Konsep SISTEM OPERASI

Pengenalan Sistem Operasi

Course Objective

- Definisi Sistem Operasi.
- Peran Sistem Operasi dalam Sistem Komputer.
- Tujuan Sistem Operasi.
- Sejarah perkembangan Sistem Operasi.

Sistem Operasi (1)

- OS (Operating System) merupakan merupakan program yang mengatur eksekusi program dan bertindak sebagai interface antara aplikasi dan perangkat keras.
- Tujuan Sistem Operasi :
  - Kemudahan.
  - Efisiensi.
  - Kemampuan untuk berkembang.

Sistem Operasi (2)

- OS sebagai interface antara user dan perangkat keras berarti menyediakan mekanisme kapada end user untuk menggunakan utilitas yang disediakan.
- End user tidak mau tahu akan detail proses yang melibatkan computer hardware, sehingga end user hanya berinteraksi via aplikasi-aplikasi yang disediakan.

Sistem Operasi (3)

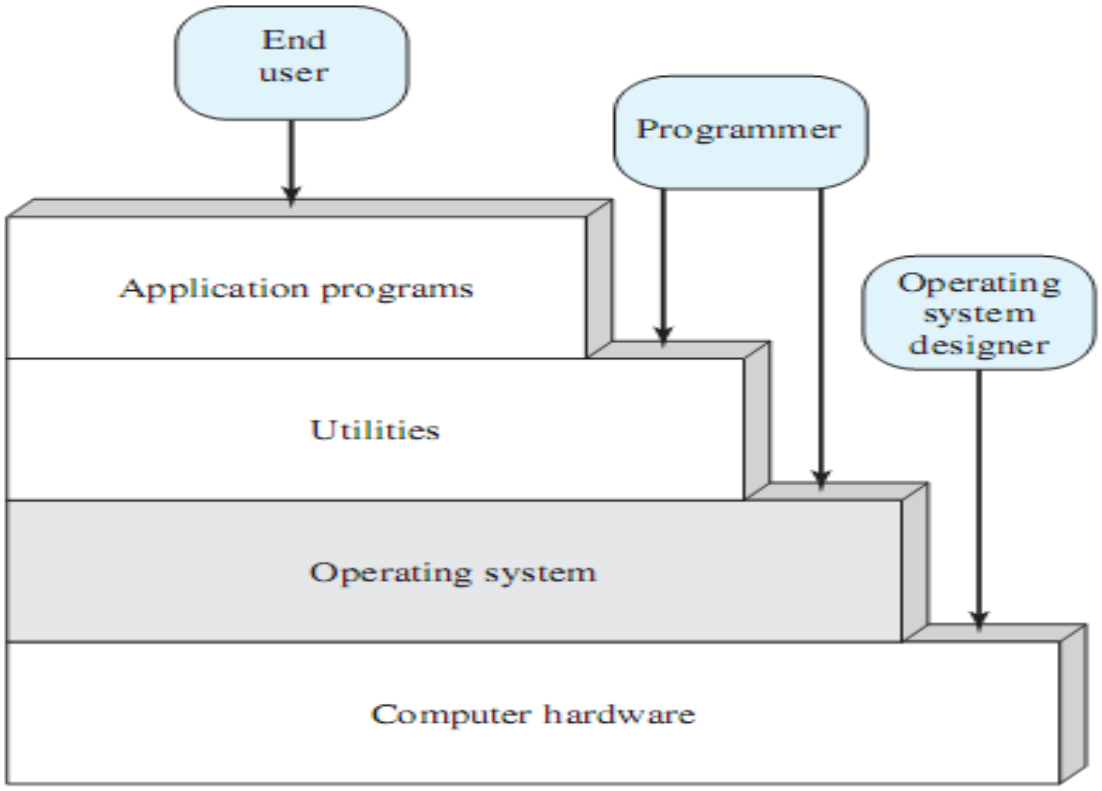


Figure 2.1 **Layers and Views of a Computer System**

Services yang disediakan OS

- Program development.
- Program execution.
- Access to I/O devices.
- Controlled access to File.
- System Access.
- Error Detection and Response.
- Accounting.

