Ýmir Þórleifsson

HW 8

# 9.1.1

A fast laser printer produces 20 pages per minute, where a page consists of 4000 characters. The system uses interrupt-driven I/O, where processing each interrupt takes 50 μsec.

1. How much overhead will the CPU experience if the output is sent to the printer one character at a time?

Pages \* chars \* time to process = 4,000,000 μsec.

1. Would polling be a better approach than interrupts?

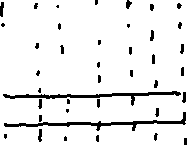
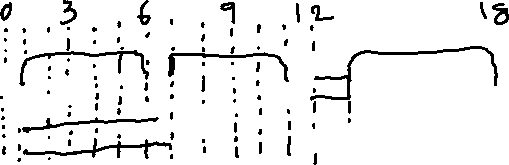
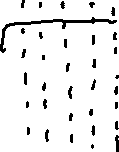
Yes, because it is an overwhelming number of interrupts.

# 9.1.3

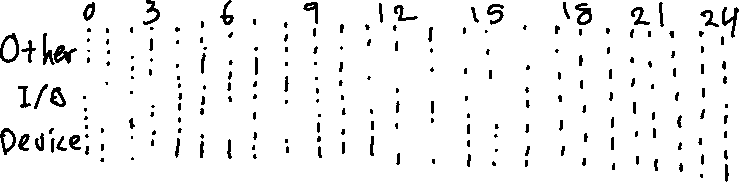
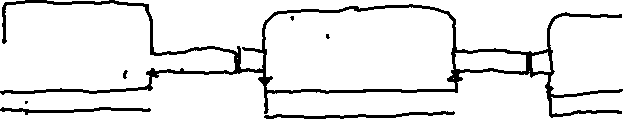
A program is outputting a series of blocks by repeatedly issuing an I/O instruction.

* Executing the I/O instruction takes 1 unit of CPU time. All other instructions take 0 units of time.
* The device starts at the end of the I/O instruction and remains busy for 6 time units.
* When polling is used, the CPU polls the device every 5 time units.
* A poll takes 1 unit of CPU time.
* When interrupts are used, the interrupt handler takes 3 time units to process the interrupt.

1. Draw a timing diagram showing when the CPU and the device are busy during the first 24 time units
2. with polling



1. with interrupts



# 9.1.4

To input a sequence of data items, the driver executes the loop:

repeat

write parameters for input operation

write opcode

repeat

read busy until FALSE

read status

if status = OK

copy data item from input buffer

else exit

1. Write the analogous loop for output.

repeat  
 copy data item from main memory to control buffer  
 write parameters for output operation  
 write opcode  
 repeat  
 read busy until FALSE  
 if status = OK  
 continue with other process  
 else exit

# 9.2.1

When the time to produce a data item, P, is equal to the time to consume the data item, C, then buffer swapping with 2 buffers keeps both the consumer and the producer busy at all times.

The numbers in the diagram indicate which buffer is being used.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Producer | 1 | 2 | 1 | 2 | 1 | 2 |
| Consumer |  | 1 | 2 | 1 | 2 | 1 |

1. If the time to consume is longer than the time to produce, then multiple concurrent consumers can be used to keep the producer busy at all times. Determine the number of concurrent consumers and the number of buffers needed when C = 2P and show the corresponding timing diagram.

Number of buffers would be the same as the factor between C and P, 2 in this case, plus one for the transition step, 3 buffers.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Producer | 1 | 2 | 3 | 1 | 2 | 3 |
| Consumer |  | 1 | 1 | 3 | 3 | 1 |
| Consumer |  |  | 2 | 2 | 1 | 1 |

1. Determine the number of consumers necessary to keep the producer busy for the general case when C is a multiple of P.

C = P – 1.

1. Determine the number of consumers necessary to keep the producer busy when C is less than P.

The same as the number of buffers.

# 9.3.2

A disk has 800 tracks per surface, 180 sectors per track, and 512 bytes per sector. The disk rotates at 5000 rpm (rotations per minute). The average seek time between random tracks is 10 ms. The seek time between adjacent tracks is 2 ms.

