Konsep Dasar Memori

Heri Kurniawan

OS-Gasal 2009/2010

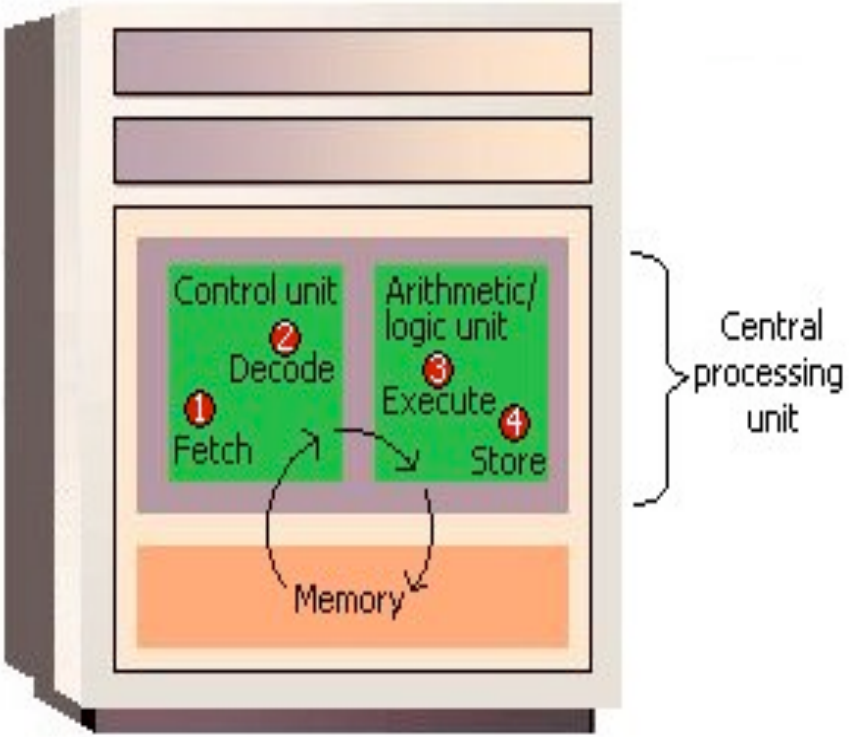
Tujuan Pembelajaran

- Memahami proses penerjemahan alamat memori
- Memahami langkah proteksi sistem operasi terhadap akses memori
- Memahami hubungan CPU dengan memori

Pendahuluan

- CPU membutuhkan memory untuk menempatkan program dan data saat eksekusi.
- Memory terdiri dari kumpulan larik byte dan word dalam jumlah besar, masing-masing memiliki alamat yang berbeda
- Memori dan register diakses langsung oleh CPU
- Jika data masih berada dalam disk, maka data harus dikirim dahulu ke memory, sebelum dapat diakses oleh CPU

Hubungan CPU dengan Memory



Sumber : http://homepage.cs.uri.edu

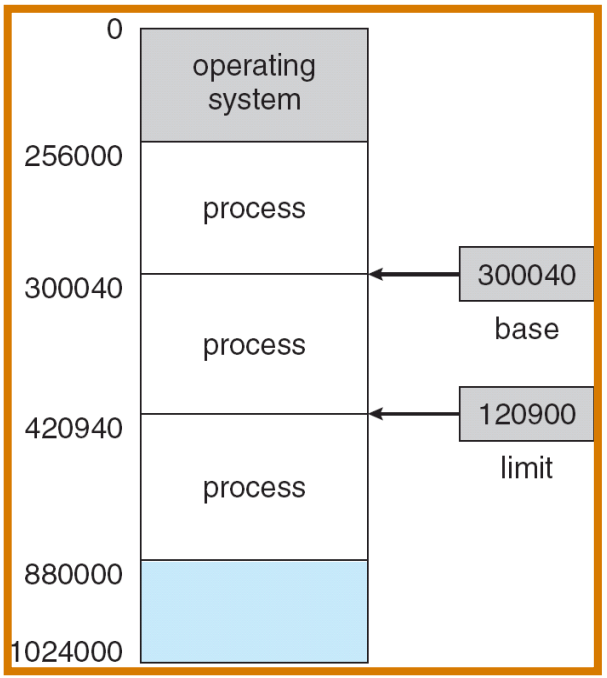
1. *Control Unit (CU) mengambil intruksi dari memori*
2. *CU menginterpretasi instruksi dan memindahkan data ke Arithmetic/ Logic Unit (ALU)*
3. *ALU mengeksekusi instruksi dan data, berdasarkan persepsi CU*
4. *ALU menyimpan hasil eksekusi ke memory data register kemudian dikirim ke memori*

