AS/A Level Computing
Syllabus 2011

Section 3

- System Software Mechanisms -

- Machine Architecture -

- Database Theory -

- Programming Paradigms -

Chapter 3.1
The Functions of Operating Systems

3.1 The Functions of Operating Systems

3.1.1 THE FUNCTIONS OF OPERATING SYSTEMS

- All Operating Systems (OS) have three main functions:
  - Controlling and Managing Hardware
  - Providing An Interface (Human-Machine and Machine-Software)
  - Facilitating Application Software To Run

- Other services/functions are OS dependant.
  - E.g. Network Operating System (NOS)
    - Data Security
    - System Access Verification
    - User Privileges
    - Data Communication

3.1 The Functions of Operating Systems

3.1.1 THE FUNCTIONS OF OPERATING SYSTEMS *(Continued…)*

- Why Put So Much Responsibility On The OS?
  - Reduce Application Software Code – less functionality to program
  - Consistency of Procedures – every application uses the same style and steps
  - Security of Data And Hardware – restricts every program's low-level access
  - Reusability – same code/functionality can be used by different programs
  - Scalability – functionality/procedures can be up graded easily
  - Reliability – functionality/procedures can be trusted
  - Reduce Development Cost, Time – no need to learn new features

3.1 The Functions of Operating Systems

- Exploring further:
  - Interrupt Management
  - Scheduling
  - Memory Management
  - SPOOLing (Simultaneous Peripheral Operations Online)
  - Disk/File Management
  - Resource Sharing and Management
  - Network Operating Systems (NOS) And Their Additional Functionality

3.1 The Functions of Operating Systems

3.1.2 SCHEDULING

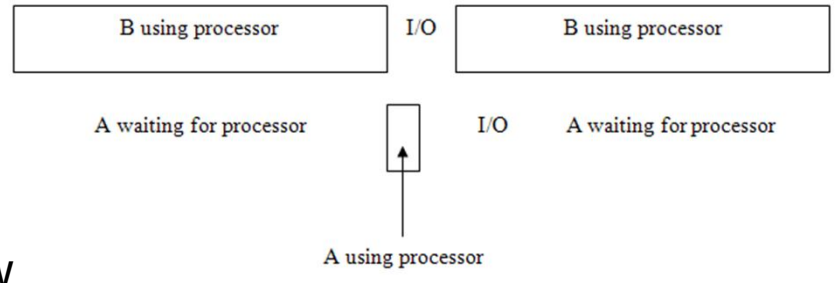
•Definition: An operating-system process that starts and ends tasks (programs), manages concurrently running processes, and allocates system resources.

REASONS FOR SCHEDULING

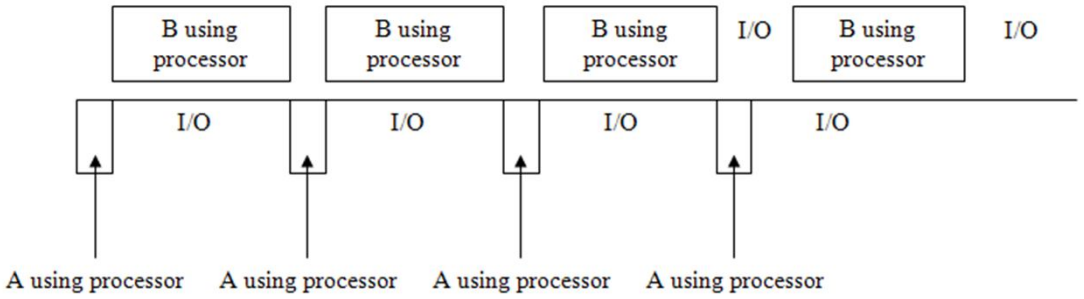
•maximise the use of the whole of the computer system
•be fair to all users
•provide a reasonable response time to all users, whether they are on-line users or a batch processing user
•prevent the system failing if it is becoming overloaded
•make sure that the system is consistent by always giving similar response times to similar activities from day to day

3.1 The Functions of Operating Systems

EXAMPLE OF SCHEDULING



- Job A is processor-bound and Job B is I/O bound.
- I/O bound jobs are given more time in order to allow them to finish in reasonable time.
- Processor bound jobs are given lesser time as they can do more in less time and shouldn't hold the I/O bound jobs.