OS as Resource Manager

- Resource management : CPU, memory space, file, storage, dll.
- Memberikan / mangalokasikan resource tersebut kepada user sesuai dengan kebutuhan.

Evolusi Sistem Operasi

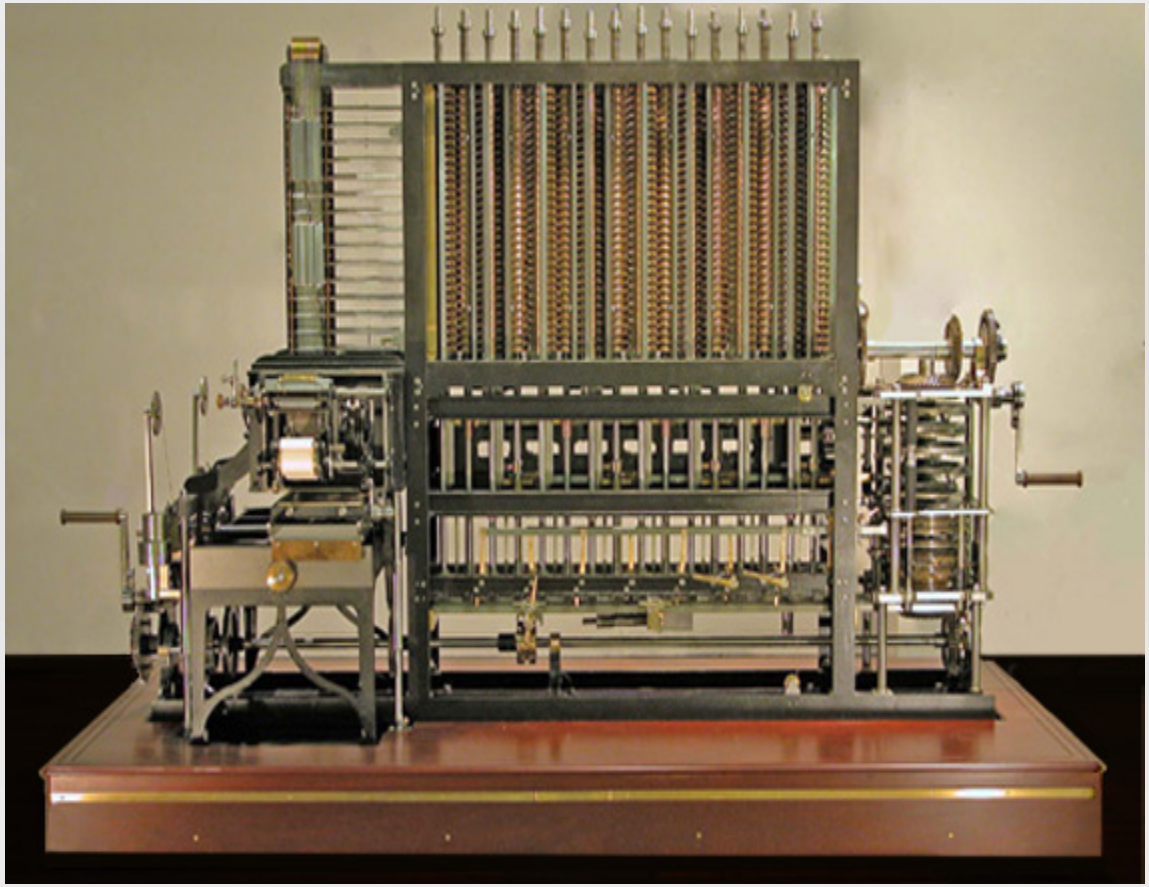
- Perkembangan sistem operasi :
  - Serial Processing.
  - Simple Batch System.
  - Multiprogramming Batch System.
  - Time-Sharing System.

Operating System History (1)

- Computer digital, pertama kali di desain oleh ahli matematika bernama Charles Babbage (1792 – 1871), memberikan gagasan tantang suatu mesin yang terprogram.
- Keterbatasan teknologi masa itu, membuat mesin yang diciptakan Charles Babbage tidak dapat bekerja sesuai dengan yang diinginkan.
- Mesin yang diciptakan hanya berupa mekanis, tanpa adanya sistem operasi.
- Bekerja sebagai mesin hitung.

