

**Problem 1 : Block mapping.**

(a)  $12 \text{ blocks} \times 4 \text{ KB/block} = 49,152 \text{ bytes} = 48 \text{ KB}$ .

(b) There are  $(4 \text{ KB} / 32 \text{ bits}) = 1024$  block pointers in each indirect block. Therefore the single indirect block adds  $(1024 \text{ blocks} \times 4 \text{ KB/block}) = 4,194,304 \text{ bytes} = 4 \text{ MB}$ , for a total of 4,243,456 bytes (4.047 MB).

(c) The double indirect block adds  $(1024 \times 1024 \times 4 \text{ KB}) = 4,294,967,296 \text{ bytes} = 4 \text{ GB}$ , for a total of 4,299,210,752 bytes (4.004 GB).

(d) The triple indirect block adds  $(1024 \times 1024 \times 1024 \times 4 \text{ KB}) = 4,398,046,511,104 \text{ bytes} = 4 \text{ TB}$ , for a total of 4,402,345,721,856 bytes (4100 GB).

(e) Now there are  $(1 \text{ KB} / 32 \text{ bits}) = 256$  block pointers in each indirect block. So:

- Direct:  $12 \text{ blocks} \times 1 \text{ KB/block} = 12,288 \text{ bytes} = 12 \text{ KB}$ ;
- Single:  $256 \times 1 \text{ KB} = 262,144 \text{ bytes} = 256 \text{ KB}$ ;
- Double:  $256 \times 256 \times 1 \text{ KB} = 67,108,864 \text{ bytes} = 64 \text{ MB}$ ;
- Triple:  $256 \times 256 \times 256 \times 1 \text{ KB} = 17,179,869,184 \text{ bytes} = 16 \text{ GB}$ ;

For a total of 17,247,252,480 bytes (16.06 GB).

(f) The quadruple indirect block adds  $(1024 \times 1024 \times 1024 \times 1024 \times 4 \text{ KB}) = 4,503,599,627,370,496 \text{ bytes} = 4 \text{ PB}$ , for a total of 4,508,001,973,092,352 bytes (4,198,404 GB).

(g) Because the block pointers are unsigned 32-bit values, you are limited to addressing only  $2^{32}$  unique blocks. This gives a maximum file size of  $2^{32} \times 4 \text{ KB} = 17,592,186,044,416 \text{ bytes} = 16,384 \text{ GB} = 16 \text{ TB}$ .

(g) Signed 32-bit numbers can address up to  $2^{31} = 2,147,483,648 \text{ bytes} (2 \text{ GB})$ . Note that these parameters address byte offsets, not block offsets.

**Problem 2 : Extents.**

(a) 32 bits for the block addresses means the file system is limited to  $2^{32}$  blocks, or a maximum of 2 TB. This could be addressed with a single extent.

(a – Alternate) Assuming you don't need to address blocks other than the first block in an extent, you could use two extents: one of length  $2^{32} - 1$  starting at block 0, and the other with length  $2^{32}$  blocks starting at block  $2^{32} - 1$ , for a total of 4 TB-1.

(b) If each extent is only one block long (e.g., a highly fragmented file), the minimum size is  $(5 \times 512 \text{ B}) + 1 \text{ byte} = 2561 \text{ bytes}$ .

(c) Yes. As shown in (b), when disks are highly fragmented, the OS may be forced to allocate very short extents. It would then be necessary to use indirect blocks (i.e., pointers to extent lists) for large files.

**Problem 3 : Polling, Interrupts, and DMA.**

(a) 100% Disk Utilization

- $\frac{\frac{20\text{MB/s}}{32\text{B}} \cdot 500}{500\text{MIPS}} = 65\%$
- $\frac{\frac{20\text{MB/s}}{32\text{B}} \cdot 500}{500\text{MIPS}} = 65\%$
- $\frac{4\text{KB}}{20\text{MB/s}} = 19.5\mu\text{s}, \frac{(1500+500\text{cycles/transfer})}{(19.5\mu\text{s/transfer})} \cdot \frac{1}{500\text{MIPS}} = 2\%$

(b) 10% Disk Utilization

- Since the CPU still has to poll at the same rate (so it is sure not to miss any transfers) and it still consumes 500 cycles per poll as overhead, the CPU utilization is going to be the same as in (a) at 65%.
- $10\% \cdot 65\% = 6.5\%$
- $10\% \cdot 2\% = 0.2\%$

- (c)
- Polling and Programmed I/O: Polling is useful for very fast event notification. In the case one is constantly polling, one will find that the event has occurred as soon as it does. However polling consumes a lot of CPU time especially if one doesn't know when to expect and event to occur. Programmed I/O is simpler to use, but also requires the CPU to be involved in much of the data movement.
  - Interrupts and Programmed I/O: Interrupts offload the CPU from having to constantly poll at the expense of a context-switch.
  - Interrupts and DMA: DMA has the benefit of offloading much of the data movement from the CPU (the CPU tells the DMA engine how much data to move and where to move it from/to). However, it does this at the expense of added complexity.