Pendahuluan

- Register diakses dengan cepat oleh CPU dalam 1 clock cycle CPU
- Memory lebih lambat (aksesnya > 1 clock cycle CPU), transfer instruksi/data melalui bus.
- Untuk mengatasi perbedaan kecepatan, maka memory buffer digunakan (disebut juga dengan **cache**)
- Untuk menjaga integritas dan keamanan, setiap proses mempunyai lokasi memory yang berbeda.

Pendahuluan

Proteksi dengan menggunakan hardware
Base Register : Start address
Limit Register : Range address



base register = 300040
limit register =120900

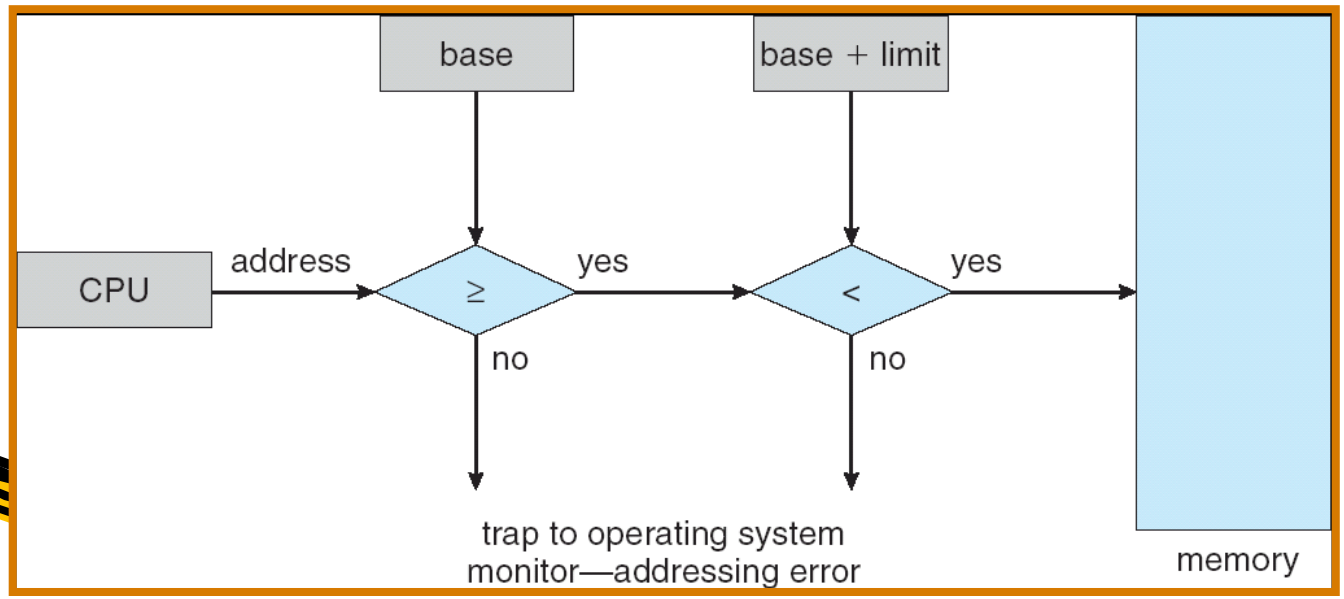
base register + limit register < 420940

Program hanya bisa mengakses alamat memori antara 300040 hingga 420939

- Sistem operasi meload nilai register base dan limit dengan instruksi khusus. Instruksi dieksekusi dalam mode kernel

Pendahuluan

- CPU membandingkan alamat yang dihasilkan pada user mode dengan register.
- Setiap ada percobaan untuk mengakses memory sistem operasi atau memori user lain, maka sinyal trap muncul dan dikirimkan ke sistem operasi.