Operating System History (2)

- Mesin yang diciptkan Charles, digunakan untuk melakukan perhitungan beberapa model matematis.
- Menggunakan punch card untuk memasukan nilai / angkat yang akan digunakan dalam perhitungan.



Operating System History (3)

- OS Generations
  - Generation 1 (1945 – 55)
    Vacuum tubes, plugboards, and serial processing
  - Generation 2 (1955 – 65)
    Transistors and batch systems
  - Generation 3 (1965 – 80)
    ICs and multiprogramming
  - Generation 4 (1980 – present)
    Personal computers and interactivity
  - Generation 5 (present – ?)
    self-organizing systems?

Serial Processing (1)

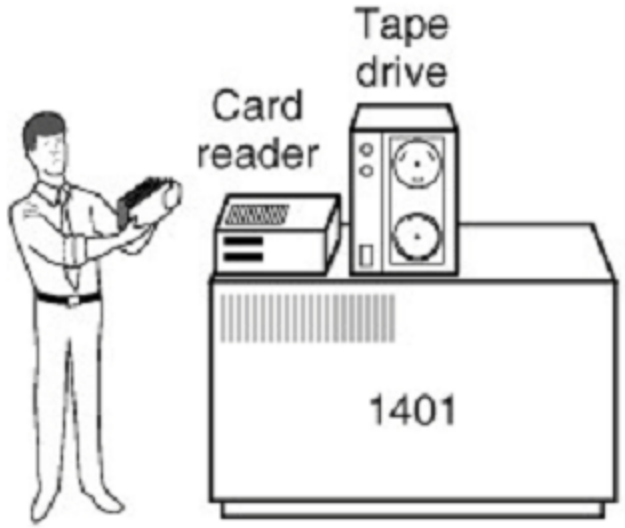
- Pada generasi ini, *programmer* berinteraksi langsung dengan perangkat keras, tidak terdapat sistem operasi.
- Operasional mesin, dikontrol dengan *console* yang berupa *display light, toggle switch, input devices*.
- *Input device* dapat berupa *card reader*, sedangkan *output devi*ces dapat berupa *printer*.
- Tidak terdapat mekanisme penjadwalan / scheduling dinamis, karena programmer menentukan alokasi waktu setiap job yang akan dikerjakan.

Serial Processing (2)

- Kelemahan pada generasi ini adalah :
  - Suatu task bisa saja telah selesai sebelum menghabiskan jatah waktu yang telah ditentukan sebelumnya, sehingga terdapat waktu yang dihabiskan untuk menunggu untuk mengerjakan job selanjutnya.
  - Atau malahan waktu yang telah dialokasikan sebelumnya tidak cukup untuk menyelesaikan job, sehingga dipaksa untuk dihentikan.

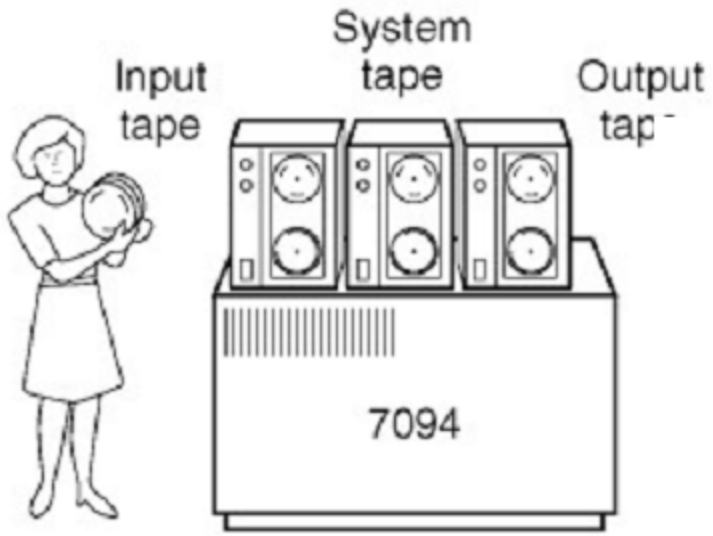
Simple Batch System (1)

- Operation of an early batch system:
  1. Programmer brings cards to 1401.
  2. 1401 reads batch of jobs onto tape.



