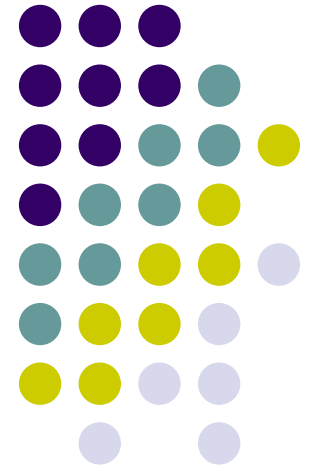


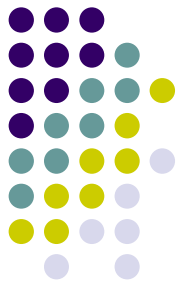
2

Struktur Sistem Komputer



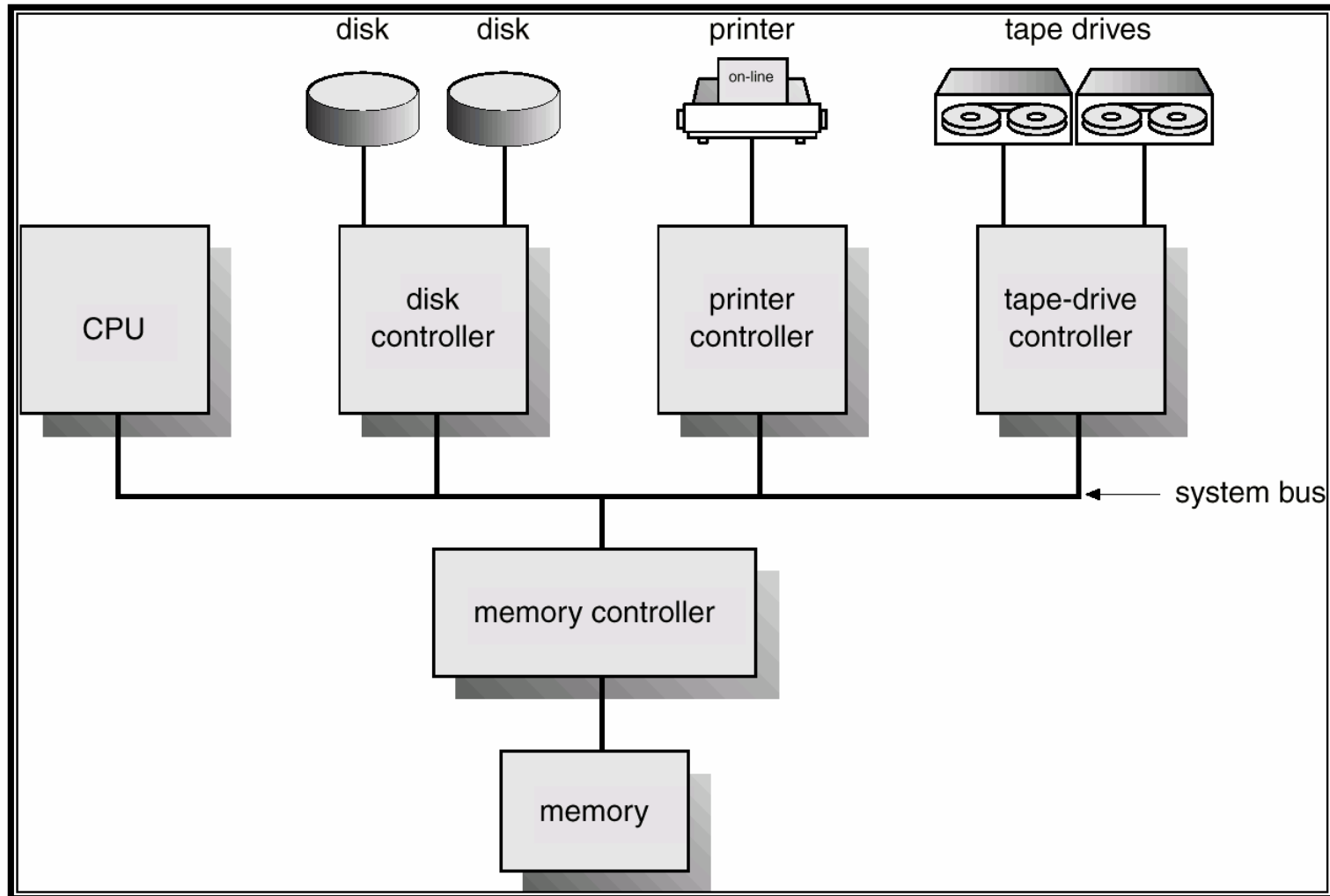
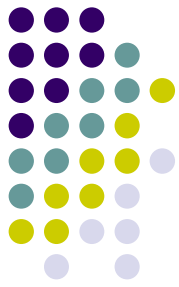
Review:

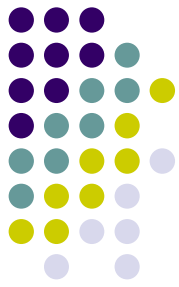
Struktur Sistem Komputer



- Operasi Sistem Komputer
- Struktur I/O
- Struktur Storage
- Proteksi Hardware

Arsitektur Sistem Komputer

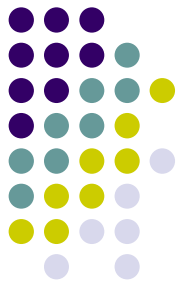




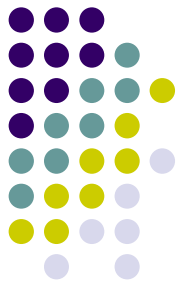
* Operasi Sistem Komputer

- CPU devices dan I/O dapat beroperasi secara serentak (concurrent)
 - Efisiensi pemakaian CPU
- Semua request ke I/O dikendalikan oleh I/O systems:
 - Setiap device terdapat controller yang mengendalikan device tertentu, misalkan video display => video card, disk => disk controller.
 - Setiap device controller mempunyai local buffer.
- CPU memindahkan data dari/ke memory ke/dari local buffer.
 - Setelah itu controller akan mengirimkan data dari buffer ke device.
- Bagaimana mekanisme I/O supaya CPU dapat melakukan switch dari satu job ke job lain?

