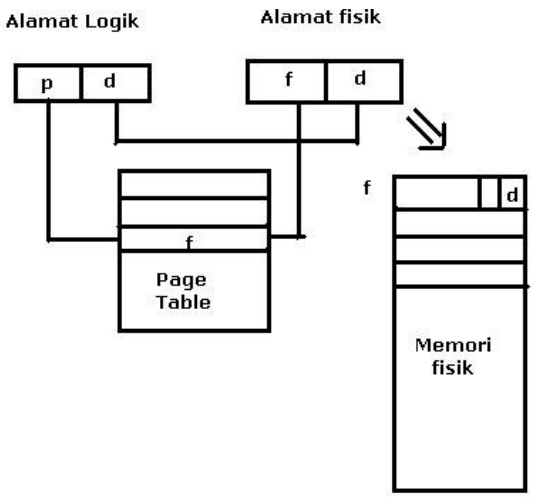
B. Sistem Paging

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



(Gambar : Penerjamahan Page)

Alamat Maya

Alamat yang dihasilkan perhitungan menggunakan index register, base register, dan segment register.

Alamat Nyata

Alamat di memori fisik.

Page

Unit terkecil pada ruang alamat maya (*virtual address space*).

Page Frame

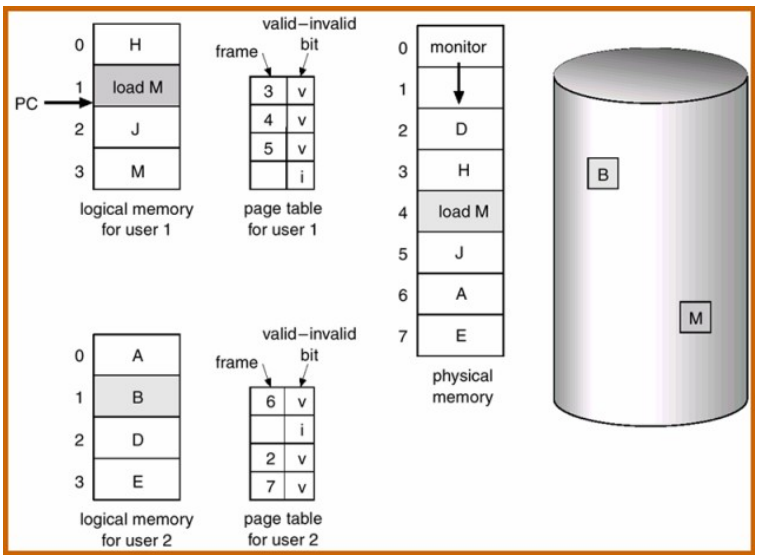
Unit terkecil pada ruang alamat fisik (*real address space*).

Page Fault

Exception untuk permintaan alokasi "page" ke memori.

Memory Management Unit (MMU)

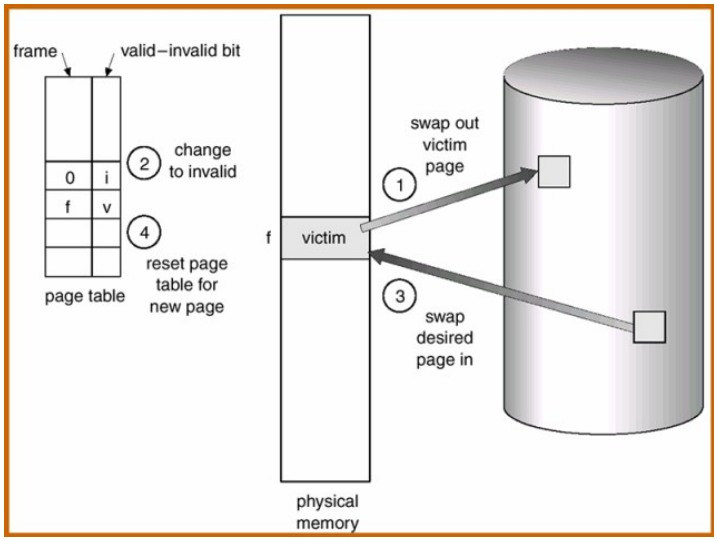
Kumpulan chip yang memetakan alamat maya ke alamat fisik.



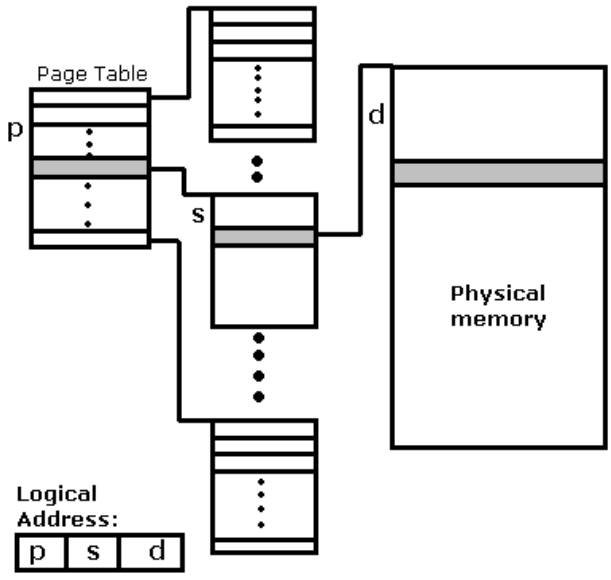
(Gambar : Ilustrasi Paging)

Rutinitas yang dilakukan dalam pemindahan halaman antara lain:

1. Mencari lokasi dari halaman yang diinginkan pada *disk*.
2. Mencari *frame* yang kosong:
   a. Jika ada, maka gunakan *frame* tersebut.
   b. Jika tidak ada, maka tentukan *frame* yang tidak sedang dipakai atau yang tidak akan digunakan dalam jangka waktu lama, lalu kosongkan *frame* tersebut. Gunakan algoritma pemindahan halaman untuk menentukan *frame* yang akan dikosongkan. Usahakan agar tidak menggunakan frame yang akan digunakan dalam waktu dekat. Jika terpaksa, maka sebaiknya segera masukkan kembali frame tersebut agar tidak terjadi *overhead*.
   c. Tulis halaman yang dipilih ke *disk*, ubah tabel halaman dan tabel *frame*.
3. Membaca halaman yang diinginkan ke dalam *frame* kosong yang baru.
4. Mengulangi proses pengguna dari awal.