3.1 The Functions of Operating Systems

STRATEGIES FOR SCHEDULING

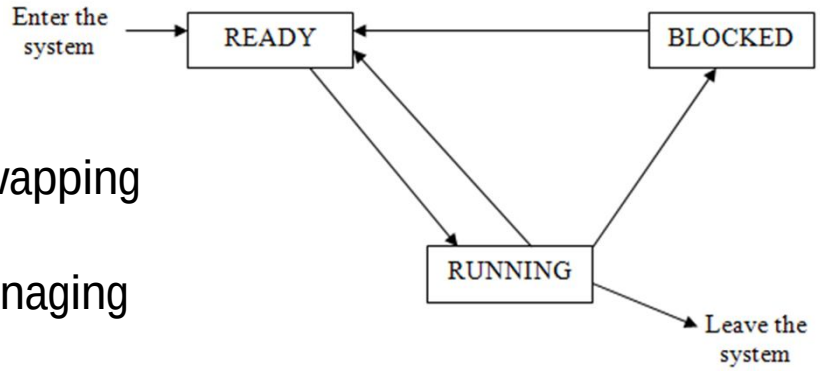
- Scheduling of processes requires careful considerations. Following are the most common:
  - Priority: Giving some jobs priority over others.
  - I/O or Processor Bound: Deciding which job is processor bound and which is is I/O bound.
  - Type of Job: Batch processing, on-line and real-time jobs all require different response times.
  - Resource Requirements: The amount of time needed to complete the job, the memory required, I/O and processor time.
  - Resources Used So Far: The amount of processor time used so far, how much I/O used so far.
  - Waiting Time: The time the job has been waiting to use the system.

3.1 The Functions of Operating Systems

PROCESS HANDLING

•A process, at any given time, could be in one of these three states:

- •Jobs enter in the ready queue.
- •High-Level Scheduler (HLS) is responsible for putting jobs in READY queue.
- •Mid-Level Scheduler (MLS) is responsible for swapping jobs between main and secondary memory.
- •Low-Level Scheduler (LLS) is responsible for managing jobs in the RUNNING and BLOCKED queues.



•Pre-emptive Scheduling allows LLS to put or remove jobs from the RUNNING queue as and when needed.

•Non Pre-emptive Scheduling allows jobs to run until they no longer need the processor.

3.1 The Functions of Operating Systems

SCHEDULING TECHNIQUES

FCFS: Simply means that the first job to enter the ready queue is the first to enter the running state. This favours long jobs.

SJF: Simply means sort jobs in the ready queue in ascending order of times expected to be needed by each job. New jobs are added to the queue in such a way as to preserve this order.

RR: This gives each job a maximum length of processor time (called a time slice) after which the job is put at the back of the ready queue and the job at the front of the queue is given use of the processor. If a job is completed before the maximum time is up it leaves the system.

SRT: The ready queue is sorted on the amount of expected time still required by a job. This scheme favours short jobs even more than SJF. Also there is a danger of long jobs being prevented from running.

PQ: Priority Queues involve queues of different priorities. Jobs in higher priority queues are executed first.

MFQ: Involves several queues of different priorities with jobs migrating downwards.

3.1 The Functions of Operating Systems

3.1.2 INTERRUPTS

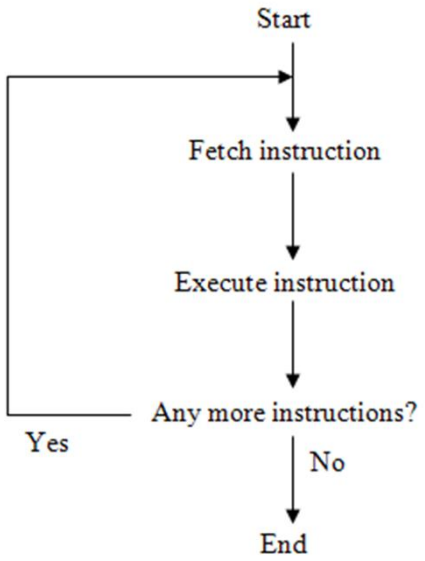
•Definition: Interrupts are signals generated by hardware or software to get the processor's attention.

•The processor processes program instructions stored in the memory. It fetches instructions one after the other and executes them.

•If a problem occurs (anywhere in the system) while the processor is busy processing, the processor must respond to the problem as soon as possible in order to avoid damage to data and/or hardware.

•This requires the processor to stop whatever it is doing and give its attention to the problem.

•The device (hardware) or software, where the problem has occurred generates a signal, called an "interrupt" to get processor's attention.