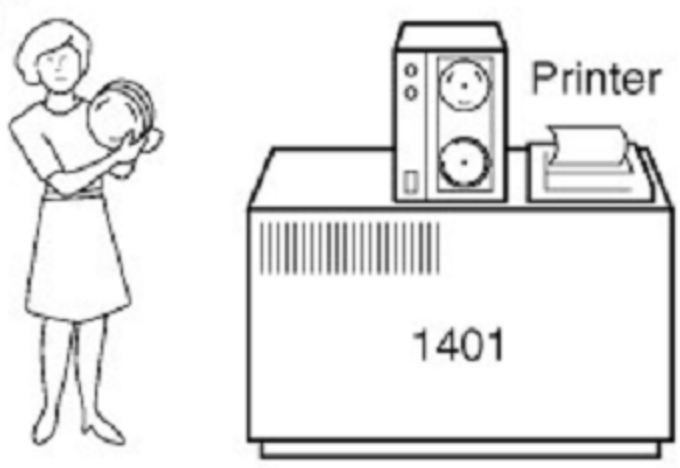
Simple Batch System (2)

- Operation of an early batch system:
  3. Operator carries input tape to 7094.
  4. 7094 does computing.



Simple Batch System (3)

- Operation of an early batch system:
  5. Operator carries output tape to 1401.
  6. 1401 prints output.

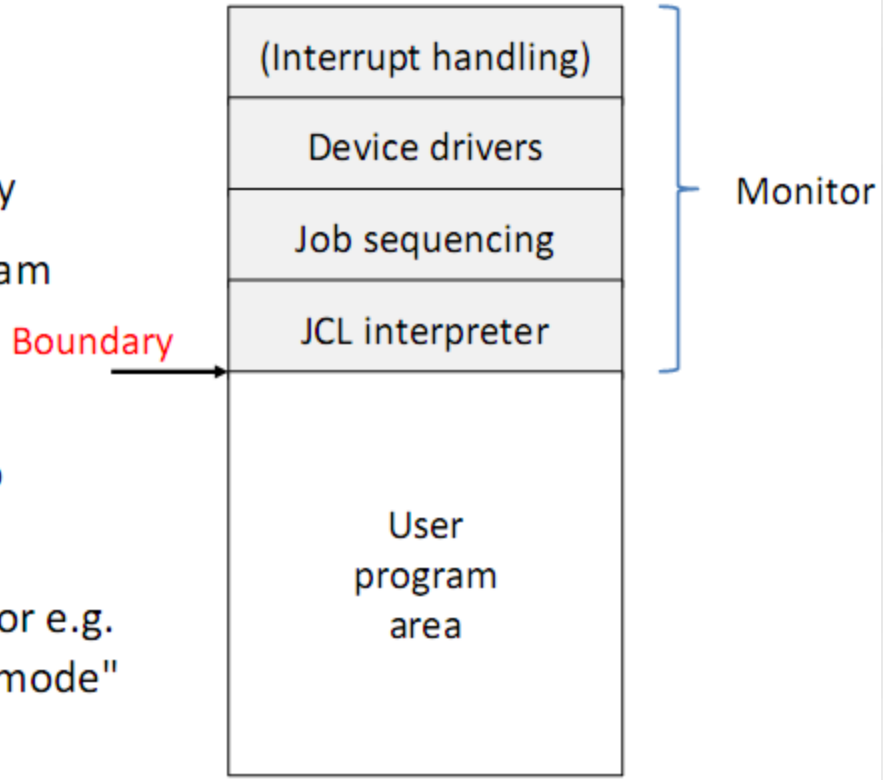


Simple Batch System (4)

- Simple batch system
  - Monitor
    - Software that controls the sequence of events
    - Batch jobs together
    - Application program returns control to monitor when finished
  - Job Control Language (JCL)
    - Special type of programming language
    - Provides instruction to the monitor:
      What compiler to use, what data to use