(Gambar : Rutinitas Paging)

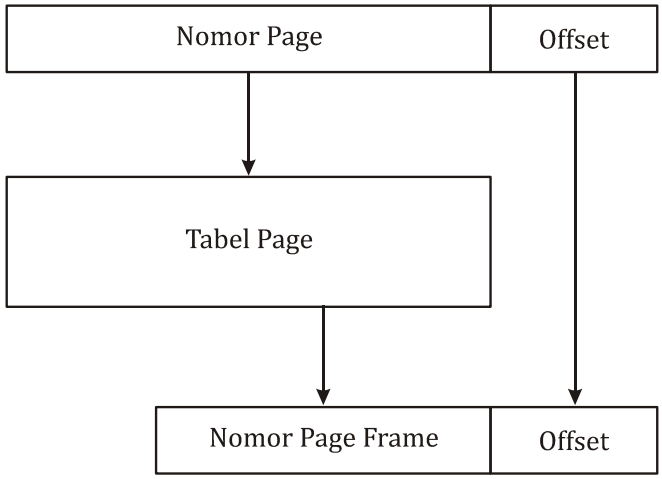


(Gambar : Hirarki Page)

1. Memory Management Unit (MMU)

Berfungsi ------> Pemetaan memori maya ke memori fisik dan menerbitkan *exception* adanya *page fault* yang melewatkan ke sisitem operasi yang menanganinya.

Pada dasarnya MMU terdiri dari tabel halaman yang merupakan sebuah rangkaian *array* dari masukan-masukan (*entries*) yang mempunyai indeks berupa nomor halaman (*p*). Setiap masukan terdiri dari *flags* (contohnya bit sahih dan nomor *frame*). Alamat fisik dibentuk dengan menggabungkan nomor *frame* dengan ofset, yaitu bit paling rendah dari alamat logis.



Komponen Internal MMU

Pemetaan :

- Nomor page maya digunakan sebagai indeks ke tabel page untuk menemukan isian page maya.
- Dari isian tabel page dapat diketahui, apakah page dipetakan ke memori fisik (dengan memeriksa presen/absent bit).
- Apabila alamat terdapat di memori fisik maka isian tabel page memuat nomor page frame. Nomor page frame di tabel page dikopi sebagai bit-bit berorder tinggi di register alamat fisik dan ditambah offset di alamat maya.
- Bila alamat tidak ada di memori fisik maka MMU menerbitkan page fault.

a. Skema Pemetaan
   Misalkan : Ruang alamat maya adalah V = {0,1, … , v-1}
   Ruang alamat fisik adalah M = {0,1, …, m-1}

   Umumnya : ruang alamat maya > ruang alamat fisik (v > m)
   MMU melakukan mekanisme translasi alamat mengasosiasikan alamat maya ke alamat fisik. MMU merealisasikan fungsi f : V -> M, yaitu :

   f (x) = {r, jika item x terdapat di memori fisik dengan lokasi d r}
   {page fault jika item x tidak terdapat pada memori fisik}

   Skenario pemetaan :

	Alamat	Memori Maya	Memori Nyata
0 – 4095	0 – 4K	7	0
4096 – 8191	4 – 8K	5	1
8192 – 12287	8 – 12K	X	2
12288 - 16383	12 – 16K	X	3
16384 – 20479	16 – 20K	0	4
20480 – 24575	20 – 24K	1	5
24576 – 28671	24 – 28K	6	6
28672 – 32767	28 – 32K	3	7
32768 – 36863	32 – 36K	X	
36864 – 40959	36 – 40K	4	
40960 – 45055	40 – 44K	X	
45056 – 49151	44 – 48K	X	
49152 – 53247	48 – 52K	X	
53248 – 57343	52 – 56K	2	
57344 – 61439	56 – 60K	X	
61440 – 65535	60 – 64K	X	

Contoh instruksi : **MOV REG, 0x08**

- Alamat maya 8 dikirim ke MMU
- MMU mengetahui alamat 8 di page 0 (page 0 memuat alamat maya 0-4095)
- Dari tabel, page 0 dipetakan ke frame 7 (page 7 adalah alamat fisik 28672-32768)
- MMU mentransformasikan alamat 8 sebagai (28672+8=28680)
- MMU mengeluarkan alamat 28680 ke bus

b. Implementasi Pemetaan
Komponen MMU, yaitu :

- *Register* Alamat Maya
  - Menyimpan alamat maya yang diacu
  - Nilai di register alamat maya dibagi dua :
    - Bit berorder tinggi menyatakan nomor page maya
    - Bit-bit sisa adalah offset alamat maya

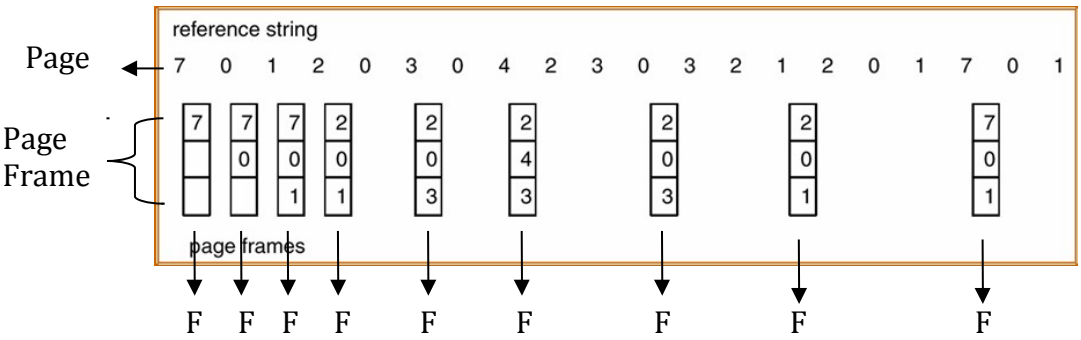
- Tabel *Page*
  Tiap elemen tabel berisi informasi :
  - Present/absent bit
    (bernilai 1 jika page di memori fisik, bernilai 0 jika tidak)
  - Nomor page frame
    Berisi nomor page frame dimana page berada

- *Register* Alamat Fisik
  Menyimpan alamat fisik yang disinyalkan ke bus.
  Nilai di register alamat fisik dibagi menjadi :
  - Bit berorder tinggi menyatakan nomor page frame
  - Bit-bit sisa adalah offset alamat frame

2. Penggantian Page

a. Algoritma penggantian page acak (Random)
   - ✓ Setiap terjadi page fault, penggantian page dipilih secara acak.
   - ✓ Tidak memakai informasi apapun untuk menentukan page yang akan diganti.
   - ✓ Semua page di memori utama dianggap memiliki bobot yang sama.
   - ✓ Dapat memilih sembarang page termasuk page yang sedang diacu.

b. Algoritma penggantian page optimal
   - ✓ Memilih page yang berpeluang dipakai kembali di masa datang yang paling kecil. (Memprediksi/melihat page berikutnya yang tidak dipakai)
   - ✓ Strategi ini menghasilkan jumlah page fault sedikit tapi tidak mungkin diterapkan.