Operasi Sistem Komputer (Cont.)

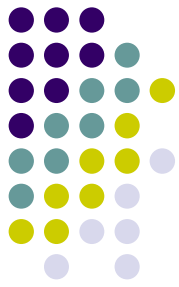


- Ilustrasi:
 - Instruksi CPU dalam orde: beberapa mikro-detik
 - Operasi read/write dari disk: 10 – 15 mili-detik
 - Ratio: CPU ribuan kali lebih cepat dari operasi I/O
 - Jika CPU harus menunggu (idle) sampai data transfer selesai, maka utilisasi CPU sangat rendah (lebih kecil 1%).
- Solusi: operasi CPU dan I/O harus overlap
 - Concurrent: CPU dapat menjalankan beberapa I/O device sekaligus
 - CPU tidak menunggu sampai operasi I/O selesai tapi melanjutkan tugas yang lain
 - Bagaimana CPU mengetahui I/O telah selesai?



Programmed I/O (1)

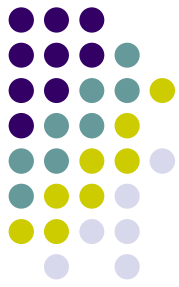
- Programmed I/O
 - Mekanisme CPU yang bertanggung jawab memindahkan data dari/ke memori ke/dari controller
- CPU bertanggung jawab untuk jenis operasi I/O
 - Transfer data dari/ke buffer
- Controller melakukan detail operasi I/O
 - Jika telah selesai memberikan informasi ke CPU => flag
- Bagaimana CPU mengetahui operasi telah selesai?
 - Apakah menguji flag? Seberapa sering?



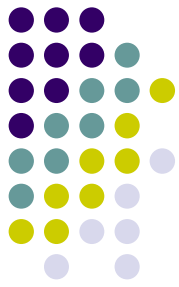
Programmed I/O (2)

- CPU harus mengetahui jika I/O telah selesai => hardware flag (controller)
- Polling: CPU secara periodik menguji flag (true or false)
 - Menggunakan instruksi khusus untuk menguji flag
 - Masalah: seberapa sering? “wasted CPU time !”? Antar I/O device berbeda “speed”!
- Interrupt:
 - Bantuan hardware – melakukan interupsi pada CPU jika flag tersebut telah di-set (operasi I/O telah selesai)

Interrupt

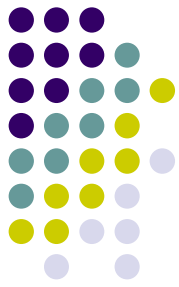


- Interrupt:
 - CPU transfer control ke “interrupt service routine”, => address dari service routine yang diperlukan untuk device tsb.
 - **Interrupt handler**: menentukan aksi/service yang diperlukan
- Struktur interrupt harus menyimpan address dari instruksi yang sedang dikerjakan oleh CPU (interrupted).
 - CPU dapat resume ke lokasi tersebut jika service routine telah selesai dikerjakan
- Selama CPU melakukan service interrupt, maka interrupt selanjutnya tidak akan dilayani “disabled”, karena CPU tidak dapat melayani interrupt (lost).
- Pengoperasian sistem tersebut menggunakan *interrupt driven*.



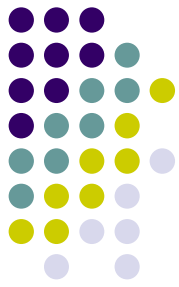
Interrupt Handling

- Hardware dapat membedakan devices mana yang melakukan interupsi.
 - Jenis interupsi :
 - *polling*
 - *vectored* interrupt system
- Tugas sistim operasi menyimpan status CPU (program counter, register dll)
 - Jika service routine telah selesai => CPU dapat melanjutkan instruksi terakhir yang dikerjakan
 - Sistim operasi akan “load” kembali status CPU tersebut.



✳ Struktur I/O

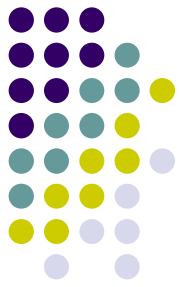
- User request I/O:
 - CPU: load instruksi ke register controller
 - Controller: menjalankan instruksi
- Setelah I/O mulai, control kembali ke user program jika operasi I/O telah selesai
 - Instruksi khusus: wait => CPU menunggu sampai ada interrupt berikutnya dari I/O tersebut.
 - Paling banyak hanya mempunyai satu I/O request.
 - Keuntungan: CPU mengetahui secara pasti device mana yang melakukan interrupt (operasi I/O selesai).
 - Kerugian: operasi I/O tidak dapat serentak untuk semua device