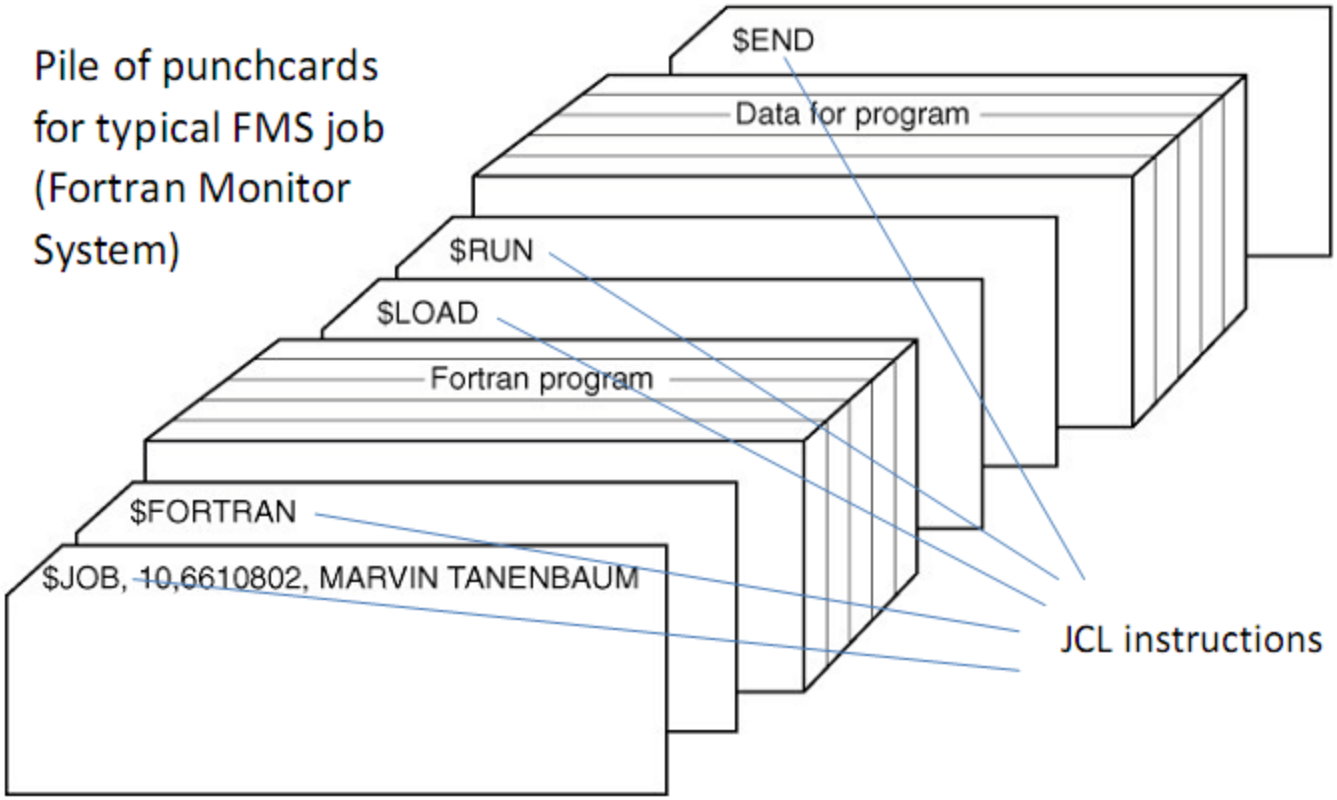
Simple Batch System (5)

- Simple batch system
  - Monitor features
    - resident in main memory
    - protection: if user program crosses boundary, abort job
    - timer to prevent a job to monopolize the system
    - privileged instructions, for e.g. I/O, available in "kernel mode" only



Simple Batch System (6)

- Simple batch system
  - Pile of punchcards for typical FMS job (Fortran Monitor System)