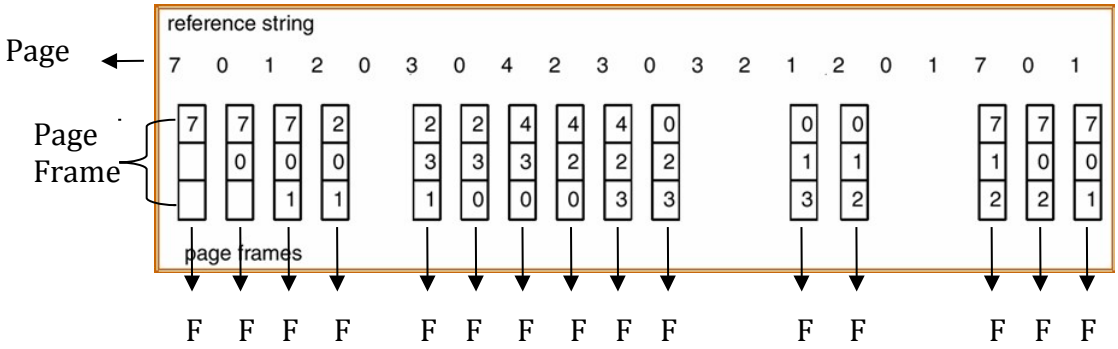
Penjelasan :
- ➢ Algoritma dengan 7 page sebagai string pengacuan (page 5 dan 6 dinyatakan invalid atau memiliki present/absent bit 0)
- ➢ Memiliki 3 page frame dan page fault.
- ➢ Page 7 diacu/ditempatkan di frame : Fault (F)
- ➢ Page 0 diacu/ditempatkan di frame : Fault (F)
- ➢ Page 1 diacu/ditempatkan di frame : Fault (F)
- ➢ Page 2 diacu/ditempatkan di frame dengan mengganti page 7 : (F)
- → Page 7 diganti karena page berikutnya adalah page 0 dan page 3, dan page 7 hanya sedikit (1x) pada antrian page berikutnya berikutnya.
- ➢ Page 0 diacu/ditempatkan di frame tanpa mengganti page.
- ➢ Page 3 diacu/ditempatkan di frame dengan mengganti page 1 : (F)
- ➢ Page 0 diacu/ditempatkan di frame tanpa mengganti page.
- ➢ Page 4 diacu/ditempatkan di frame dengan mengganti page 0 : (F)
- → Page 0 diganti karena page berikutnya adalah page 2 dan page 3.
- ➢ Page 2 diacu/ditempatkan di frame tanpa mengganti page.
- ➢ Page 3 diacu/ditempatkan di frame tanpa mengganti page.
- ➢ Page 0 diacu/ditempatkan di frame dengan mengganti page 4 : (F)
- → Page 4 diganti karena page berikutnya adalah page 3 dan page 2, bahkan page 4 sudah tidak dipakai kembali.
- ➢ Page 3 dan page 2 diacu/ditempatkan di frame tanpa mengganti page.
- ➢ Page 1 diacu/ditempatkan di frame dengan mengganti page 3 : (F)
- ➢ dan seterusnya… hingga urutan page selesai

c. Algoritma penggantian page NRU

Page diberi dua bit mencatat status page :

- Bit R (referenced) menyatakan page sedang diacu
  Bit R = 0 , page sedang diacu
  Bit R = 1, page tidak sedang diacu
- Bit M (Modified) menyatakan page telah dimodifikasi
  Bit M = 0 , page belum dimodifikasi
  Bit M = 1, page telah dimodifikasi

d. Algoritma penggantian page FIFO
Bila terjadi page fault, page elemen terdepan diganti dan page baru ditambahkan di bagian belakang senarai.



Penjelasan :

- Page 7 diacu/ditempatkan di frame : Fault (F)
- Page 0 diacu/ditempatkan di frame : Fault (F)
- Page 1 diacu/ditempatkan di frame : Fault (F)
- Page 2 diacu/ditempatkan di frame dengan mengganti page terdepan dalam senarai yaitu page 7 : Fault (F)
- Page 0 diacu/ditempatkan di frame tanpa mengganti page.
- Page 3 diacu/ditempatkan di frame dengan mengganti page terdepan setelah page 7 dalam senarai yaitu page 0 : Fault (F)
- Page 0 diacu/ditempatkan di frame dengan mengganti page terdepan setelah page 0 dalam senarai yaitu page 1 : Fault (F)
- Page 4 diacu/ditempatkan di frame dengan mengganti page terdepan setelah page 1 dalam senarai yaitu page 2 : Fault (F)
- Page 2 diacu/ditempatkan di frame dengan mengganti page terdepan setelah page 2 dalam senarai yaitu page 3 : Fault (F)

→ Page 0 setelah page 2 tidak diperiksa karena tidak fault

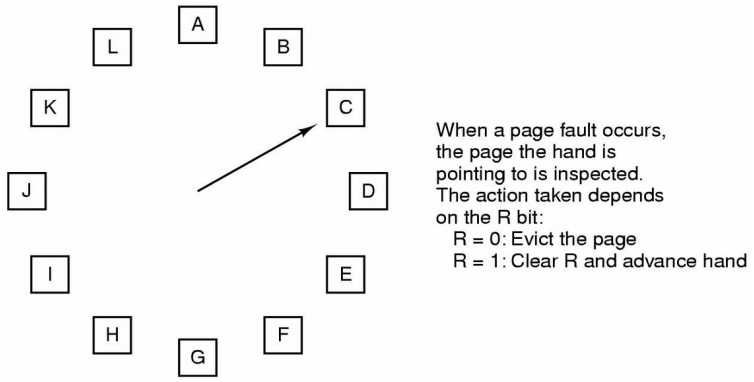
- Page 3 diacu/ditempatkan di frame dengan mengganti page terdepan setelah page 3 dalam senarai yaitu page 0 : Fault (F)
- Page 0 diacu/ditempatkan di frame dengan mengganti page terdepan setelah page 0 dalam senarai yaitu page 4 : Fault (F)
- Page 3 dan page 2 diacu/ditempatkan di frame tanpa mengganti page.
- Page 1 diacu/ditempatkan di frame dengan mengganti page terdepan setelah page 4 dalam senarai yaitu page 2 : Fault (F)
- Page 2 diacu/ditempatkan di frame dengan mengganti page terdepan setelah page 2 dalam senarai yaitu page 3 : Fault (F)
- dan seterusnya… hingga senarai selesai