3.1 The Functions of Operating Systems

3.1.2 INTERRUPTS *(Continued…)*

TYPES OF INTERRUPTS

- Hardware: Generated by hardware devices. (Hardware failure. Hardware going off-line)
- Software: Generated by software applications. (Violation of memory space. Virus found)
- Timer: Generated by internal clock. (Scheduling of processes; an event is due)
- I/O: Generated by an I/O device. (Printer out of paper; CD copying completed)

3.1 The Functions of Operating Systems

3.1.2 INTERRUPTS *(Continued…)*

- There are several methods of servicing Interrupts.
- One method is to store an interrupt in a queue as it is generated.
  - The queue is then processed in FIFO (First In, First Out) order.
  - Priority is not important in this method.
- Another method is to have queues of different priorities and store each interrupt according to its priority in the related queue.
  - The interrupts in the high priority queue are given more importance and are serviced first.

3.1 The Functions of Operating Systems

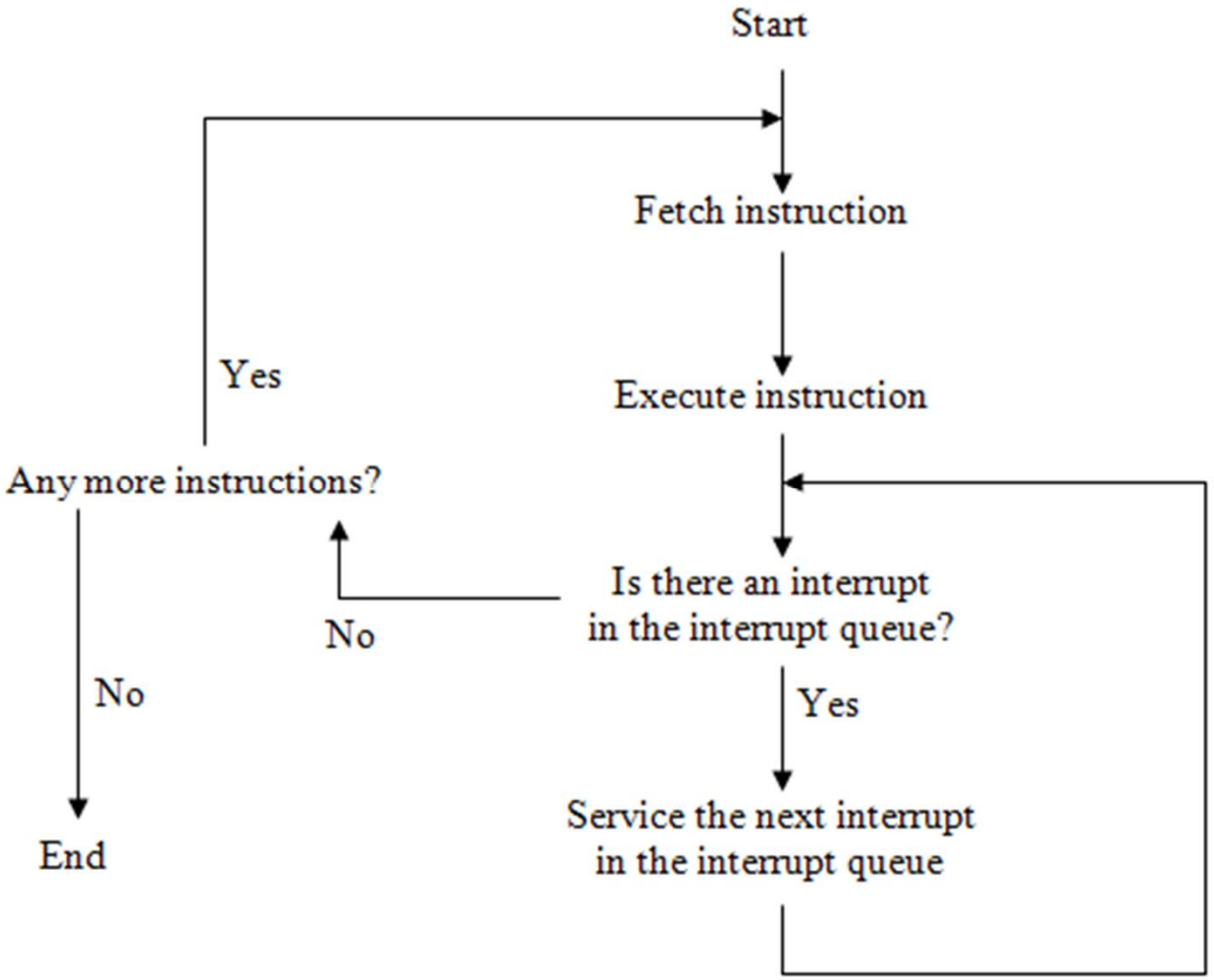
3.1.2 INTERRUPTS *(Continued…)*

•The processor 'services' or attends to the interrupts in the following manner:

a. The processor stops the execution of the program and attends to the interrupt in the following manner:
b. The processor finishes executing the current instruction.
c. The processor saves the contents of the 'special registers' that hold vital information about the program being executed.
d. The processor then checks the 'Interrupt Queue' to see which interrupt is there and attempts to service the interrupt.
e. If there are more than one interrupts in the queue, they are serviced in order of their priority.
f. After the interrupt queue is empty, the processor reloads the saved contents of the special registers and continues the execution of the program from where it was interrupted.

3.1 The Functions of Operating Systems

3.1.2 INTERRUPTS *(Continued…)*

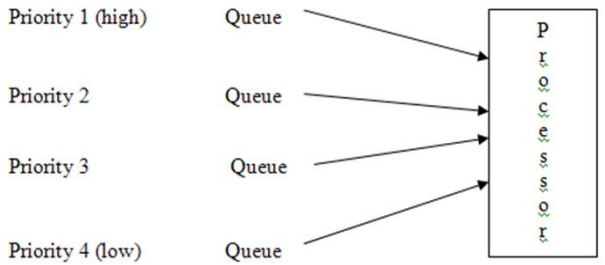


3.1 The Functions of Operating Systems

3.1.4 JOB QUEUES

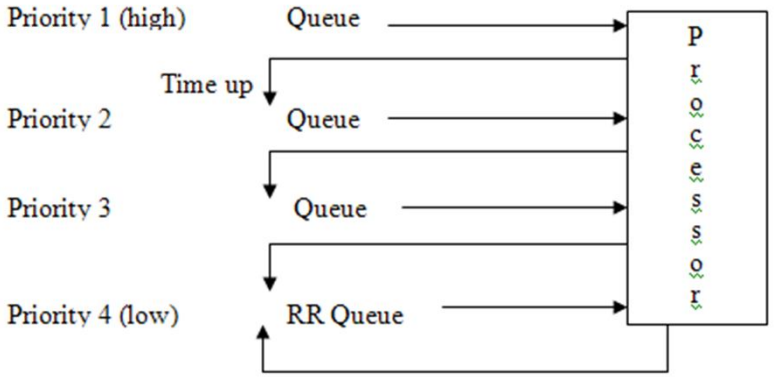
FEEDBACK QUEUES

•In this type of scheduling queues of multiple priorities are used.
•Jobs in the higher priority queues are run first. Then jobs in the next level, and so on.



MULTI-LEVEL FEEDBACK QUEUES

•In this type of scheduling, jobs are demoted to queues of lower priority once they have run (and still not finished).
•Once the jobs reach the lowest priority, Round Robin (RR) strategy is used as there is no queue below it.



3.1 The Functions of Operating Systems

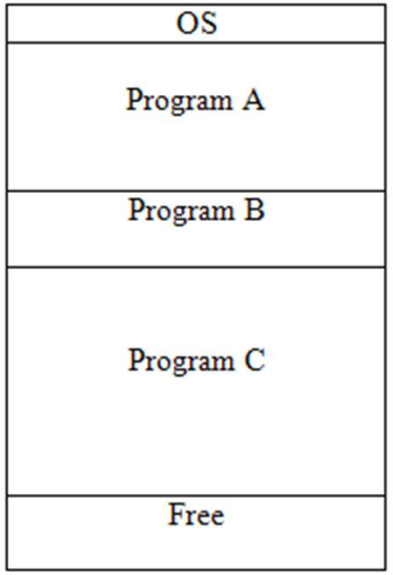
3.1.5 MEMORY MANAGEMENT

- When a job needs to be processed, it is stored in the main memory along with its data.
- The job's code and data must be managed and protected.
- In most of the modern OS, more than one job can be loaded into the memory at one time.

- Definition: Memory Management is the process of managing jobs and their data in the memory.

3.1 The Functions of Operating Systems

3.1.5 MEMORY MANAGEMENT *(Continued…)*



Main Memory

- Main Memory can be imagined as a slotted space.
- Different OS, at times, employ different strategies to divide the memory into slots.
- Sizes of the slots depend upon the strategy used.