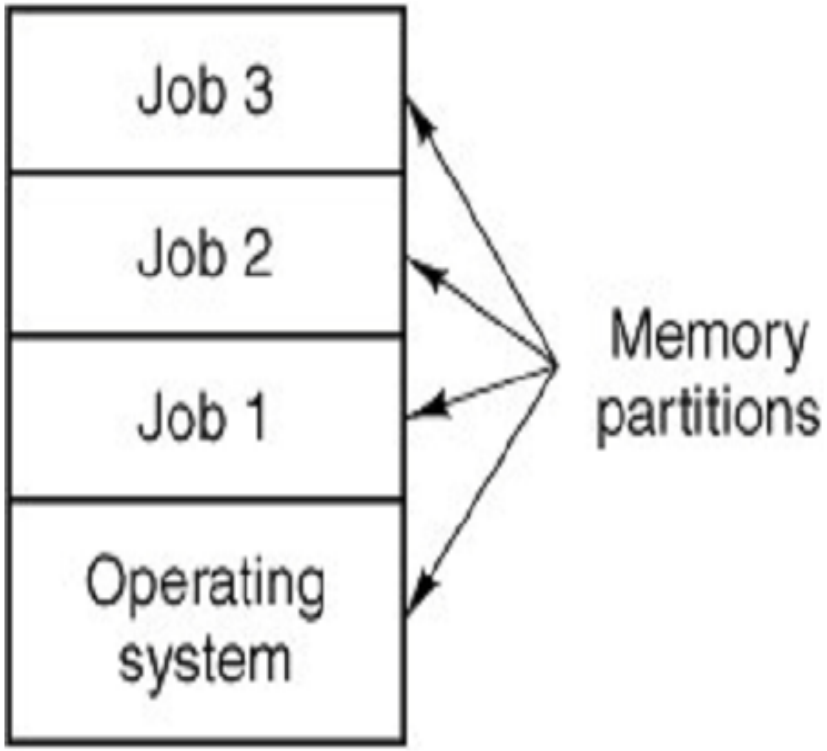
Simple Batch System (7)

- Sistem Operasi Batch Sederhana menghasilkan mekanisme pengurutan dan pengelompokan instruksi secara otomatis.
- Masalah timbul jika mekanisme eksekusi instruksi berhubungan dengan I/O.
- Masalahnya I/O relatif lambat jika dibandingkan dengan processor, sehingga terdapat banyak sekali kondisi idle.

Multiprogramming (1)

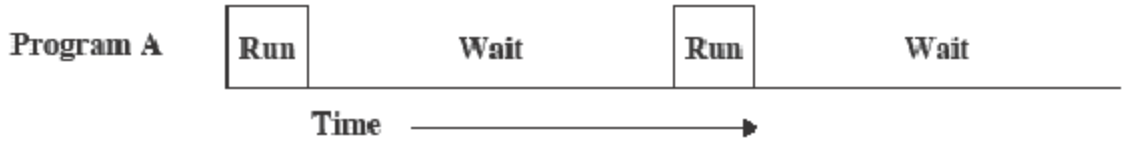
- Kelemahan pada Sistem Batch Sederhana, yaitu penggunaan utilitas processor yang seringkali dalam keadaan idle, pada saat menunggu mekanisme dari I/O.
- Ide : pada saat processor menunggu mekanisme dari I/O, processor dapat melakukan eksekusi instruksi yang lain.

- Multiprogramming
  - Put more jobs into memory!



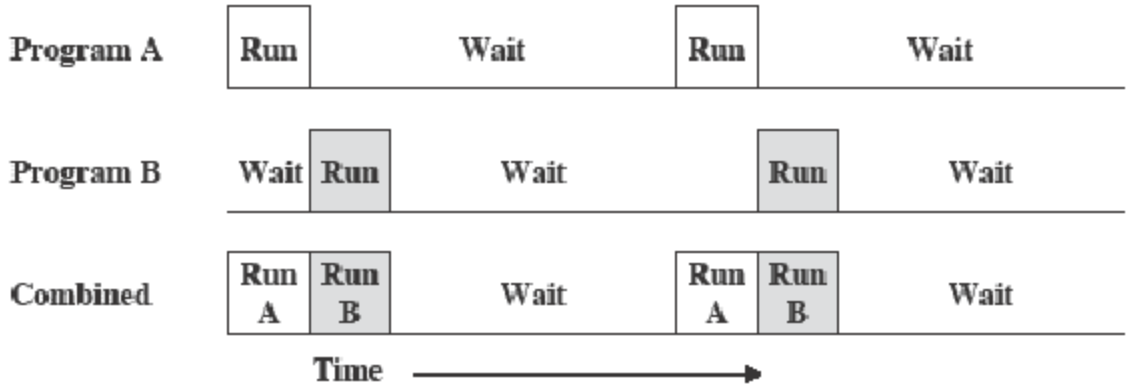
Multiprogramming (2)