**Problem 4 : Those that do not learn from the mistakes of history are doomed to repeat them.**

(a)  $1024 \text{ cylinders} * 256 \text{ heads/cylinder} * 63 \text{ sectors/head} = 16,515,072 \text{ sectors} = 7.875 \text{ GB}.$

(b) Another way of looking at the table is:

Standard	Maximum cylinders	Maximum heads	Maximum sectors
ATA	65,536	<b>16</b>	255
Int 13	<b>1024</b>	256	<b>63</b>
Combined	<b>1024</b>	<b>16</b>	<b>63</b>

$1024 \text{ cylinders} * 16 \text{ heads/cylinder} * 63 \text{ sectors/head} = 1,032,192 \text{ sectors} = 0.492 \text{ GB}.$

(c) Because 28 address bits are available:  $2^{28} = 268,435,456 \text{ sectors} = 128 \text{ GB}.$

(d) Set up the following equations, where  $t$  is the number of 18-month periods to reach 128 GB capacity:

$$\begin{aligned} C \cdot 2^0 &= 76.335 \\ C \cdot 2^t &= 128 \end{aligned}$$

$C = 76.335$ , so we solve for  $t$  using logarithms:

$$\begin{aligned} 76.335 \cdot 2^t &= 128 \\ \log_2(2^t) &= \log_2\left(\frac{128}{76.335}\right) \\ t &= \frac{\ln 1.6768}{\ln 2} \\ t &= 0.75 \end{aligned}$$

Therefore, the industry will reach the addressable limit ( $0.75 * 12$  months) = 9 months after August 2000: May 2001.

(e)  $2^{48} = 281,474,976,710,656$  sectors = 134,217,728 GB = 128 PB.

(f) Set up the equations:

$$\begin{aligned} 76.335 \cdot 2^t &= 134,217,728 \\ \log_2(2^t) &= \log_2\left(\frac{134,217,728}{76.335}\right) \\ t &= \frac{\ln 1,758,272}{\ln 2} \\ t &= 20.75 \end{aligned}$$

Therefore, the industry will surpass the addressable limit of the 48-bit ATA LBA extensions ( $20.75 * 12$  months) = 249 months after August 2000: May 2021.

(g)  $2^{64} = 18,446,744,073,709,551,616$  sectors = 8,796,093,022,208 GB = 8 ZB (no kidding).

(h) Set up the equations:

$$\begin{aligned} 76.335 \cdot 2^t &= 8,796,093,022,208 \\ \log_2(2^t) &= \log_2\left(\frac{8,796,093,022,208}{76.335}\right) \\ t &= \frac{\ln 1,152,30,143,737}{\ln 2} \\ t &= 36.75 \end{aligned}$$

Therefore, the industry will surpass the addressable limit of the INT 13 extensions ( $36.75 * 12$  months) = 441 months after August 2000: May 2037.

The disk drive sizes in this problem may seem ludicrous to you, but imagine how ludicrous an 80 GB hard drive must have seemed in 1981. Who knows—in ten years *you* may be sitting on a standards board that's evaluating options to extend the ATA LBA specification...

## Problem 5 : SCSI Busses.

### Part I

- (a) In a target asynchronous send, the target places data on the bus, that it holds for  $t_3$ ns, after which it asserts the REQ signal. Since we assumed that the delay between the REQ being asserted to the ACK asserted is 0 (thus there is no propagation delay on the wire), the initiator acknowledges the data by immediately asserting the ACK signal. Once the target sees the AC asserted it must hold the data for  $t_4$ ns, after which the target can put the next data byte on the wire (since  $t_4 > t_1$ ).

So maximum transfer rate is:  $\frac{1\text{byte}}{t_3\text{ns}+t_4\text{ns}} = \frac{1\text{byte}}{75\text{ns}} = 13.3 \text{ MB/s}$

### Part II

- (b) The main tradeoff is that between performance and responsiveness to errors. On one hand with a very small offset we have something like asynchronous scsi which is typically slower (because of the handshaking on every request) but very responsive to errors (the transmitter knows immediately when he fails to receive and ack). On the other hand the handshaking is practically eliminated giving very good streaming performance, but the sender may have transmitted a large amount of data before finding out about an error that may have occurred. There is another concern that the receiver can consume data at the given rate.