Main Points

- A "loader system utility" is used to load and unload jobs from the main memory.
- Part of OS is always present in the main memory.
- More OS components are loaded as and when required. They are unloaded when their task is finished.
- Each memory location has a unique "address". Thus, everything loaded into the main memory has an "address".
- Jobs and their data are loaded into their own space. No other job, or its data, can share the space of another job or its data.
- SIZE DOES MATTER!

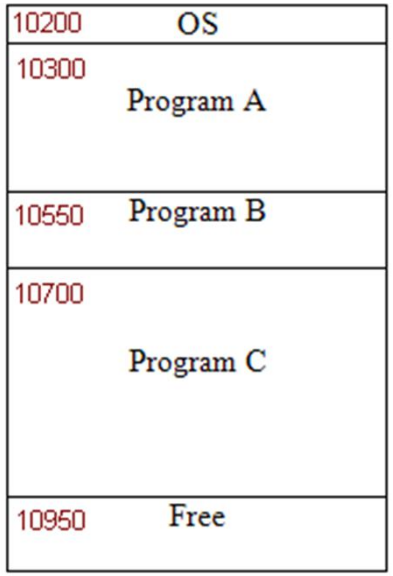
3.1 The Functions of Operating Systems

3.1.5 MEMORY MANAGEMENT *(Continued…)*

Address Calculation and Memory Management

•Whenever a job is loaded into the memory, its address is stored into a table.
•The (some) addresses in the table have to be recalculated:
- •when a new job enters the memory.
- •when a job is temporarily taken out of the memory.
- •when a job terminates permanently.
- •when a job is moved to another memory location.
- •at regular intervals.

•Calculating and recalculating the addresses of jobs is a very processor intensive task and takes a lot of processor time.



What's the Big Deal?

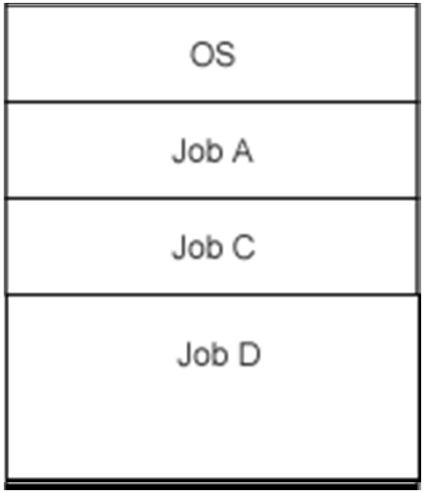
- •If the addresses are not calculated and maintained properly jobs, and their data, can be overwritten by other jobs or their data.
- •Jobs could be loaded into sensitive memory locations (where OS is loaded).
- •Jobs will be loaded in a haphazard manner and memory will usage will not be optimal.

3.1 The Functions of Operating Systems

3.1.5 MEMORY MANAGEMENT *(Continued…)*

Loading and Unloading of Jobs

- One way of loading and unloading is to provide space to jobs as they are needed.
- Jobs (full code and data) are loaded into memory at first available space.
- If the space is not enough, then "Out of Memory" exception occurs.
- If a job is unloaded, it leaves a "hole" in the memory.
- If now another job enters the memory, and it can't fit in the hole it can't be loaded.
- The OS uses "variable partitioning with compaction" technique to move jobs in order to eliminate "holes" from the memory.



3.1.5 MEMORY MANAGEMENT *(Continued…)*

REENTRANTS

•What if two jobs (e.g. two source codes) want to access another job (e.g. a compiler) at the same time?

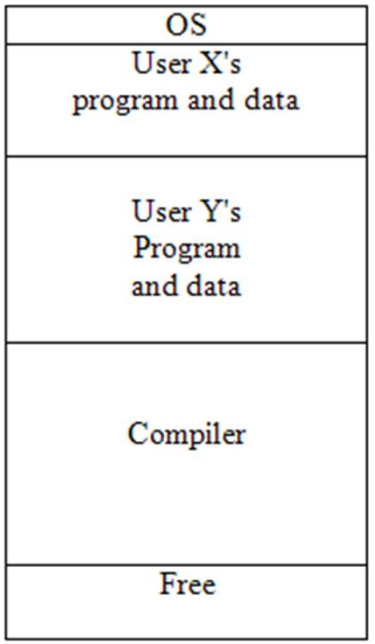
•It will be a waste of memory to load the same compiler twice.

•The OS will load only one instance of the compiler but allow it to work with two separate jobs at the same time.

•The source code jobs will keep their code and data in separate memory locations but will access the same compiler in turns.