e. Algoritma penggantian page modifikasi FIFO

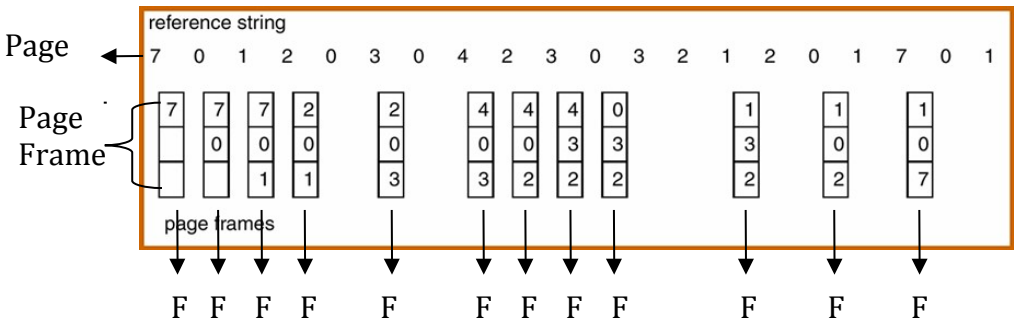
1) Algoritma penggantian page kesempatan kedua
2) Algoritma penggantian clock page

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*
Fault		F	F		F	F		F		F			

(* diacu, > ditunjuk pointer)



f. Algoritma penggantian page LRU
Beberapa instruksi terakhir dari page, kemungkinan masih dipakai. Jika terjadi page fault, maka algoritma mengganti page yang paling lama tidak digunakan.



Penjelasan :
- Page 7 diacu/ditempatkan di frame : Fault (F)
- Page 0 diacu/ditempatkan di frame : Fault (F)
- Page 1 diacu/ditempatkan di frame : Fault (F)
- Page 2 diacu/ditempatkan di frame dengan mengganti page terdepan dalam senarai yaitu page 7 : Fault (F)
- Page 0 diacu/ditempatkan di frame tanpa mengganti page.
- Page 3 diacu/ditempatkan di frame dengan mengganti page yang tidak digunakan pada instruksi sebelumnya yaitu page 1 : Fault (F)

→ Page 0 dan 2 merupakan instruksi sebelumnya yang masih dipakai.

- Page 0 diacu/ditempatkan di frame tanpa mengganti page.
- Page 4 diacu/ditempatkan di frame dengan mengganti page yang tidak digunakan pada instruksi sebelumnya yaitu page 2 : Fault (F)

→ Page 0 dan 3 merupakan instruksi sebelumnya yang masih dipakai.

- Page 2 diacu/ditempatkan di frame dengan mengganti page yang tidak digunakan pada instruksi sebelumnya yaitu page 3 : Fault (F)

→ Page 0 dan 4 merupakan instruksi sebelumnya yang masih dipakai.

- Page 3 diacu/ditempatkan di frame dengan mengganti page 0: Fault (F)

→ Page 2 dan 4 merupakan instruksi sebelumnya yang masih dipakai.

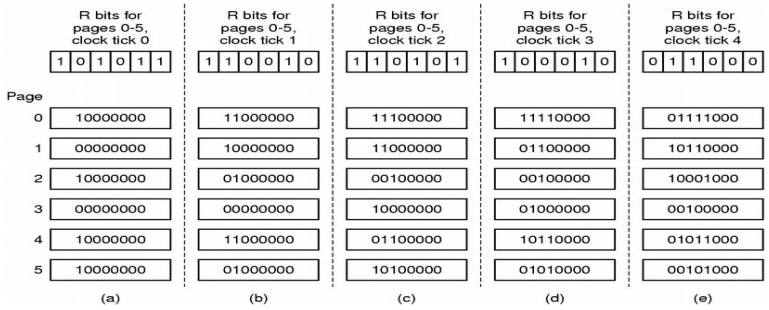
- Page 0 diacu/ditempatkan di frame dengan mengganti page 4: Fault (F)

→ Page 2 dan 3 merupakan instruksi sebelumnya yang masih dipakai

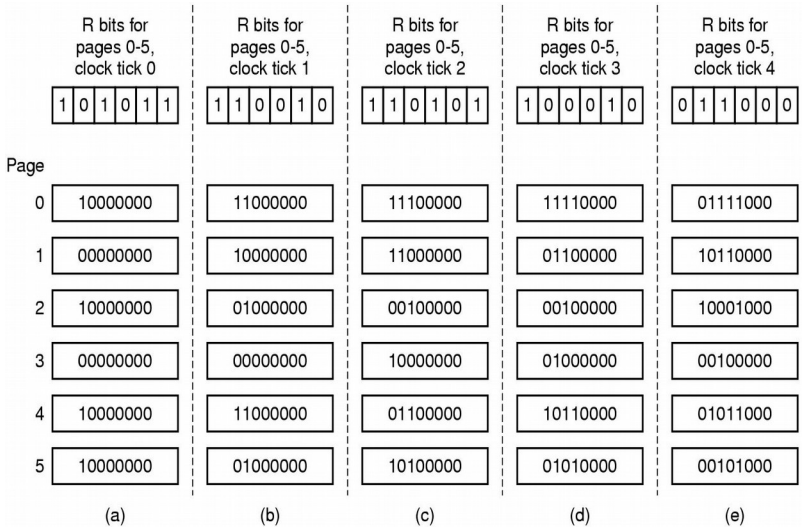
- Page 3 dan page 2 diacu/ditempatkan di frame tanpa mengganti page.
- Page 1 diacu/ditempatkan di frame dengan mengganti page 0: Fault (F)

→ Page 2 dan 3 merupakan instruksi sebelumnya yang masih dipakai

- dan seterusnya... hingga senarai selesai



- The aging algorithm simulates LRU in software
- Note 6 pages for 5 clock ticks, (a) – (e)



- The aging algorithm simulates LRU in software
- Note 6 pages for 5 clock ticks, (a) – (e)