I/O Interrupt

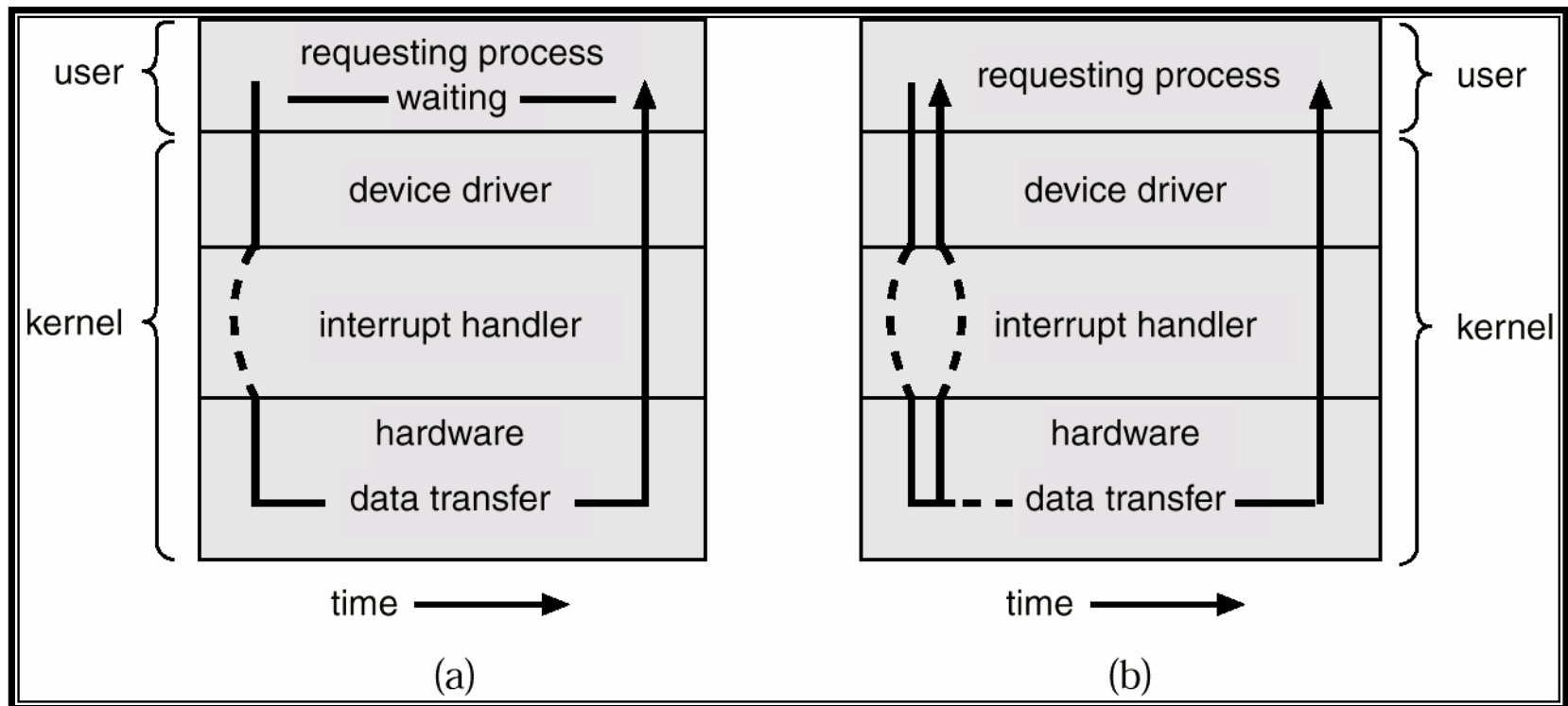
- Pilihan lebih baik: asynchronous I/O
- Setelah I/O mulai, kendali langsung kembali ke user program tanpa menunggu I/O selesai
 - CPU dapat melanjutkan operasi I/O untuk device yang lain
 - User program dapat menjalankan program tanpa menunggu atau harus menunggu sampai I/O selesai.
 - *System call* – request ke OS untuk operasi I/O dan **menunggu** sampai I/O selesai.
- Potensi lebih dari satu device
 - User hanya dapat menggunakan I/O melalui system call
 - *Device-status table* memuat informasi untuk setiap I/O device: tipe, alamat, status dll
 - OS mengatur tabel ini dan mengubah isinya sesuai dengan status device (interrupt)