•Jobs that behave in this manner (the compiler job) are called "reentrants".

•Definition: Code written so that it can be shared by several programs at the same time.



3.1 The Functions of Operating Systems

3.1.5 MEMORY MANAGEMENT *(Continued…)*

PAGING

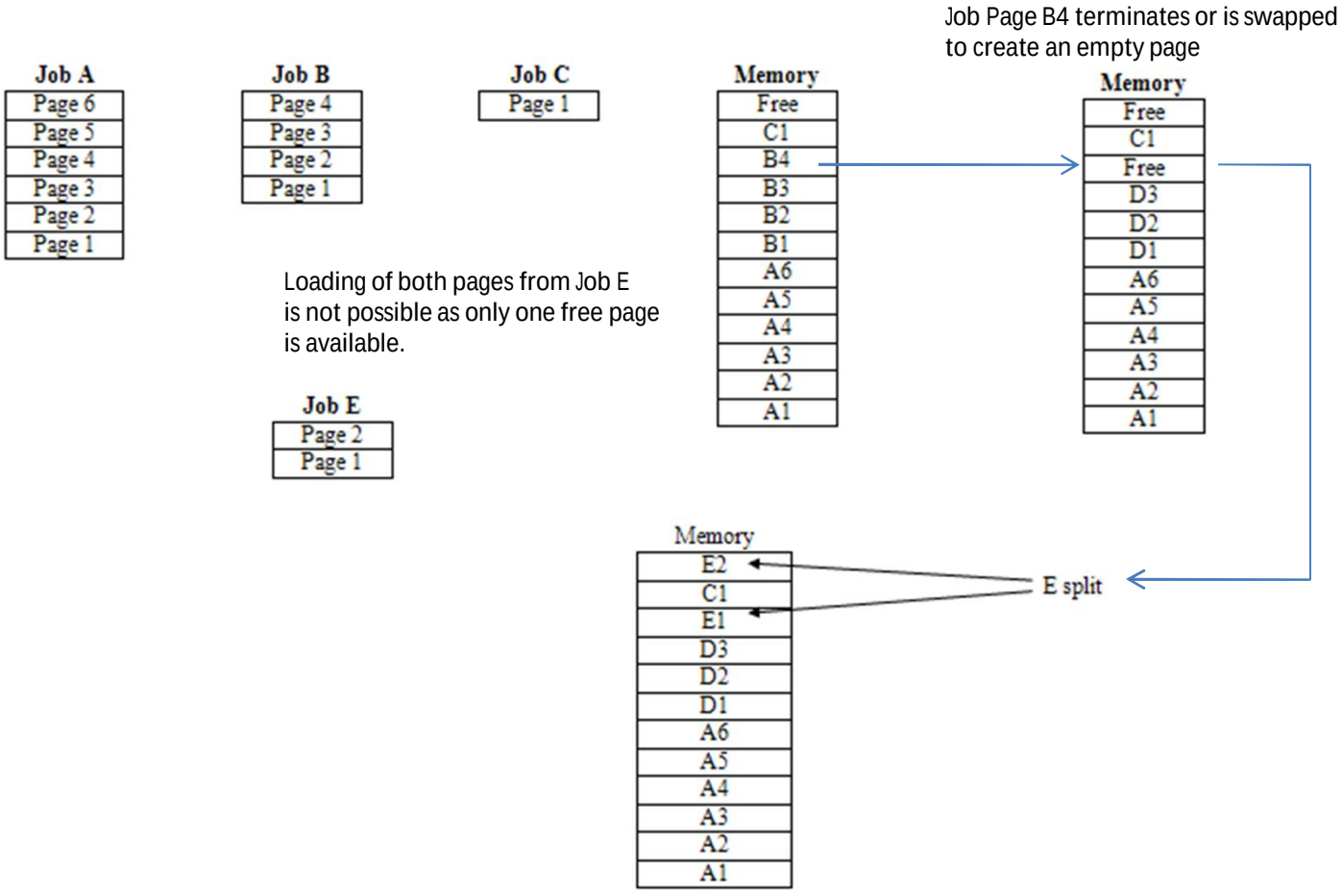
•Paging is a technique in which memory is divided into equal sized pages.
•Jobs are loaded in shape of pages into the main memory.
•These pages represent features/functionality of the job that is currently being used.
•If the job requires a feature/functionality that is present in an another page, then that page is loaded into memory.