Review Page Replacement Algorithm

Algorithm	Comment
1. Optimal	Tidak dapat diimplementasi tetapi berguna sebagai Benchmark
2. NRU	Tidak optimal
3. FIFO	Sering mengganti Page yang penting
4. Second Chance	Modifikasi dari FIFO
5. Clock	Realistis
6. LRU	Bagus tetapi sulit untuk diimplementasi
7. Aging	Efisien dalam mengimplementasi LRU
8. Working Set	Mahal untuk diimplementasi
9. WSClock	Cukup Efisien

3. Masalah Utama Sistem Paging

a. Working set model

1) Prinsip Lokalitas

Proses-proses cenderung mengacu penyimpanan secara tak seragam, mempunyai pola-pola sangat setempat.

a) Lokalitas berdasarkan waktu

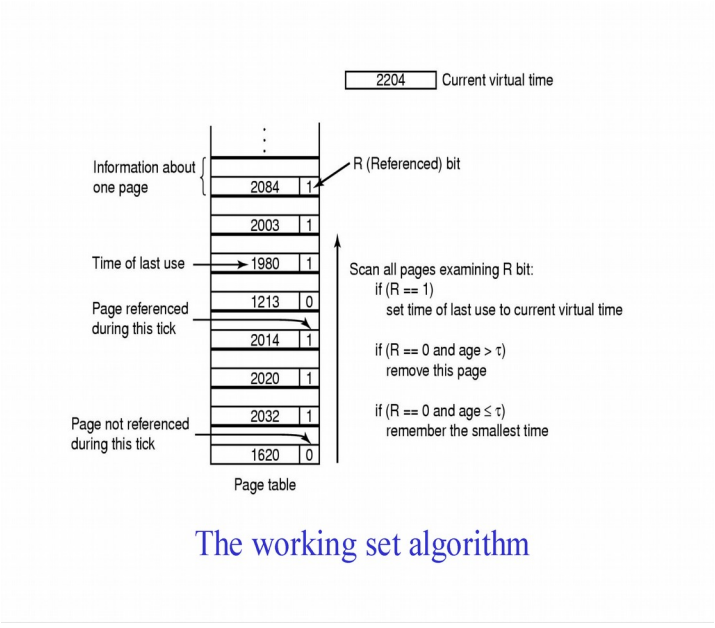
Proses cenderung terkonsentrasi acuannya ke satu interval waktu eksekusi yang dekat.

b) Lokalitas berdasarkan ruang

Proses cenderung terkonsentrasi acuannya ke satu kelompok data yang berdekatan.

2) Working Set of Program Behavior

Kumpulan page proses yang secara aktif diacu yang terlokalisasi pada waktu itu dan harus dijaga berada di memori utama. Perpindahan program dari satu lokalitas ke lokalitas lain saat dieksekusi.



The working set algorithm

3) Prepaging
Teknik memuatkan page-page lebih dulu sebelum proses berjalan.

4) Demand Paging
Teknik yang segera memuatkan page-page begitu page dibutuhkan.

b. Kebijaksanaan penggantian lokal vs global
1) Penggantian lokal yaitu page yang dipilah untuk diganti hanya pada partisi dimana proses diletakkan.
2) Penggantian global yaitu page yang dipilah untuk diganti adalah tempat kosong dengan tidak mempedulikan partisi proses.

c. Frekuensi page fault
- ✓ Sistem operasi mendefinisikan frekuensi page fault kritis sistem (atau per proses)
- ✓ Sistem operasi mengukur waktu proses maya dan menyimpan waktu page fault mutakhir di PCB proses
- ✓ Ketika page fault terjadi, sistem operasi bertindak :
  - Jika page fault terakhir terjadi kurang dari T=1/P ms detik yang lalu, proses beroperasi di atas ambang maka page frame baru ditambahkan.
  - Selain itu, berarti proses beroperasi di bawah ambang PFF, maka page frame page bit R=0 dan W=0 dibebaskan untuk alokasi wage baru proses lain.

d. Ukuran page
- ❖ Ukuran page ditentukan perancang sistem operasi.
- ❖ Ukuran page harus ditentukan agar sistem berperilaku optimal.
- ❖ Penentuan ukuran page memerlukan penilaian dan pemahaman mendalam tentang perangakat keras, perangkat lunak, dan aplikasi sistem.
- ❖ Ukuran page lebih kecil berarti jumlah page dan page frame lebih banyak sehingga memerlukan tabel page lebih besar.
- ❖ Ukuran page besar berarti sejumlah informasi yang tidak diacu juga dimasukkan ke memori utama sehingga terjadi fragmentasi internal yang tinggi.
- ❖ Transfer masukan/keluaran relatif sangat mengkonsumsi waktu sehingga perlu meminimumkan jumlah transfer masukan/keluaran saat program berjalan.
- ❖ Program cenderung mengikuti prinsip lokalitas yang cenderung berukuran kecil.

Nama Sistem Komputer	Ukuran Page (byte)
DEC PDP 10	512
DEC PDP 11	8192
DEC PDP 20	512
DEC VAX 8800	512
Honeywell Multics	1024
IBM 370	2048 atau 4096

IBM 370/XA atau IBM 370/ESA	4096
IBM AS/400	512
Intel 80386	4096
Motorola 68030	Dapat diprogam antara 256- 32768

4. Masalah Implementasi Sistem Paging

a. Back up instruksi yang terakhir dijalankan sebelum terjadi page fault
b. Buffer perangkat masukan/keluaran (penguncian page di memory)
c. Page yang dipakai bersama
d. Backing store
e. Paging daemon
f. Pananganan page fault

C. Segmentasi

Segmentasi merupakan skema manajemen memori yang mendukung cara pandang seorang programmer terhadap memori. Ruang alamat lojik merupakan sekumpulan dari segmen-segmen. Masing-masing segment mempunyai panjang dan nama. Alamat diartikan sebagai nama segmen dan offset dalam suatu segmen. Jadi jika seorang pengguna ingin menunjuk sebuah alamat dapat dilakukan dengan menunjuk nama segmen dan offsetnya. Untuk lebih menyederhanakan implementasi, segmen-segmen diberi nomor yang digunakan sebagai pengganti nama segment. Sehingga, alamat lojik terdiri dari dua tupple: [segment-number, offset].