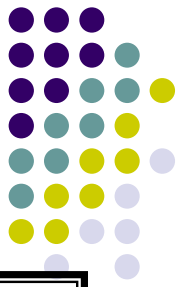
Dua Metode I/O



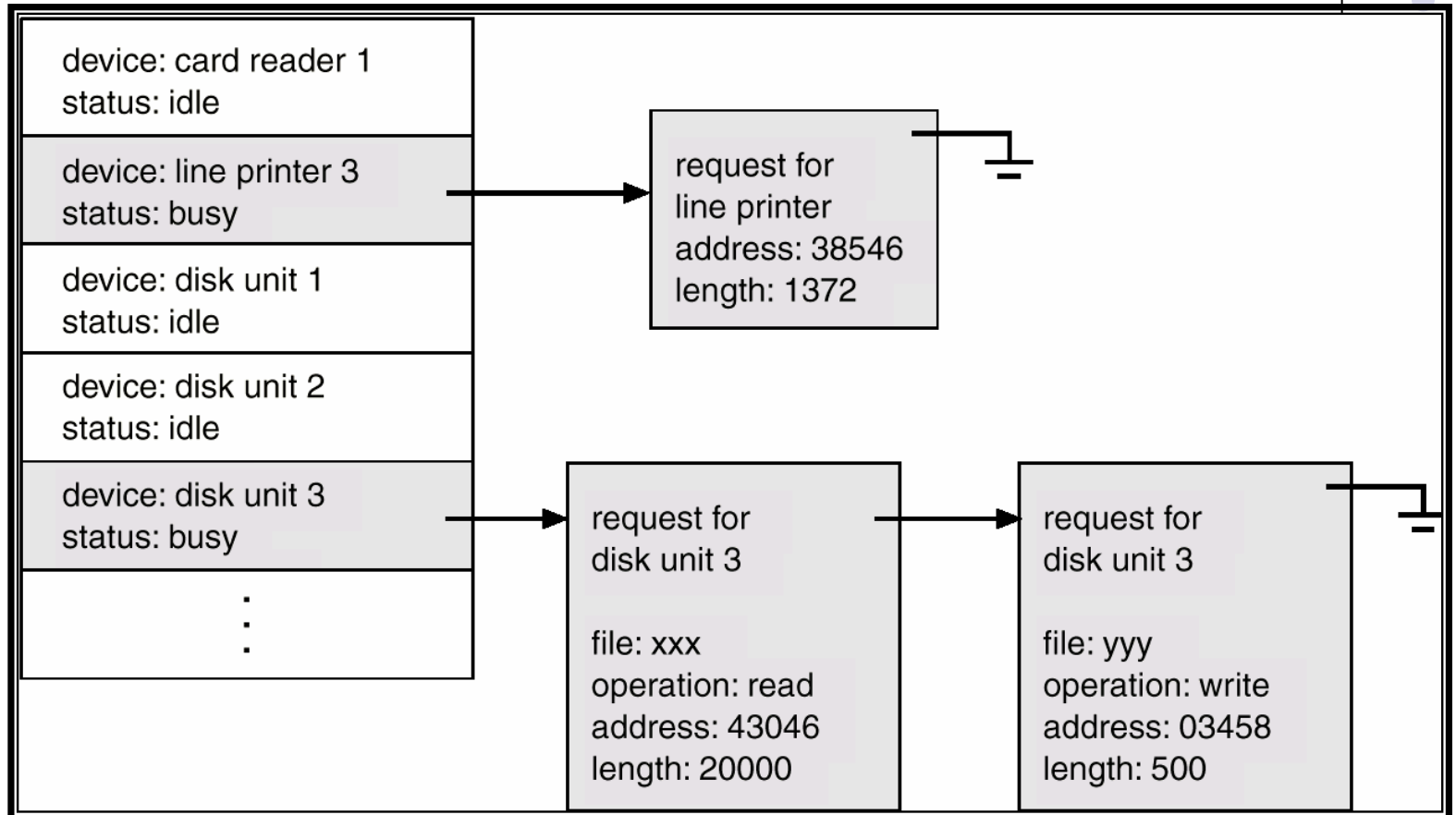
Synchronous

Asynchronous

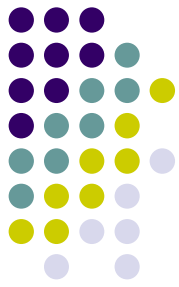




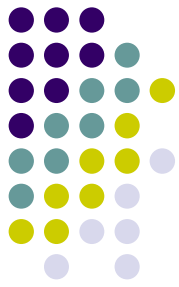
Device-Status Table



Direct Memory Access (DMA)



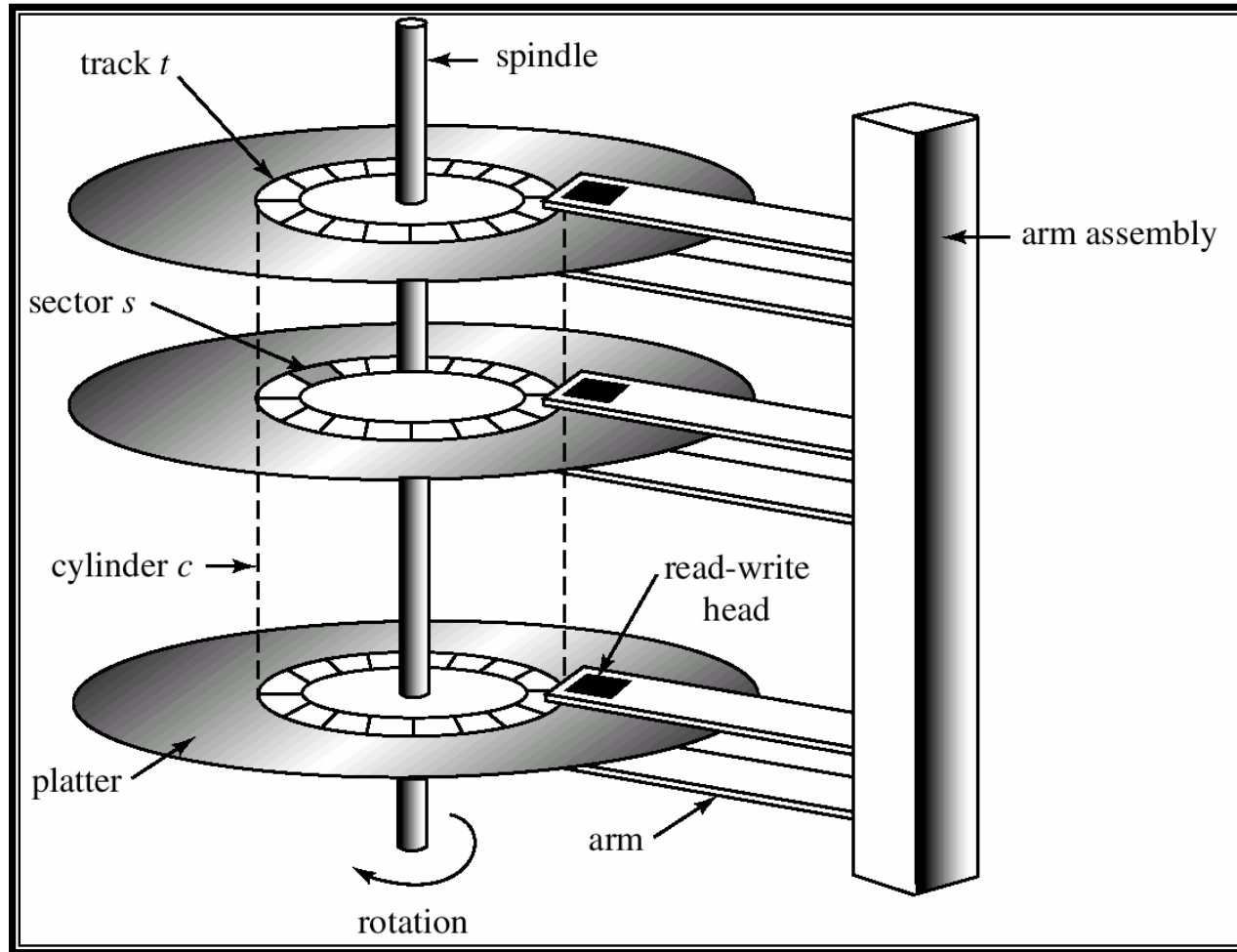
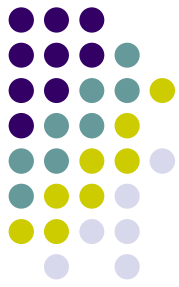
- Jika I/O devices sangat cepat (“high-speed”), beban CPU menjadi besar harus mengawasi transfer data dari controller ke memory dan sebaliknya.
- Hardware tambahan => DMA controller dapat memindahkan blok data dari buffer langsung ke memory tanpa mengganggu CPU.
 - CPU menentukan lokasi memory dan jika DMA controller telah selesai => interrupt ke CPU
 - Hanya satu interrupt ke CPU untuk sekumpulan data (blok).

