Paper 11 2G/Dip OSfdns 2003 on Part III of course JMB

a) Examples of potentially concurrent access to writeable data:
- hardware-software synchronisation
  wait or sleep (anEvent) } are methods which change (R/W)
  signal or wakeup(") } process state.
- user-level call into OS for I/O } mutual exclusion + synchronisation needed
  bottom-up, interrupt driven I/O } w.r.t. I/O buffers.

<DIAGRAMS> + extended discussion.

b) Forbidding interrupts is only an option on a uniprocessor
On a multiprocessor we still have simultaneous execution.
Even on a uniprocessor forbidding interrupts is only feasible at low level of the OS.

c) read-and-clear register, flag-in-memory
- can be used to change the value of flag from 1 to 0 in a single instruction.
- after execution, if register = 0 flag was already set
  if " = 1 ~~you~~ I set it.

- for mutual exclusion - initialise flag to 1.
  entry protocols for critical regions clear flag
  exit " set flag to 1 again.
- for condition synch, process making condition true sets flag to 1 (initialised to 0). Waiting process loops waiting to set flag back to 0

d) ~~wait~~ ~~signal~~ (sem) if sem > 0 then sem := sem - 1 else queue process on sem
- Signal (sem) free a queued process (if any) else increment sem
- wait and signal operations must be atomic as they R/W shared data (sem value and queue).
- associate a boolean with each semaphore and flip it using read-&-clear (preferable to a single boolean on all semaphore management). processes must busy-wait as in c) to execute semaphore methods.

e) mutual exclusion
initialise lock : 1
entry protocol for CR is wait(lock)
exit protocol is signal(lock)

condition synch
initialise cond = n (n = number of resources, n=1 if 2-process synch)
wait(cond) when acquiring a resource
Signal(cond) when releasing " "
processes block on wait(cond) when resource is exhausted.