Pengalamatan

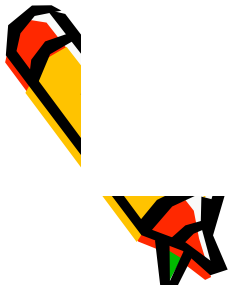
- Program yang ada dalam disk tidak dapat dieksekusi jika tidak pindahkan dulu ke memory
- Umumnya antrian proses dalam disk dipilih untuk dipindahkan ke memori agar dapat dieksekusi CPU.
- Proses user dapat ditempatkan pada sembarang lokasi pada memori.
- Walaupun alamat memory fisik dimulai dari 00000, namun alamat awal memori proses user tidak harus dari 00000

Pengalamatan

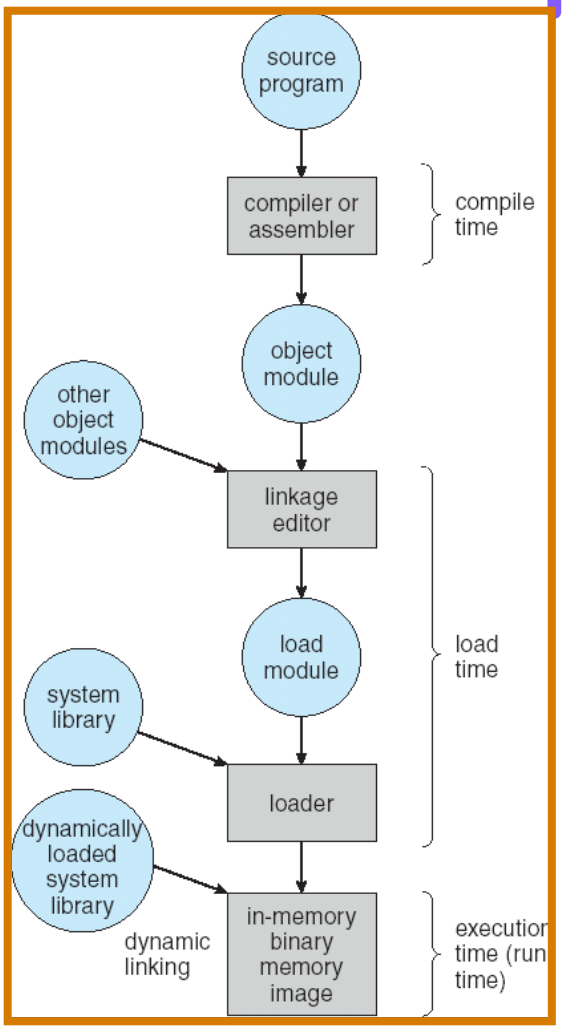
- Proses pengalamatan instruksi dan data ke alamat memori dapat terjadi pada tiga fase:
  - Compile Time
    - Lokasi alamat fisik proses pada memory telah diketahui, maka compiler mengeluarkan *absolute address* (alamat fisik)
    - Jika awal alamat memori berubah, kompilasi ulang kode dilakukan kembali