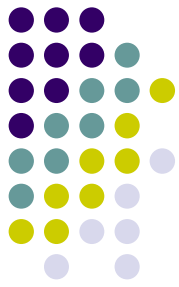


* Struktur Storage

- Main memory
 - Media penyimpanan, dimana CPU dapat melakukan akses secara langsung
- Secondary storage
 - Tambahan dari main memory yang memiliki kapasitas besar dan bersifat nonvolatile
- Magnetic disks
 - Metal keras atau piringan yang terbungkus material magnetik
 - Permukaan disk terbagi secara logikal dalam *track*, yang masing-masing terbagi lagi dalam *sector*
 - Disk controller menentukan interaksi logikal antara device dan komputer

Mekanisme Pergerakan Head-Disk

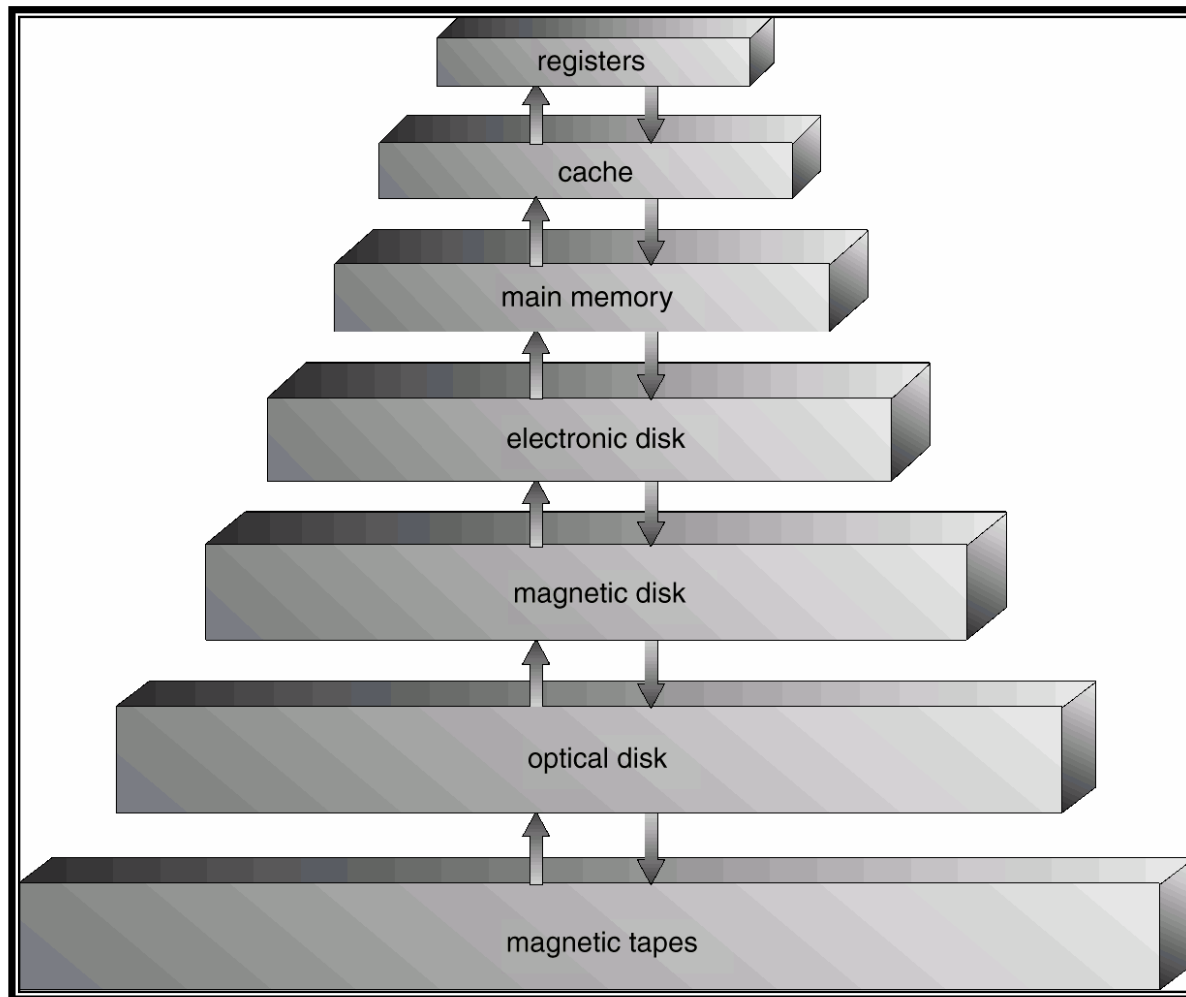
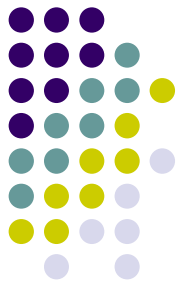