- •If the space allocated for jobs is full and the required page can not be loaded, an existing page is unloaded from the memory to make space.
- •If the page being unloaded is 'dirty', i.e. it has been changed since the last save, it is transferred to the auxiliary storage (usually the hard disk).
- •Otherwise, it is just discarded, as the same page can be loaded from the saved pages.

•Paging is an important part of implementing 'virtual memory' (VM) technique.
•VM is used for increasing addressable RAM by using a portion of auxiliary storage (usually the hard disk) as an extension of RAM.
•A 'page fault' occurs when a job is looking for a page that is not currently in the RAM.

3.1 The Functions of Operating Systems

PAGING *(Continued…)*



3.1 The Functions of Operating Systems

PAGING *(Continued…)*

•Paging does not require consecutive memory space to store pages. This is helpful as pages can be loaded wherever the space is available.
•Address calculation is easy as each page is of fixed size.

•Some pages can not be unloaded as they contain main features/functionality or data that is currently being work on.
•With high number of pages being 'swapped' between the main memory and the auxiliary storage (e.g. hard disk), large number of addresses have to be recalculated.
•Also, 'page faults' and pages becoming corrupt is possible.
•The constant use of auxiliary storage (hard disk) for swapping pages is called 'thrashing'.

3.1 The Functions of Operating Systems

PAGING *(Continued...)*

•In Paging, jobs do not have occupy contiguous pages. This provide better memory management but introduces the problem of 'address calculation'.
•A special table is maintained that shows which memory pages are being used for which job pages.
•Each address in a job consists of a page number and the distance of the required location from the start of the page. This enables the OS to perform the 'conversion' of address.
•Example:
In Job A, an instruction refers to location 46 in page 5.

5	46

•Page A5 is stored in memory page 8. Thus:

5	46	Becomes	8	46

Job Page	Memory Page
A1	4
A2	5
A3	6
A4	7
A5	8
A6	9

3.1 The Functions of Operating Systems

SEGMENTATION

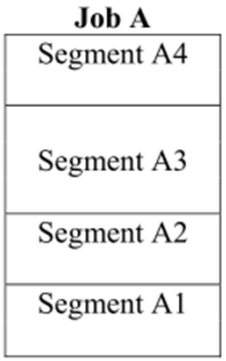
•Paging uses fixed length slots. This wastes a lot of memory as it is hard to write/design code according to the page size.
•An alternative technique is to use 'variable length' slots. These slots are called 'segments'.
•The segment size depends on the functionality of the segment rather than a fixed size value.

•Definition: In segmentation, programmers divide jobs into segments, usually of different sizes. These segments usually contain data, subroutines or groups of related subroutines.

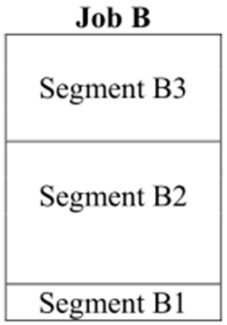
•Since segments may be of different size, address calculation must be done carefully.
•Unlike Page Table, the Segment Table not only contains the starting address of each segment in the memory, but also its size.
•Storing segment size in the table ensures that an address doesn't go 'out of range'.

3.1 The Functions of Operating Systems

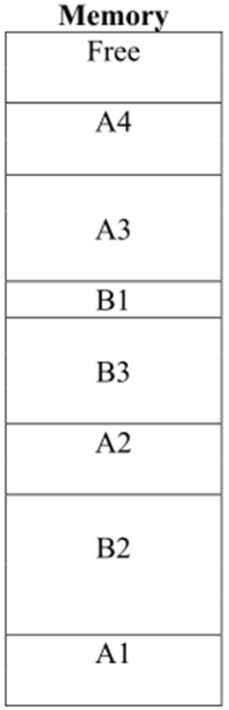
SEGMENTATION *(continued...)*



Job A has 4 segments.



Job B has 3 segment.



These segments when loaded into the memory are arranged as above.

