Q1.
4 CPU Cores

Memory Usage: 8142244 kB
MemFree: 3214684 kB
% Memory Free ~ 39.5%

No. Context Switches Since Bootup 359096266
No. of Processes forked since bootup 1654687

Q2.

Code	Bottleneck Resource	Reasoning in identifying Bottleneck	Justification from code
cpu1.c	CPU	top command shows the process using around 100% CPU	Code does large number of arithmetic computations (multiplications)
cpu2.c	CPU	top command shows the process using around 100% CPU	Code performs large number of function calls, resulting in large number of computations on the CPU (as it calculates the time using the Time Stamp Counter)
cpu1print.c	I/Os	htop command shows IO Writes column of gnome-terminal process as the bottleneck when this program is running	This is printing to terminal repeatedly. This also does a large number of computations. However, printing (I/O) is the bottleneck because computations happen faster than I/O (syscall, context switch, etc.)
disk.c	Disk I/Os	'%util' seen in 'iostat -x 1' is high when the code is running	The program reads several files from the disk and does very few computations. This means that fetching data from the disk would be a bottleneck (rather than computation) because of the seek times are several magnitudes higher than cpu clock cycle times
disk1.c	Disk I/Os	'%util' seen in 'iostat -x 1' is high when the code is running	The program reads several files from the disk and does very few computations. Hence, the same reasoning as above applies

Q3.

Code	User Mode	Kernel Mode
cpu1.c	2753	1
cpu1print.c	20	123
cpu2.c	3073	0

cpu1print spends most of its time writing to the terminal which is done through system calls .

The other 2 programs don't use any system calls and hence spend negligible time in Kernel Mode.
Note- in cpu2 gettimeofday() is used but this doesn't not result in a system call and is computed directly using the CPU Clock.

Q4.

Code	Voluntary Context Switches	Involuntary Context Switches
cpu1.c	1	2399
disk.c	25367	456

'cpu1' has mostly Involuntary context switches while 'disk' has mostly voluntary context switches.

When going through the code, we see that 'cpu1.c' does mostly computation. This means that all context switches would result mainly due to the operating system.

'disk.c' does a large number of disk accesses. Hence, there are a large number of voluntary context switches because the program itself gives up access to CPU for disk accesses. The Involuntary Context Switches result from the context switching by the OS.