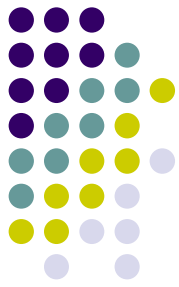
Hirarki Storage

- Hirarki sistem storage, diorganisasikan dalam bentuk :
 - Kecepatan
 - Biaya
 - Volatilitas
- *Caching*
 - Penduplikasian informasi ke dalam sistem storage yang cepat dapat dilakukan melalui cache pada secondary storage

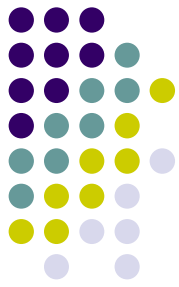
Hirarki Storage-Device



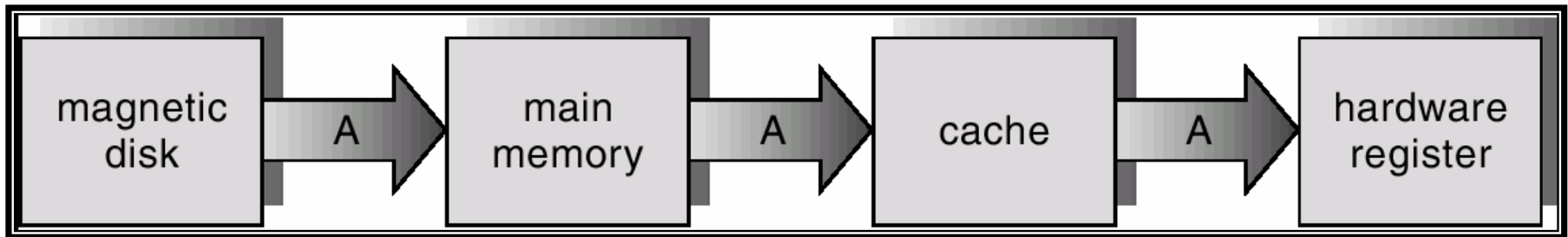
Caching

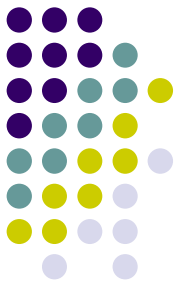


- Menggunakan memori berkecepatan tinggi untuk menangani akses data saat itu juga (yang terbaru)
- Membutuhkan manajemen cache.
- Caching mengenalkan tingkatan lain dalam hirarki storage, dimana data secara serentak disimpan pada lebih dari satu tingkatan secara konsisten



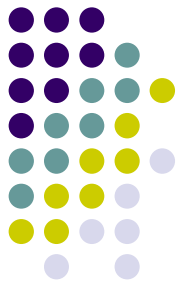
Migrasi dari Disk ke Register





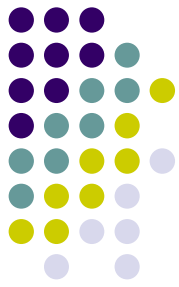
* Proteksi Hardware

- Dual-Mode Operation
- Proteksi I/O
- Proteksi Memory
- Proteksi CPU



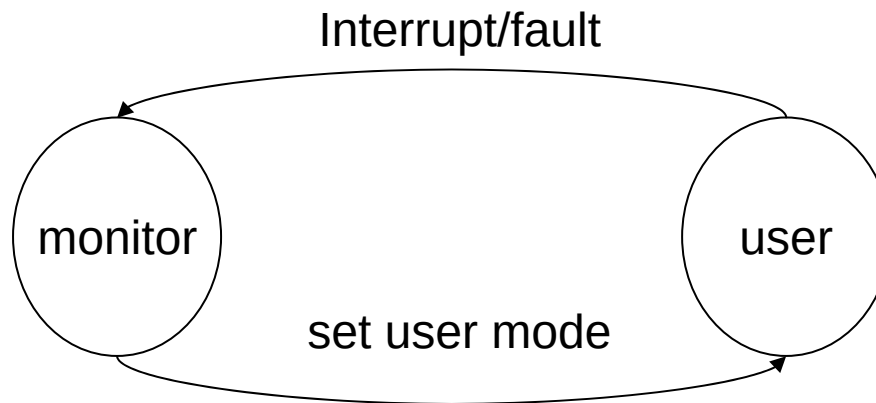
Dual-Mode Operation

- Penggunaan resource sharing membutuhkan sistem operasi yang menjamin suatu program yang salah tidak menyebabkan program lain tidak terpengaruh
- Menyediakan dukungan hardware yang dibedakan ke dalam dua mode operasi :
 1. *User mode* – eksekusi dilakukan untuk kepentingan user.
 2. *Monitor mode* (disebut juga *kernel mode* atau *system mode*) – eksekusi dilakukan untuk kepentingan sistem operasi.

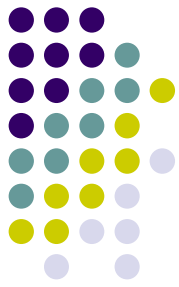


Dual-Mode Operation (Cont.)

