Endianess

Big Endian	01 23 45 67
Little Endian	67 45 23 01

32 bit integer is 0x01234567

IPC

Semaphore

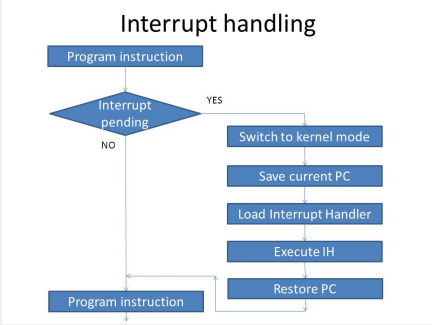
Mutex

Message Passing

Shared Memory

Sockets

Interrupts



Priority Inversion

In a scenario where a low priority task holds a shared resource (example semaphore) that is required by a high priority task. This causes the execution of the high priority task to be blocked until the low priority task has released the resource. This scenario is averted by the OS by increasing the priority of the low priority process until it completes the task and releases the resources

Stack and Heap

Stack Allocation

> The allocation happens on contiguous blocks of memory. It happens in the function call stack. The size of memory to be allocated is known to compiler and whenever a function is called, its variables get memory allocated on the stack. And whenever the function call is over, the memory for the variables is deallocated. Handled by the compiler

Heap Allocation

> The memory is allocated during execution of instructions written by programmers. It is called heap because it is a pile of memory space available to programmers to allocated and de-allocate. If a programmer does not handle this memory well, memory leak can happen in the program.

Detect Endianess

```
void endian()
{
  unsigned int i = 1;
  char c = (char)&i;
  if (*c)
     printf("Little endian");
  else
     printf("Big endian");
}
```

RTOS

An RTOS will have a deterministic scheduler.
For any given set of tasks your process will always execute every number of microseconds or milliseconds exactly, and the same number from schedule to schedule.
OS services consume only known and expected amounts of time.
In UNIX or Windows the scheduler are usually soft RT (as opposed to some hard real time RTOS). Soft realtime means that the scheduler tries to assure your process runs every X number of milliseconds, but may fail to do so on some occasions.
Modern RTOSs simply make sure that a) no interrupt is ever lost, and b) no interrupt can be blocked by a lower priority process.

Memory Management

Paging	Blocks of memory that are stored in auxiliar memory and replaced in main memory when a program needs it.
Caching	Data is temporarily stored in high speed memory for faster access.
Segmentation	Segmentation is a memory management scheme. This is the technique used for memory protection. Any accesses outside premitted area would result in segmentation fault.
Virtual Memory	This technique enables noncontiguous memory to be accessed as if it were contiguous. Same as paging

Stack Growth Direction

```
void checkStack()
{
   int i=2;
   int j=3;
   if(&i > &j) printf("downwards");
   else printf("upwards");
}
```

Define 2 local variables one after other and compare their addresses.