- Uniprogramming



Processor must wait for I/O instruction to complete before preceding

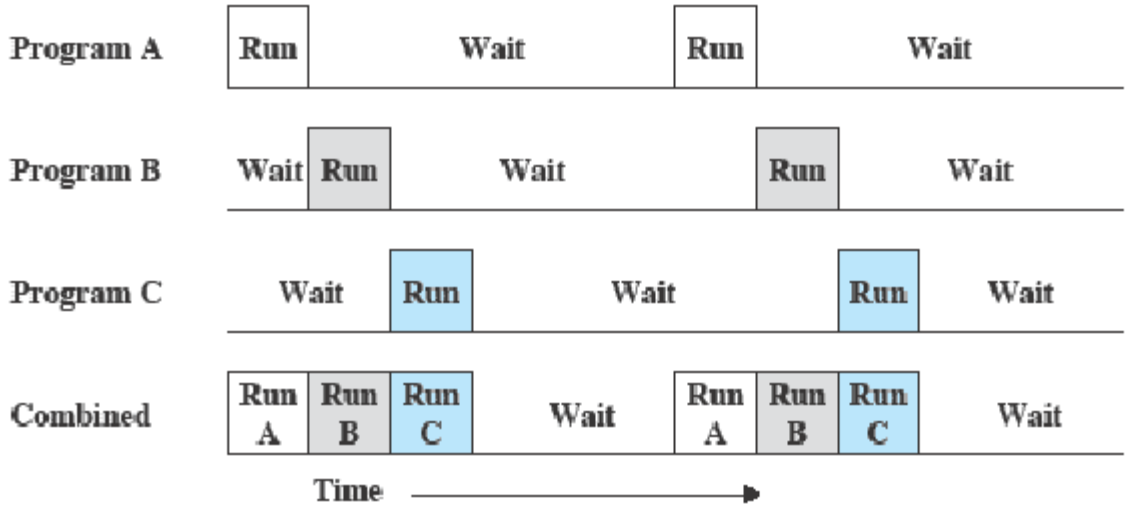
- Multiprogramming with two programs



When one job needs to wait for I/O, the processor can switch to the other job

Multiprogramming (3)