Secara sederhana segmentasi bisa diartikan sebagai suatu **ruang alamat** atau **segment yang berada di memori**. Segment-segment itu dalam keadaan independent. Setiap segment berisi alamat 0 sampai maksimum secara linier. Panjang setiap segment berbeda-beda sampai panjang maksimun, perobahan panjang segment terjadi selama proses eksekusi. Segment stack **bertambah** ketika terjadi **operasi push** dan turun saat **operasi pop**, dimana setiap segment merupakan ruang alamat terpisah segment-segment dapat tumbuh dan mengkerut secara bebas tanpa mempengaruhi yang lain.

Alamat terdiri dari dua bagian pada memori bersegment yaitu :

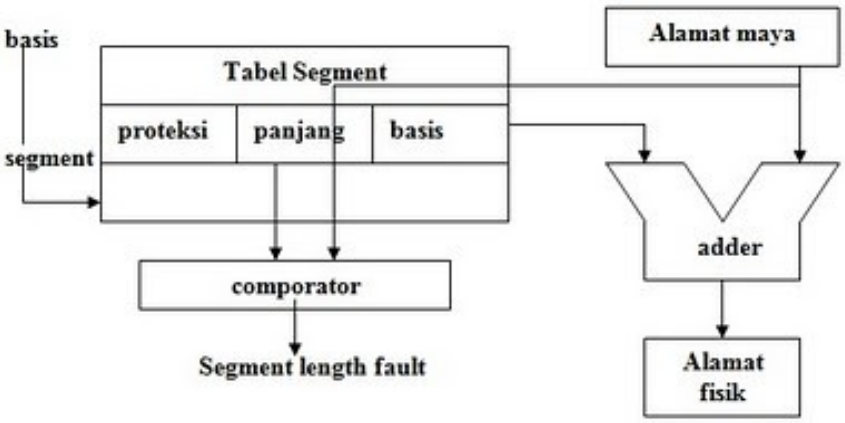
- Nomor segment
- Alamat pada segment (offset).

Segment dapat berisi :

- Prosedure
- Array
- Stack
- Kumpulan variable skala.

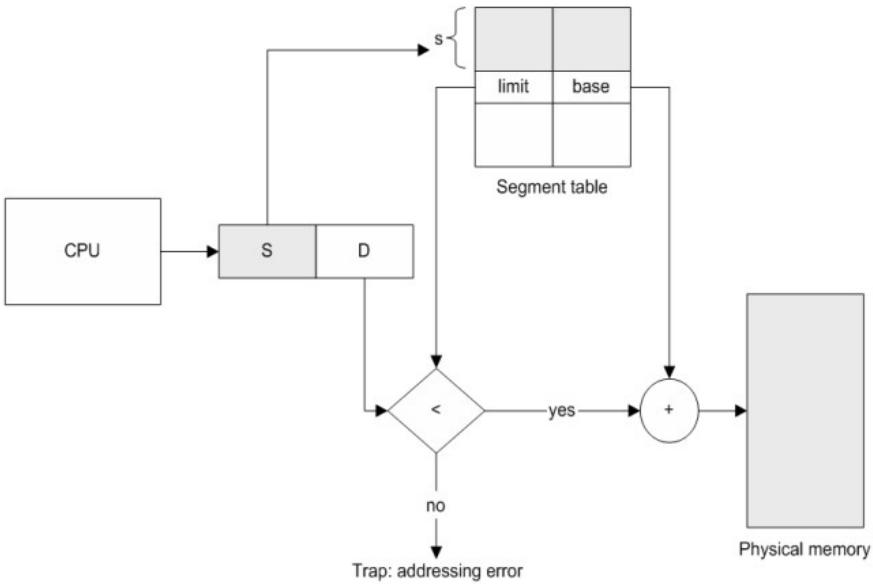
Sistem Segmentasi

Sistem dengan memori maya dengan segmentasi murni adalah alamat maya adalah offset di segment, setiap proses mempunyai tabel segment dan pada saat proses running alamat awal maya tabel dimuatkan ke register dasar. Nomor segment digunakan mencari deskriptor segment di tabel segment yang menyediakan alamat fisik awal dari segment, panjang dan bit-bit proteksinya. Alamat fisik dihitung dengan menambahkan alamat dasar segment ke alamat maya.



Gambar 4. Skema Segmentasi

Ilustrasi penggunaan segmen dapat dilihat pada Gambar "Arsitektur Segmentasi". Suatu alamat lojik terdiri dari dua bagian, yaitu nomor segmen(s), dan offset pada segmen(d). Nomor segmen digunakan sebagai indeks dalam segmen table. Offset d alamat lojik harus antara 0 hingga dengan segmen limit. Jika tidak maka diberikan pada sistem operasi. Jika offset ini legal maka akan dijumlahkan dengan segmen base untuk menjadikannya suatu alamat di memori fisik dari byte yang diinginkan. Jadi segmen table ini merupakan suatu array dari pasangan base dan limit register.

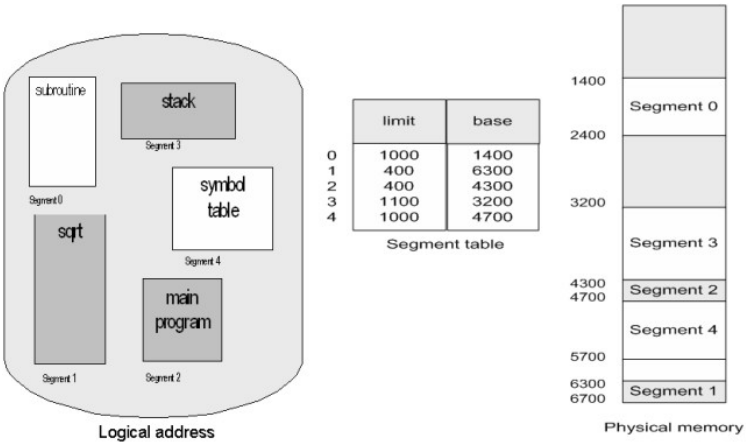


Gambar : Arsitektur Segmentasi

Sebagai contoh, kita mempunyai nomor segmen dari 0 sampai dengan 4. Segmen-segmen ini disimpan dalam suatu memori fisik. Tabel segmen berisi data untuk masing-masing segmen, yang memberikan informasi tentang awal alamat dari segmen di fisik memori (atau base) dan panjang dari segmen (atau limit). Misalkan, segmen 2 mempunyai panjang 400 dan dimulai pada lokasi 4300. Jadi, referensi di byte 53 dari segmen 2 dipetakan ke lokasi 4300 + 53 = 4353. Suatu referensi ke segmen 3, byte 852, dipetakan ke 3200 (sebagai base dari segmen) +

852 = 4052. Referensi ke byte 1222 dari segmen 0 akan menghasilkan suatu trap ke sistem operasi, karena segmen ini hanya mempunyai panjang 1000 byte.

