1.

a)

context switching between processes

Interrupt/system call
Step 1: save process 1 state to PCB0
Step 2: load process 2 state from PCB1
execute process 1
Interrupt/system call
Step 4: save process 2 state to PCB1
Step 5: load process 1 state form PCB0

b)

context switching between user threads in the same processes

Thread 1 is blocked
Step 1: store thread 1’s registers in thread table
Step 2: find a ready thread (thread 2) on the thread table
Step 3: load reload the machine registers with thread 2’s saved values
Thread 2 executes

c)

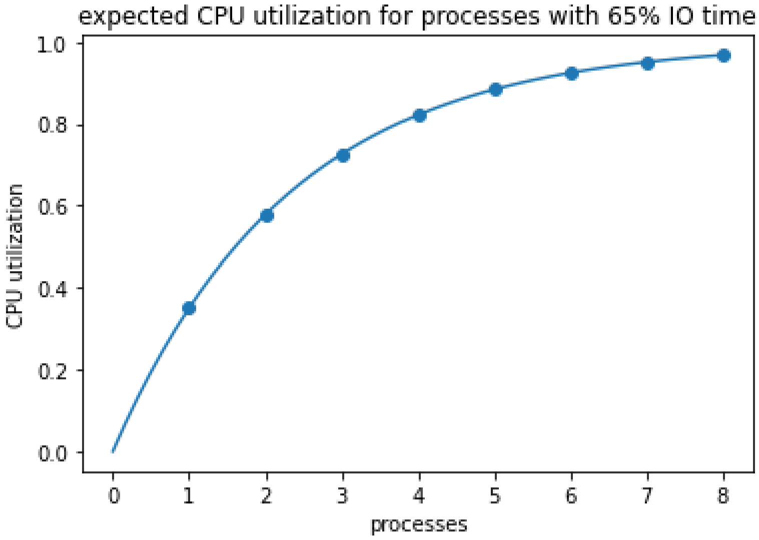
context switching between kernel threads in the same processes

Same as b) but the thread table is implemented at the kernel level so blocking requires a system call, this also allows for switching to threads from other processes.

2.

Using the equation given in class for probabilistically estimating the utilisation of a single CPU under varying degrees of IO-boundness, plot the expected CPU utilisation curve when you have 1 to 8 processes in memory each with a 65% IO time. Ensure you label your axis correctly.

CPU utilization = 1 - IO time ^ num processes



3.

a)

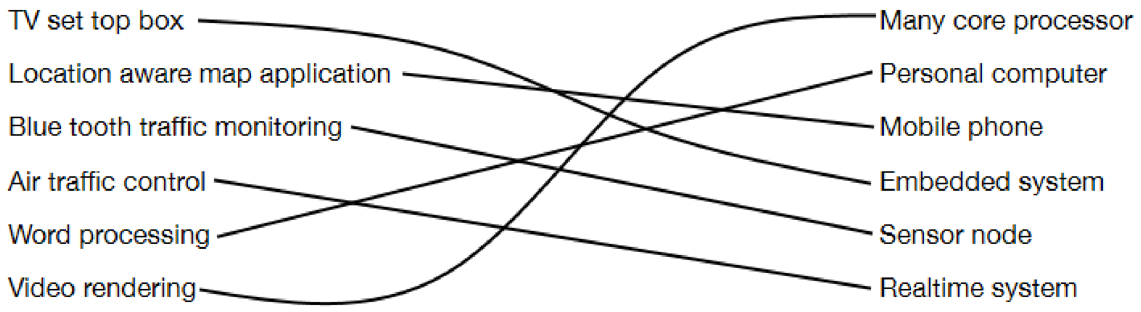
16 processes

b)

6 processes

4.

Match the application to the most appropriate computing system:



5.

In a real time system there are processes that must be completed at specific times. Whereas in a general purpose computer tasks can be delayed to more efficiently use the hardware. When choosing which task to run, real time systems need to consider how long each task will take and which tasks need to be completed soonest. General purpose computers should spread delays between users but don't need to consider task deadlines.