Pengalamatan

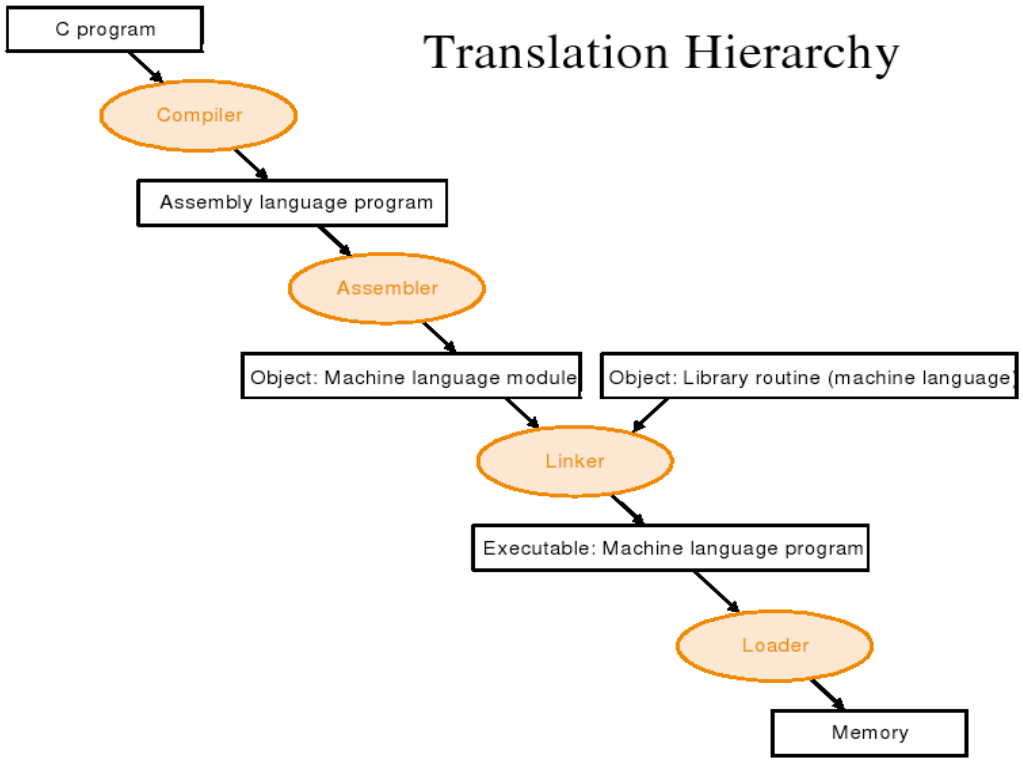
- Load Time
  - Jika alamat fisik memori tidak diketahui saat *compile time*, maka compiler memberikan *relocatable address*. Pengalamatan akhir diundur hingga *load time*. Saat load time relocatable address diubah menjadi absolute address.
- Execution Time
  - Jika terjadi swap-out dan swap-in proses, maka proses pengalamatan diundur hingga *run time*.
  - Butuh dukungan hardware untuk melakukan pemetaan alamat pada fase ini.