Segmen-segmen dapat berukuranberbeda dan dinamis. Pengacuan-pengacuan memori berbentuk (nomor segmen,offset).



Saling Berbagi dan Proteksi

Segmen dapat terbagi jika terdapat elemen di tabel segmen yang berasal dari dua proses yang berbeda yang menunjuk pada alamat fisik yang sama. Saling berbagi ini muncul di level segmen dan pada saat ini terjadi semua informasidapat turut terbagi. Proteksi dapat terjadi karena ada bit-proteksi yang berhubungan dengan setiap elemen dari segmen tabel. Bit-proteksi ini berguna untuk mencegah akses ilegal ke memori. Caranya: menempatkan sebuah array di dalam segmen itu sehingga perangkat keras manajemen memori secara otomatis akan mengecek indeks array -nya legal atau tidak.

Segmentasi dengan Pemberian Halaman

Kelebihan Pemberian Halaman : tidak ada fragmentasi luar - alokasinya cepat.
Kelebihan Segmentasi : saling berbagi - proteksi.

Metode segmentasi dan paging masing-masing memiliki keuntungan dan kerugian. Selain kedua metode itu ada metode pengaturan memori lain yang berusaha menggabungkan metode segmentasi dan paging. Metode ini disebut dengan segmentation with paging. Dengan metode ini jika ukuran segmen melebihi ukuran memori utama maka segmen tersebut dibagi-bagi jadi ukuran-ukuran halaman yang sama ==> paging.

Keuntungan segmentasi :

1. Menyederhanakan penanganan struktur data yang berkembang. Seringkali penanganan struktur data menuntut perubahan panjang data. Hal ini dimungkinkan dengan adanya segmentasi. Jadi dengan segmentasi membuat penanganan struktur data menjadi fleksibel.

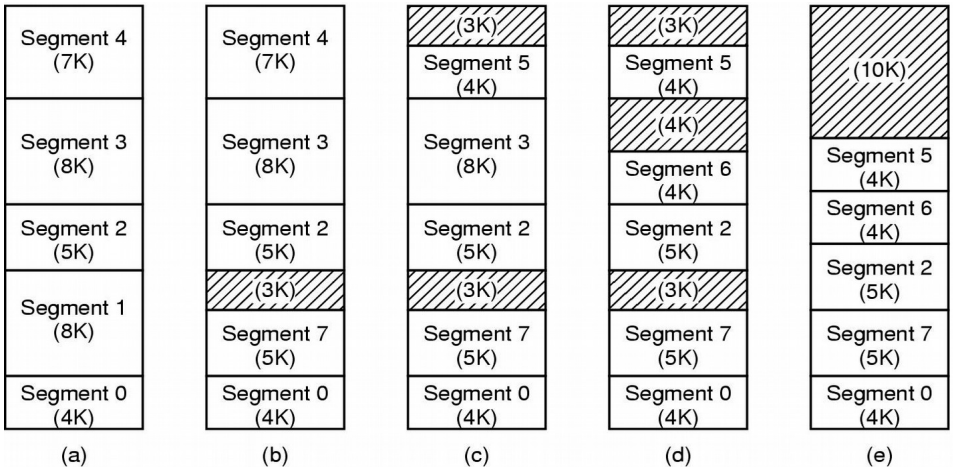
2. Kompilasi ulang independen tanpa mentautkan kembali seluruh program. Teknik ini memungkinkan program-program dikompilasi ulang secara independen tanpa perlu mentautkan kembali seluruh program dan dimuatkan kembali. Jika masing-masing prosedur terdapat di segmen terpisah beralamat 0

sebagai alamat awal, maka pentautan prosedur-prosedur yang dikompilasi secara terpisah sangat lebih mudah. Setelah semua prosedur dikompilasi dan ditautkan, panggilan ke prosedur di segmen n akan menggunakan alamat dua bagian yaitu (n,0) mengacu ke word alamat 0 (sebagai titik masuk) segmen ke n. Jika prosedur di segmen n dimodifikasi dan dikompilasi ulang, prosedur lain tidak perlu diubah (karena tidak ada modifikasi alamat awal) walau versi baru lebih besar dibanding versi lama.

3. Memudahkan pemakaian memori bersama di antara proses-proses. Pemrogram dapat menempatkan program utilitas atau tabel data berguna di segmen yang dapat diacu oleh proses-proses lain. Segmentasi memberi fasilitas pemakaian bersama terhadap prosedur dan data untuk dapat diproses, berupa shared library. Pada workstation modern yang menjalankan sistem Windows sering mempunyai pustaka grafis sangat besar. Pustaka ini diacu hampir semua program. Pada sistem bersegmen, pustaka grafis diletakan di satu segmen dan dipakai secara bersama banyak proses sehingga menghilangkan mempunyai pustaka di tiap ruang alamat proses.

4. Memudahkan proteksi karena segmen dapat dikonstruksi berisi sekumpulan prosedur atau data terdefinisi baik, pemrogram atau administrator sistem dapat memberikan kewenangan pengaksesan secara nyaman.

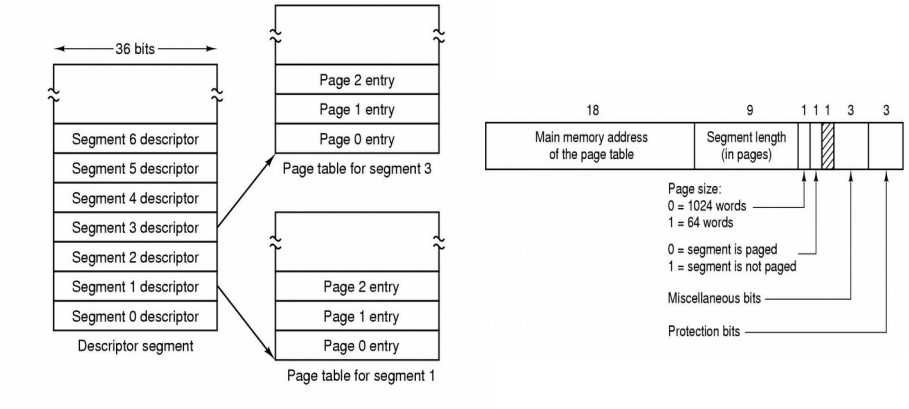
Implementasi segmentasi Murni



(a)-(d) Development of checkerboarding
(e) Removal of the checkerboarding by compaction