1. Determine the total storage capacity of the disk.

512\*180\*800 = 73,728,000 bytes ~ 73.7 MB

1. Determine the number of seek operation to read the entire disk.

799

1. Determine the peak data rate of the disk (MB/second).

5000 rpm = 83.333 rps, 1 revolution = 1/83.333 = 0.012 sec

Each track = 73.7 MB / 800 = 0.092 MB

Peak data rate = 7.68 MB/s per track

1. If the r/w head is at track t, how long will it take to read a sector on track t + 1?

2 ms + 12 ms = 14 ms

1. Determine the sustained data rate of the disk for accessing blocks distributed randomly over the disk (MB/second).

n \* (0.092 MB / 0.022 s), where n is number of blocks.

4.2n MB/s

# 9.3.4

The r/w head of a disk is at track 143. The previous position was track 0. Requests to access the following tracks have arrived:

143, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

1. In which order will the tracks be visited using: FIFO, SSTF, Scan, C-Scan.
2. Starting from track 143, determine the number of tracks traversed by the r/w head under each algorithm to service all requests.

* FIFO: 143, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Tracks traversed = 7081

* SSTF: 143, 130, 86, 913, 948, 1022, 1470, 1509, 1750, 1774

Tracks traversed = 1745

* Scan: 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 130, 86

Tracks traversed = 3319

* C-Scan: 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 0, 86, 130

Tracks traversed = 3535

# 9.4.1

To encode 8-bit bytes, Hamming code requires 4 parity bits at positions 1, 2, 4, and 8.

The table shows which position are covered by which parity bit.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|  | p1 | p2 | d1 | p4 | d2 | d3 | d4 | p8 | d5 | d6 | d7 | d8 |
| p1 | x |  | x |  | x |  | x |  | x |  | x |  |
| p2 |  | x | x |  |  | x | x |  |  | x | x |  |
| p4 |  |  |  | x | x | x | x |  |  |  |  | x |
| p8 |  |  |  |  |  |  |  | x | x | x | x | X |

1. Encode the strings:
2. 1 0 1 0 1 0 1 0 Hamming code: **1** **1** 1 **1** 0 1 0 **0** 1 0 1 0
3. 1 1 1 1 1 1 1 1 Hamming code: **1** **1** 1 **1** 1 1 1 **0** 1 1 1 1
4. 0 0 0 0 1 1 1 1 Hamming code: **0 0** 0 **1** 0 0 0 **0** 1 1 1 1
5. Determine which bit, if any, is incorrect in the strings below. Show the corrected data, without the parity bits.
6. 1 1 1 0 1 1 1 0 1 1 0 1

p1 = gives error p2 = gives error p4 = checks out p8 = gives error

1+2+8 = 11, bit nr. 11 is incorrect: 1 1 1 0 1 1 1 0 1 1 1 1

Decoded = 1 1 1 1 1 1 1 1

1. 0 0 0 1 0 0 0 0 1 1 1 1

p1 = checks out p2 = checks out p4 = checks out p8 = checks out

Decoded = 0 0 0 0 1 1 1 1

1. 1 1 1 0 0 1 0 0 1 0 1 0

p1 = checks out p2 = checks out p4 = gives error p8 = checks out

Only p4 is incorrect, bit 4. 1 1 1 1 0 1 0 0 1 0 1 0

Decoded = 1 0 1 0 1 0 1 0

1. How many data bits can be encoded before another parity bit is needed?

When the Hamming code becomes a new multiple of 2. In this case we can add 3, and the Hamming code becomes 15 bits.

# 9.4.2

With sector forwarding, bad blocks are mapped to successive sectors in the spare sectors area.

Assume that block 15 fails and is mapped to spare 1. Next, block 10 fails and is mapped to spare 2.

1. How many revolutions are needed to sequentially access all sectors on the track?

2 revolutions.

1 revolution: b[0] - b[15] are read.

2 revolution: b[16] and onwards are read.

1. How many revolutions would be needed for the same task if block 10 failed before block 15?

3 revolutions.

1 revolution: b[0] - b[10] are read.

2 revolution: b[10] - b[15] are read.

3 revolution: b[16] and onwards are read.

1. How many revolutions would be needed for the same task if sector slipping were used instead of sector forwarding?

1 revolution. b[0] – b[10] are read, then continue only shifted by 1, b[11 + 1] – b[14 + 1], then continue shifted again by 1, b[15 + 2] and onwards.