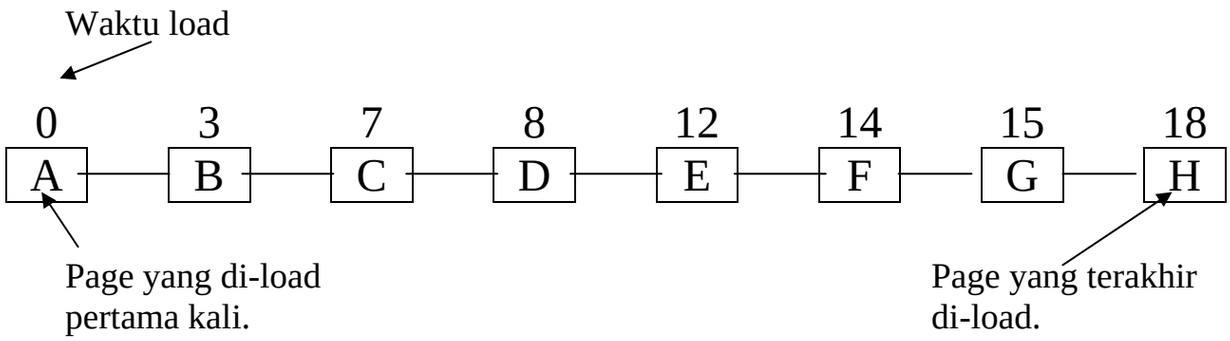
String Pengacuan		0	1	2	3	0	1	4	0	1	2	3	4	
Page Termuda		0	1	2	3	0	1	4	4	4	2	3	3	
			0	1	2	3	0	1	1	1	4	2	2	
Page Tertua				0	1	2	3	0	0	0	1	4	4	
Fault		F	F			F	F	F		F		F	F	9 Fault

(a)

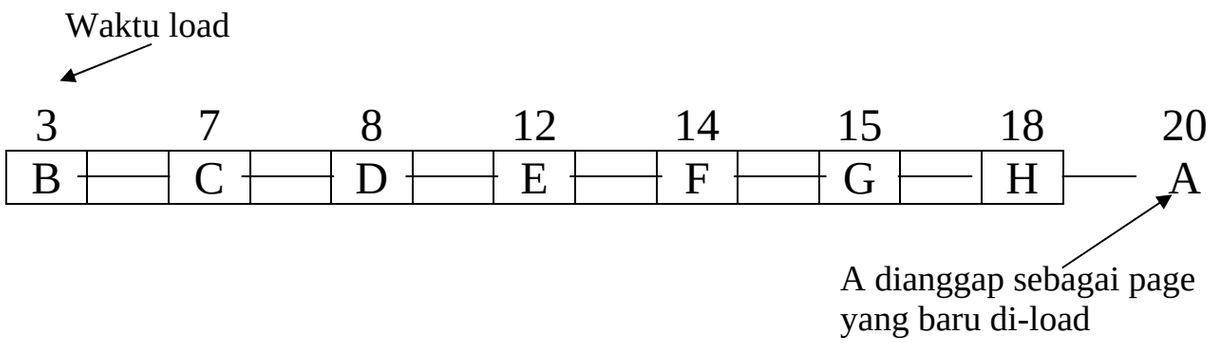
String Pengacuan		0	1	2	3	0	1	4	0	1	2	3	4	
Page Termuda		0	1	2	3	3	3	4	0	1	2	3	4	
			0	1	2	2	2	3	4	0	1	2	3	
				0	1	1	1	2	3	4	0	1	2	
Page Tertua					0	0	0	1	2	3	4	0	1	
Fault		F	F			F	F	F		F		F	F	10 Fault

(b)

5. **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)



(b)

Gambar 1. (a) Page dalam urutan FIFO. (b) Daftar page setelah page fault pada waktu 20 dan bit R page A dalam keadaan set.

(Tanenbaum, Bab 3, Hlm. 110)

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 :

* diacu

> ditunjuk pointer

6. **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