Implementasi Segmentasi dengan Paging :

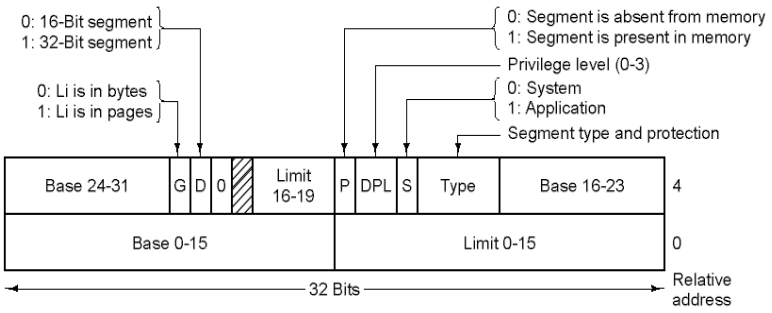
MULTICS



- Descriptor segment points to page tables
- Segment descriptor – numbers are field lengths

Implementasi Segmentasi dengan Paging :

INTEL PENTIUM



- Pentium code segment descriptor
- Data segments differ slightly

Perbedaan Segmentasi dan Paging

Ada beberapa perbedaan antara Segmentasi dan Paging diantaranya adalah:

1. Segmentasi melibatkan programer (programer perlu tahu teknik yang digunakan), sedangkan dengan paging, programer tidak perlu tahu teknik yang digunakan.

2. Pada segmentasi kompilasi dilakukan secara terpisah sedangkan pada paging, kompilasinya tidak terpisah.

3. Pada segmentasi proteksinya terpisah sedangkan pada paging proteksinya tidak terpisah.

4. Pada segmentasi ada shared code sedangkan pada paging tidak ada shared code.

5. Pada segmentasi terdapat banyak ruang alamat linier sedangkan pada paging hanya terdapat satu ruang alamat linier.

6. Pada segmentasi prosedur dan data dapat dibedakan dan diproteksi terpisah sedangkan pada paging prosedur dan data tidak dapat dibedakan dan diproteksi terpisah.

7. Pada segmentasi pengubahan ukuran tabel dapat dilakukan dengan mudah sedangkan pada Paging pengubahan ukuran tabel tidak dapat dilakukan dengan mudah.

8. Segmentasi digunakan untuk mengizinkan program dan data dapat dipecahkan jadi ruang alamat mandiri dan juga untuk mendukung sharing dan proteksi sedangkan paging digunakan untuk mendapatkan ruang alamat linier yang besar tanpa perlu membeli memori fisik lebih.

D. Kombinasi Sistem Paging dan Segmentasi

- Pada kombinasi paging dan segmentasi, ruang alamat pemakai dibagi menjadi sejumlah segmen sesuai kehendak pemrogram.
- Tiap segmen dibagi menjadi sejumlah page berukuran tetap, berukuran sama dengan page frame memori utama.
- Jika segmen kurang dari ukuran page, segmen hanya memerlukan satu page.

1. Memori Maya dengan Segmentasi Murni
   Perangkat keras memberikan pengacuan memori suatu segmen tertentu. Pilihan segmen dapat dibuat dengan sembarang kombinasi berikut :
   a. Instruksi
   b. Target dari suatu alamat (program atau data)
   c. Status saat itu (proses, sistem, interupsi)

   - ✓ Alamat maya adalah offset di segmen.
   - ✓ Tiap proses mempunyai tabel segmen.
   - ✓ Ketika proses running, alamat awal tabel dimuatkan ke register dasar.
   - ✓ Nomor segmen digunakan untuk mencari deskriptor segmen di tabel segmen yang menyediakan alamat fisik awal dari segmen, panjang, dan bit-bit proteksinya.
   - ✓ Alamat fisik dihutung dengan menambahkan alamat dasar segmen ke alamat maya.
   - ✓ ***Contoh sistem :*** Intel 80386
2. One Level Paging
   - ✓ Nomor page maya digunakan sebagai indeks ke tabel page yang biasanya berlokasi di memori utama.

- ✓ Isian tabel page berisi nomor page fisik dan bit-bit proteksi.
- ✓ Offset pada page fisik sama dengan offset pada page maya.
- ✓ Register panjang (Length register) digunakan untuk menetapkan akhir tabel page untuk menghindari ruang yang akan disiakan untuk isian-isian tak berguna.
- ✓ ***Contoh Sistem :*** DEC PDP-11, DEC VAX, Data General Eclipse, Motorola MC86030

3. Two Level Paging
   - ✓ Indeks 1 ditambahkan ke alamat root atau dasar tabel segmen untuk memperoleh alamat isian tabel segmen (Segment Tabel Entry-STE)
   - ✓ STE dibaca dari memori dan alamat dasarnya ditambah indeks 2 untuk memperoleh alamat isian tabel page (Page Tabel Entry).
   - ✓ PTE dibaca dari memori untuk memperoleh nomor page fisik. Nomor page fisik ini ditambah offset di alamat maya untuk memperoleh alamat fisik akhir.
   - ✓ ***Contoh Sistem :*** DEC VAX, IBM S/370, Data General Eclipse 32 bit, Motorola MC68030, Intel 80386.

4. Three Level Paging
   - ✓ Translasi nomor page maya ke nomor page fisik memerlukan tiga tahap.
   - ✓ Tiap fields indeks ditambahkan alamat dasar tabel yang berkorespondensi untuk menemukan isian tabel berikutnya.
   - ✓ Isian tabel menyediakan bit-bit proteksi dan alamat dasar tabel berikutnya.
   - ✓ ***Contoh Sistem :*** SUN SPARC

5. Four Level Paging
   - ✓ Translasi nomor page maya ke nomor page fisik memerlukan empat tahap.
   - ✓ Tiap fields indeks ditambahkan alamat dasar tabel yang berkorespondensi untuk menemukan isian tabel berikutnya.
   - ✓ Isian tabel menyediakan bit-bit proteksi dan alamat dasar tabel berikutnya.
   - ✓ ***Contoh Sistem :*** Motorola MC68030

KUIS :

1. **Jelaskan Implementasi Pemetaan Komponen MMU?**
2. **Jelaskan perbedaan pengelolaan memori Paging dan Segmentasi?**
3. **Jelaskan bagaimana cara kerja Algoritma Random, Optimal, FIFO dan LRU?**