- *Mode bit* ditambahkan pada computer hardware (CPU) untuk indikasi mode sekarang: monitor (0) atau user (1).
- Jika terjadi interrupt/fault/error => hardware mengubah mode ke monitor



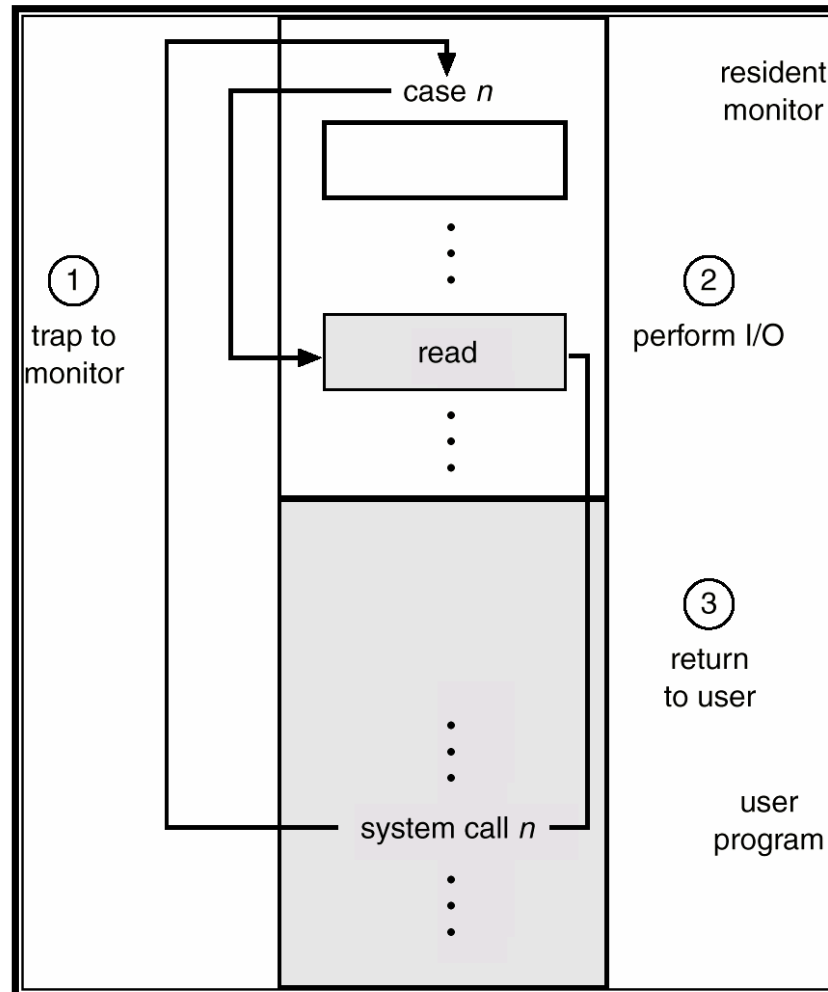
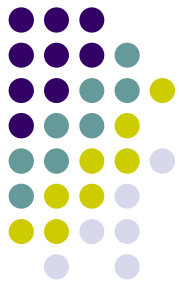
Instruksi Privileged hanya dapat diberikan dalam mode monitor

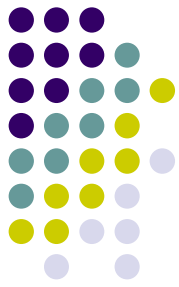


Proteksi I/O

- Semua instruksi I/O adalah instruksi privileged:
 - Hanya dapat dilakukan melalui OS
 - OS dapat mencegah “request” ke I/O dengan melihat mode saat ini.
- OS menjaga supaya program user tidak dapat menjadi “monitor mode” untuk mencegah user program melakukan:
 - Menangani interrupt: dengan mengubah alamat interrupt vector.
 - Mengubah status dan data pada “device table”

Penggunaan System Call untuk Pengoperasian I/O

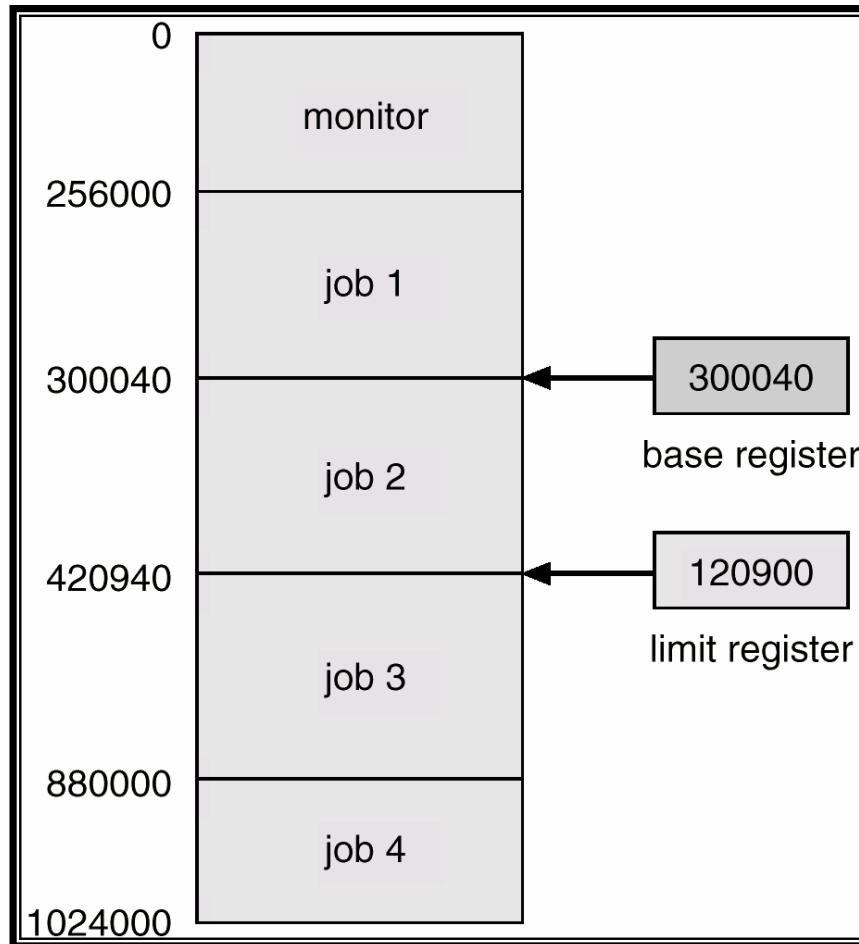
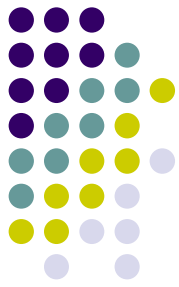