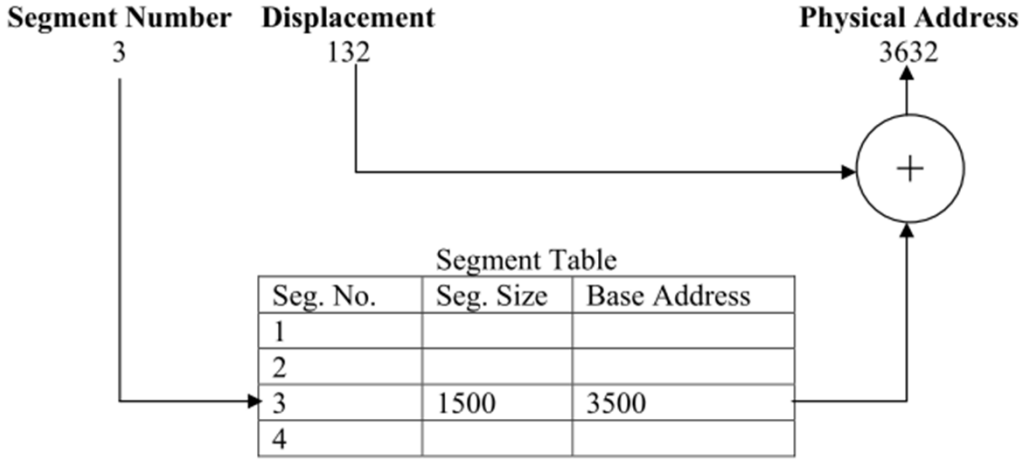
3.1 The Functions of Operating Systems

SEGMENTATION *(continued…)*

Example of address calculation:

- An instruction in a code refers to segment number 3 and displacement (distance from the start of the segment) 132.
- The OS will lookup the 'base address' in the 'Segment Table' in the memory of Segment 3.
- The OS will check if the displacement is within the segment size or not.
- If the displacement is within the segment size, the OS will add the displacement to the base address to calculate the actual memory address of the statement.
- If the displacement is outside the segment size, an error is produced.

Diagram shows how physical addresses are produced by adding displacement to the base address of a segment in the memory.

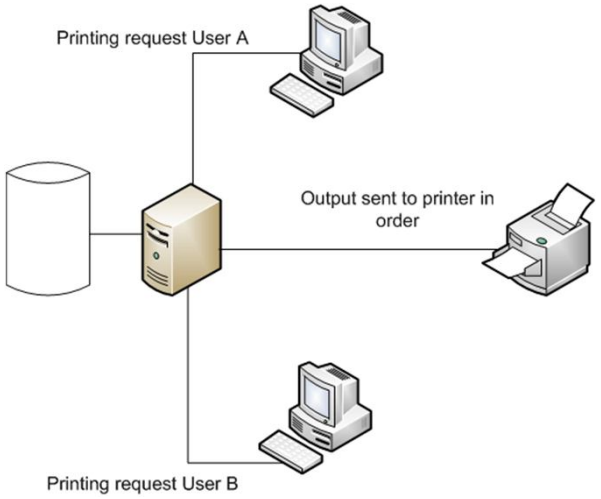


3.1 The Functions of Operating Systems

SPOOLing

•SPOOL = Simultaneous Peripheral Operations Online
•Definition: the input or output is stored on a fast auxiliary storage device (usually hard disk) so that slower peripheral devices do not hold up the processor.
•Systems such as multi-programming, multi-access, multitasking or network OS, employ spooling in order to keep input/output from each job separate while attending to other jobs.

•The input/output is kept in a queue and handled on First Come, First Serve (FCFS) basis.
•IMPORTANT: The I/O itself is not stored on the queue, but its name (pointers) is stored. The input/output itself is kept on the auxiliary storage in a separate area.
•The system utility that handles this process is called a 'spooler'.
•Spooling of printing jobs is handled by 'print spooler'.



3.1 The Functions of Operating Systems

DESKTOP OS AND THE BOOTING PROCESS

•The following is the booting process on Windows/DOS based OS.

1. Control is taken by ROM.
2. Power On Self Test (POST) routine is run.
3. POST clears internal registers (IAS) of the CPU and load the address of the first instruction of POST into Program Counter (PC).
4. Boot program is stored in ROM BIOS and contains the Basic Input/Output System.
5. Boot program takes control and verifies its own integrity and then check POST routine.
6. POST routine checks all available hardware (control/data buses, HD, VDU, RAM, display adapters etc.)
7. At this time, if any hardware has its own BIOS, it is incorporated into the main BIOS is usually copied into RAM for faster access.
8. It generates beep sound codes to indicate the status of the tests.
9. If no errors are found, PC will start the OS loading process.

3.1 The Functions of Operating Systems

DESKTOP OS AND THE BOOTING PROCESS *(continued...)*

- Boot program checks drive A (floppy) to see if the OS is present.
- If not it will look in drive C (hard disk).
- Boot program will look for IO.SYS and MSDOS.SYS
- If found, the boot program loads the boot record of that drive (~512 b)