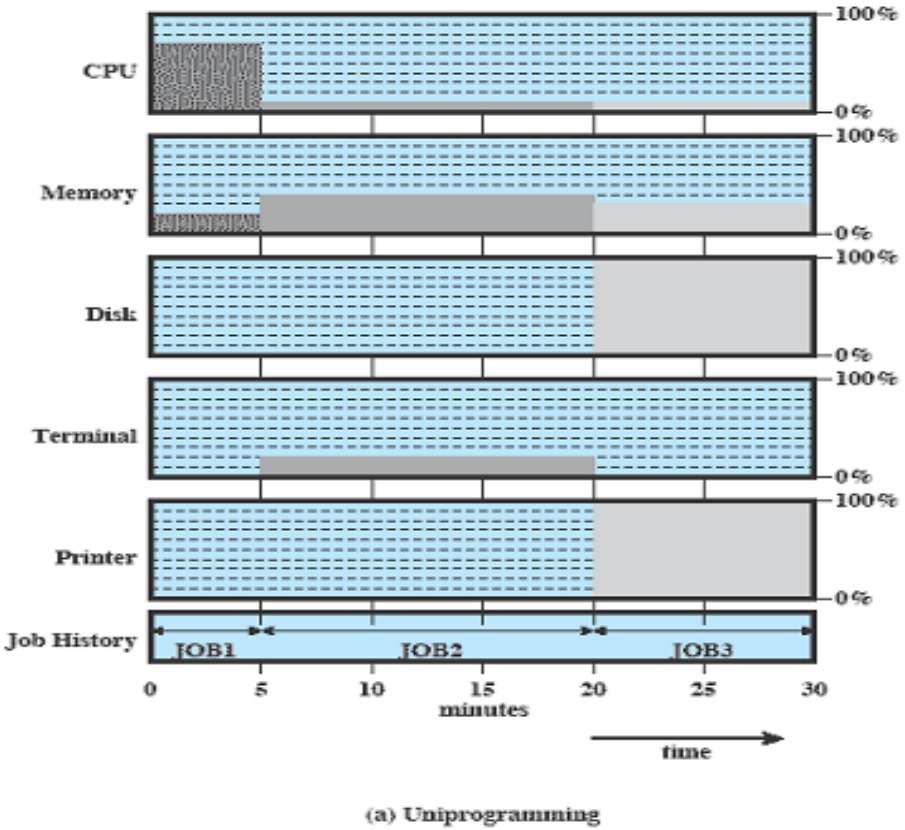
- Multiprogramming with three programs



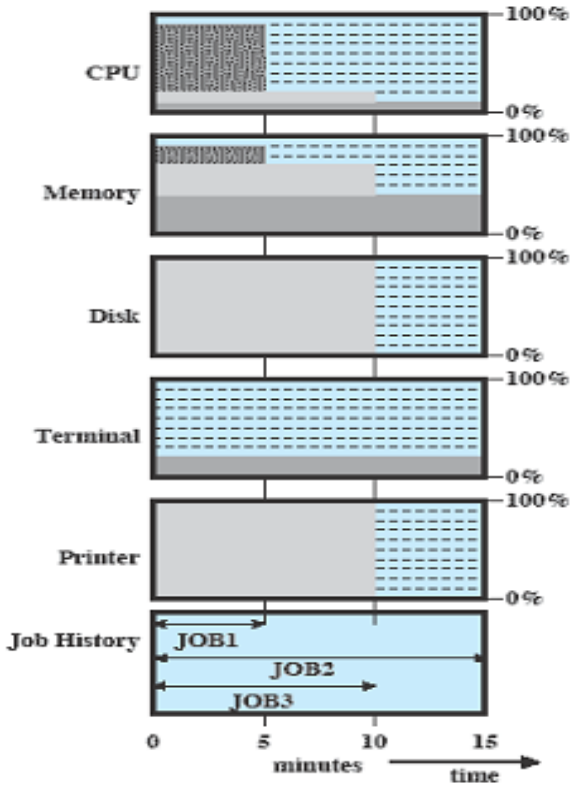
  - Price for task switching should be less than gain in system utilization!

Multiprogramming (4)

- Utilization graphs for example



(a) Uniprogramming



(b) Multiprogramming

Multiprogramming (5)

- Effects of multiprogramming for example

	Uniprogramming	Multiprogramming
Processor use	20 %	40 %
Memory use	33 %	67 %
Disk use	33 %	67 %
Printer use	33 %	67 %
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

Time-Sharing OS (1)

- Time Sharing Systems
  - Using multiprogramming to handle multiple interactive jobs
  - Processor's time is shared among multiple users
  - Multiple users simultaneously access the system through terminals
  - Goals compared:

	Batch Multiprogr.	Time Sharing
Principal objective	Maximize processor usage	Minimize response time
Source of directives to OS	JCL commands provided with job	Commands entered at the terminal

Time-Sharing OS (2)

- Time Sharing Systems
  - One of the first time-sharing OS: Compatible Time-Sharing System (CTSS), developed at the Massachusetts Institute of Technology (MIT)
  - CTTS developed for IBM 709, later IBM 7094
    - 32 k words à 36 bits, 5 k words for CTTS
    - timer interrupt period: 200 ms
      → return control from user program to CTSS
    - focus on minimizing transfers to/from disk

Time-Sharing OS (3)

Table 2.3 Batch Multiprogramming versus Time Sharing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Time-Sharing OS (4)

- Sample CTSS run