B033040009 資工系 二年級 蔡宜勳

1. (10%) Explain what memory-mapped I/O is and how it works.

Ans.
Memory-mapped I/O is a complementary method of performing I/O between the CPU and peripheral devices in a computer.
It uses the same address bus to address both memory and I/O devices – the memory and registers of the I/O devices are mapped to address values. So when an address is accessed by the CPU, it may refer to a portion of physical RAM, but it can also refer to memory of the I/O device. Thus, the CPU instructions used to access the memory can also be used for accessing devices. Each I/O device monitors the CPU's address bus and responds to any CPU access of an address assigned to that device, connecting the data bus to the desired device's hardware register. To accommodate the I/O devices, areas of the addresses used by the CPU must be reserved for I/O and must not be available for normal physical memory. The reservation may be permanent or temporary.

2. (10%) Explain what DMA is and how it works.

Ans.
DMA means “direct memory access”, which is a method that allow an I/O device to send or receive data directly to or from the main memory, bypassing the CPU to speed up memory operations.

DMA controller is an unit that control the whole transmission process. While operating DMA transmission, the CPU send request to the DMA controller, wait for the DMA controller done its job and give permission back to the CPU.

A DMA controller can generate memory addresses and initiate memory read or write cycles. It contains several hardware registers that can be written and read by the CPU. These include a memory address register, a byte count register, and one or more control registers. The control registers specify the I/O port to use, the direction of the transfer, the transfer unit, and the number of bytes to transfer in one burst.

To carry out an input, output or memory-to-memory operation, the host processor initializes the DMA controller with a count of the number of words to transfer, and the memory address to use. The CPU then sends commands to a peripheral device to initiate transfer of data. The DMA controller then provides addresses and read/write control lines to the system memory. Each time a byte of data is ready to be transferred between the peripheral device and memory, the DMA controller increments its internal address register until the full block of data is transferred.

DMA transfers can either occur one byte at a time or all at once in burst mode. If they occur a byte at a time, this can allow the CPU to access memory on alternate bus cycles – this is called cycle stealing since the DMA controller and CPU contend for memory access. In burst mode DMA, the CPU can be put on hold while the DMA transfer occurs and a full block of possibly hundreds or thousands of bytes can be moved. When memory cycles are much faster than processor cycles, an interleaved DMA cycle is possible, where the DMA controller uses memory while the CPU cannot.

In a bus mastering system, the CPU and peripherals can each be granted control of the memory bus. Where a peripheral can become bus master, it can directly write to system memory without involvement of the CPU, providing memory address and control signals as required. Some measure must be provided to put the processor into a hold condition so that bus contention does not occur.

B033040009 資工系 二年級 蔡宜勳

3. Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

Process	Burst Time	Priority
P1	8	4
P2	1	1
P3	2	3
P4	1	5
P5	6	4

The processes are assumed to have arrived in the order P 1 , P 2 , P 3 , P 4 , P 5 , all at time 0.
(a) (5%) Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, a non-preemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1) scheduling.
(b) (5%) What is the turnaround time of each process for each of the scheduling algorithms in part 3a?
(c) (5%) What is the waiting time of each process for each of the scheduling algorithms in part 3a?
(d) (5%) Which of the schedulers in part 3a results in the minimal waiting time (over all processes)?

Ans.
(a)
FCFS

1	2	3	4	5
8	1	2	1	6

RR

1	2	3	4	5	1	3	5	1	5	1	5	1	5	1	5	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2

SJF

2	4	3	5	1
1	1	2	6	8

Priority

2	3	1	5	4
1	2	8	6	1

(b)

	FCFS	RR	SJF	Priority
P1	8	18	18	11
P2	9	2	1	1
P3	11	7	4	3
P4	12	4	2	18
P5	18	16	10	17

B033040009 資工系 二年級 蔡宜勳

(c)

	FCFS	RR	SJF	Priority
P1	0	10	10	3
P2	8	1	0	0
P3	9	5	2	1
P4	11	3	1	17
P5	12	10	4	11

(d) Shortest Job First scheduler.

4. (10%) A UNIX process has two parts—the user part and the kernel part. Is the kernel part like a subroutine and a coroutine? Why?

If the kernel part, or system call is blocking , I would think the kernel part is like a subroutine. If the system call is not blocking, then both user and kernel parts can execute at the same time, so it acts like a coroutine. Since a blocking coroutine can become a subroutine, but a subroutine cannot execute in same time as the caller, I think it has to be a coroutine.