### Part III

- (c) Asynchronous SCSI – we have four signals (REQ high, ACK high, REQ low, ACK low) that need to propagate along the cable and four signals that incur turn-around time within the chip. These four signals are serialized.

- 1 foot cable:  $(4 \cdot 1.7\text{ns/ft} \cdot 1\text{ft}) + (4 \cdot 40\text{ns}) = 167\text{ns} = 5.71\text{MB/s}$
- 6 meter cable:  $(4 \cdot 1.7\text{ns/ft} \cdot 19.6\text{ft}) + (4 \cdot 40\text{ns}) = 293\text{ns} = 3.25\text{MB/s}$
- 25 meter cable:  $(4 \cdot 1.7\text{ns/ft} \cdot 82\text{ft}) + (4 \cdot 40\text{ns}) = 717.6\text{ns} = 1.33\text{MB/s}$

- (d) Synchronous SCSI

- At 1 Byte per 5 MHz = 5 MB/s
- Since only 1 signal needs to propagate:  
 $(1 \cdot 5.25\text{ns/m} \cdot \text{length}) + (1 \cdot 40\text{ns}) = \frac{1}{5\text{MB/s}} = 200\text{ns}$   
 $(5.25\text{ns/m} \cdot \text{length}) + 40\text{ns} = 200\text{ns}$   
 $\text{length} = \frac{160\text{ns}}{1.7\text{ns/ft}} = 94.11\text{ft} \approx 30\text{m}$

## Problem 6 : I/O System Design.

(a) IOPS for CPU:  $\frac{3000\text{MIPS}}{10,000} = 300,000$  IOPS

IOPS for Mem:  $\frac{\frac{1}{50\text{ns}} \cdot 16}{8\text{KB}} = 39,062$  IOPS

IOPS for I/O bus:  $\frac{33\text{MHz} \cdot 32\text{bit}}{8\text{KB}} = 16,113$  IOPS

IOPS for a SCSI controllers:  $\frac{1}{.2\text{ms} + \frac{8\text{KB}}{320\text{MB/s}}} = \frac{1}{.2\text{ms}} = 5,000$  IOPS

IOPS for a disk:  $\frac{1}{6\text{ms} + \frac{.5}{10,000\text{RPM}} + \frac{8\text{KB}}{25\text{MB/s}}} = \frac{1}{6\text{ms} + 3\text{ms} + .3\text{ms}} = \frac{1}{9.3\text{ms}} = 108$  IOPS

(b) Fully configured system = 1 CPU, 1 PCI Bus, 8 Controllers with 7 drives each.

7 Drives =  $7 \cdot 108 = 756$  IOPS

8 Controllers = (with 7 of current drives)  $7 \cdot 756 = 6048$  IOPS, max =  $7 \cdot 3,356 = 23,489$  IOPS

So with current drives it seems that the drives are the bottleneck, everything else is underutilized. But if the I/O bus can not handle the controllers at their maximum throughput – so the bus will become a bottleneck.

(c) To maximize performance we need as many disks as possible, since they are the major bottleneck. So if we had 56 disks (8 controllers \* 7 disks) we would have a total of  $56 \cdot 108$  IOPS = 6,048 IOPS. We know that a single controller can easily handle 7 drives and since  $6,048 < 6,113$  (the max of the PCI bus), our system can handle this load.

The (somewhat unreasonable) cost of this system is:

$8 \cdot \$350 + 56 \cdot (36 \cdot \$4) = \$2,800.00 + 56 \cdot \$144 = \$10,864.00.$

(d) IOPS for new disks:  $\frac{1}{4\text{ms} + \frac{.5}{15,000\text{RPM}} + \frac{8\text{KB}}{30\text{MB/s}}} = \frac{1}{4\text{ms} + 2\text{ms} + .26\text{ms}} = \frac{1}{6.26\text{ms}} = 160$  IOPS

IOPS for new bus:  $\frac{66\text{MHz} \cdot 64\text{bit}}{8\text{KB}} = 64,453$  IOPS

IOPS for new CPU: (I messed up and used 1000MIPS instead of 10,000MIPS, there's no way a slower processor would be a good idea unless it might be cheaper)

For the system in (c) the new disks would be most beneficial.