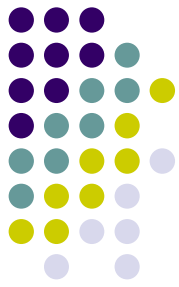


Proteksi Memory

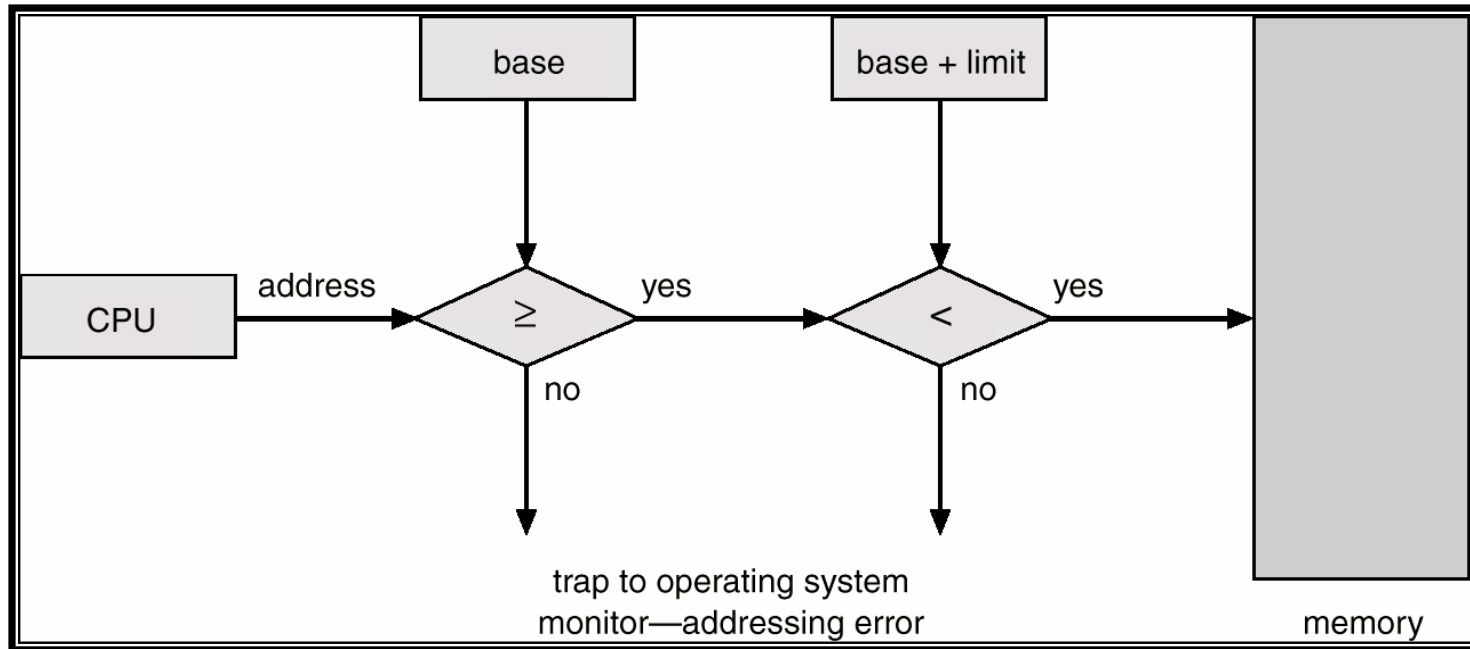
- Melindungi memori terutama untuk isi:
 - interrupt vector dan interrupt service routines.
- Cara umum adalah setiap user program hanya dapat mengakses lokasi memori yang telah dibatasi (disediakan untuk program tsb).
 - **Range address** – alamat yang valid
 - **Base register** – menyimpan alamat terkecil memori secara fisik
 - **Limit register** – besarnya jangkauan memori yang diijinkan
- Memori diluar range tersebut tidak dapat diakses oleh user program tsb.

Penggunaan Base dan Limit Register

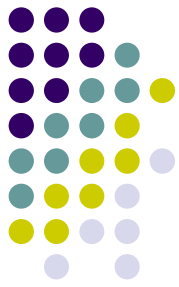




Proteksi Alamat Hardware



- Ketika eksekusi pada mode monitor, OS dapat mengakses semua lokasi memori.
- Pemuatan instruksi ke base dan limit register tergantung instruksi privileged



Proteksi CPU

- *Timer*
 - Interupsi secara berkala oleh hardware: => transfer control ke OS.
 - Nilai timer akan berkurang sesuai “clock tick” dari hardware komputer.
 - Saat nilai timer menjadi 0, interrupt terjadi
 - Housekeeping: melakukan CPU scheduling (jatah CPU), status device table dll.
- Timer digunakan untuk system time.