Pengalamatan



- Fase proses program user

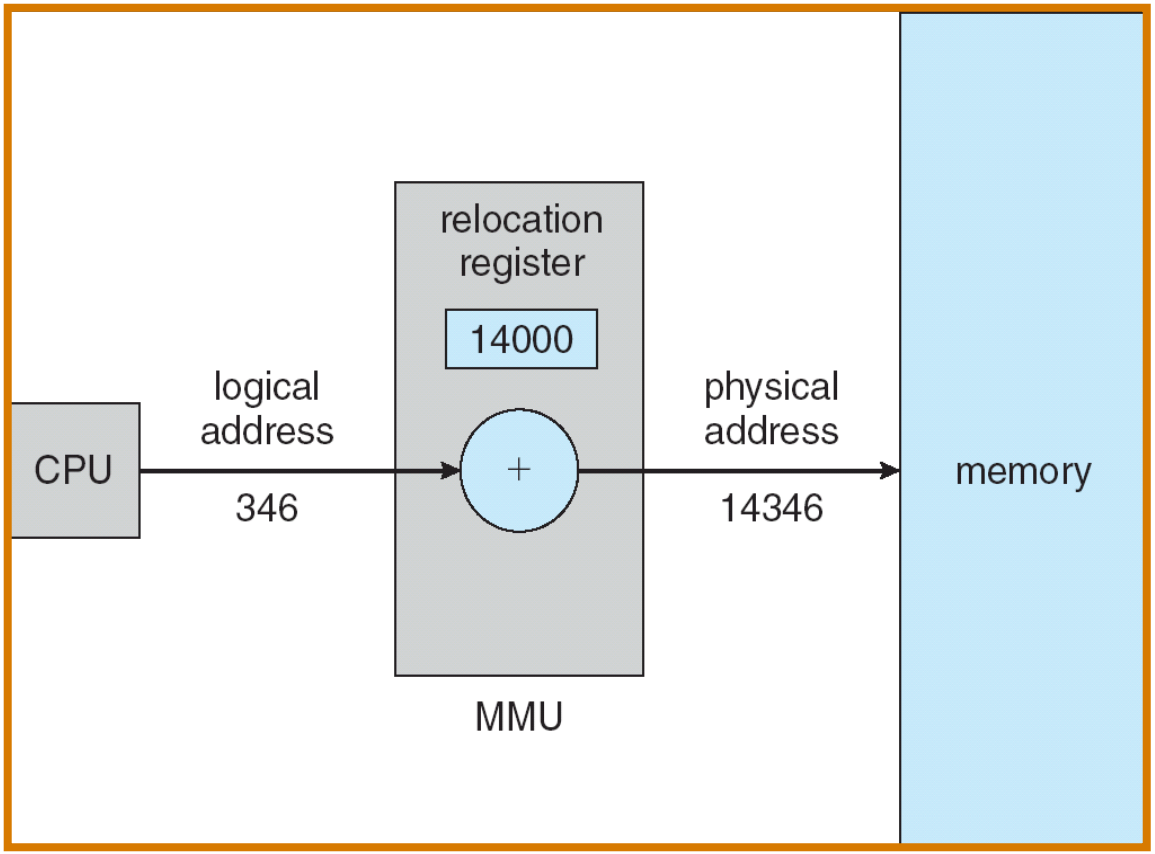


Fase penerjemahan program user secara hirarki

Ruang alamat logika dan fisik

- Alamat logika (alamat virtual) ⇒ alamat yang dihasilkan CPU
- Alamat fisik ⇒ alamat pada unit memory

Ruang alamat logika dan fisik



Relokasi dinamis dengan menggunakan relocation register

Memory-Management Unit (MMU)

- Perangkat keras yang memetakan alamat virtual ke alamat fisik
- MMU memetakan dengan cara menambahkan nilai pada *relocation register* (*base register*) ke setiap alamat yang dihasilkan pada proses user
  - Jika base register=14000, setiap akses user ke lokasi 0 akan dipetakan ke lokasi 14000. Akses user ke 346 dipetakan ke 14346
- Program user hanya mengetahui *alamat virtual*, namun tidak *alamat fisik*

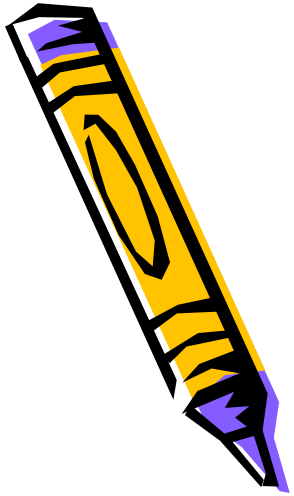
Memory-Management Unit (MMU)

- Alamat Virtual : 0 – max
- Alamat Fisik: base R+0 – base R+max

Dynamic Loading

- Kapasitas memori terbatas sehingga routine hanya di load jika dibutuhkan
- Dynamic loading digunakan untuk meningkatkan utilisasi memori
- Keuntungan : routine yang tidak pernah digunakan, tidak diload ke memori
- Dynamic loading tidak membutuhkan dukungan khusus dari sistem operasi
- Metode dynamic loading didesain dalam program user.

Dynamic Linking



- Proses linking terjadi saat execution time
- Tanpa dynamic linking, system library seperti *subroutine library* dicopy ke executable image
- Pencarian routine dalam memory menggunakan *stub* yang terdapat pada *image* aplikasi.
- Membutuhkan dukungan sistem operasi

Pertanyaan Ulasan

- Apa yang dimaksud dengan Dynamic Loading
- Apa yang dimaksud dengan Dynamic Linking
- Apa perbedaan antara dynamic linking dengan dynamic loading
- Apa yang dimaksud base dan limit register? Apa manfaatnya?
- Apa yang dimaksud dengan MMU?