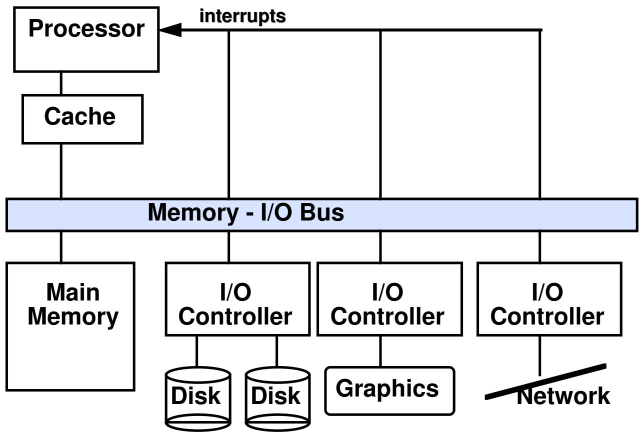
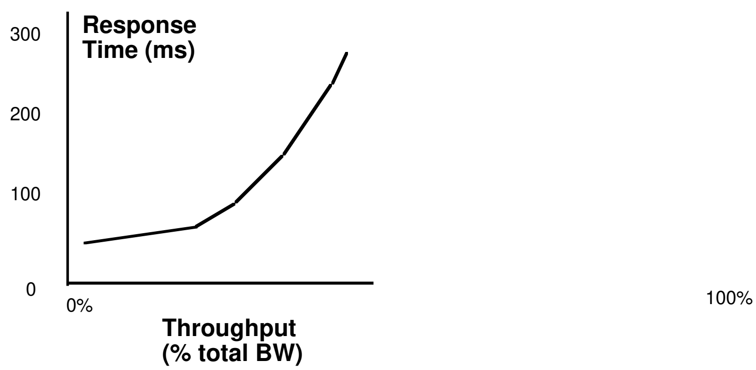
I/O Systems

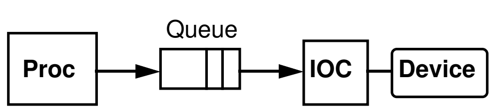


Time(workload) = Time(CPU) + Time(I/O) - Time(Overlap)

Disk I/O Performance

Metrics: Response Time and Throughput





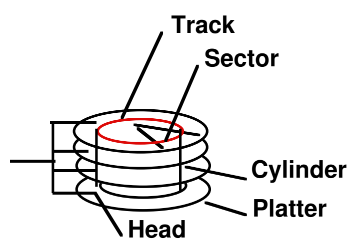
Response time = Queue + Device Service time

Storage System Issues

- Historical Context of Storage I/O
- Storage I/O Performance Measures
- Secondary and Teriary Storage Devices
- A Little Queuing Theory
- Processor Interface Issues
- I/O Buses
- Redundant Arrarys of Inexpensive Disks (RAID)
- ABCs of UNIX File Systems
- I/O Benchmarks
- Comparing UNIX File System Performance

Devices: Magnetic Disks

- Purpose:
  - Long-term, nonvolatile storage
  - Large, inexpensive, slow level in the storage hierarchy
- Characteristics:
- Seek Time (~20 ms avg, 1M cyc at 50MHz)
  - positional latency
  - rotational latency
- Transfer rate
  - about a sector per ms (1-10 MB/s)
  - blocks
- Capacity
  - Gigabytes
  - quadruples every 3 years (areodynamics)



3600 RPM = 60 RPS => 16 ms per rev
ave rot. latency = 8 ms
32 sectors per track => 0.5 ms per sector
1 KB per sector => 2 MB / s
32 KB per track
20 tracks per cyl => 640 KB per cyl
2000 cyl => 1.2 GB

Response time
= Queue + Controller + $\underbrace{\text{Seek + Transfer}}_{\text{Service time}}$

Disk time example

- Transfer size is 8K bytes
- Adversited average seek is 12 ms
- Disk spins at 7200 RPM
- Transfer rate is 4 MB/sec
- Controller overhead is 2 ms
- Assume that disk is idle so no queuing delay
- Average Disk Access Time for a Sector?

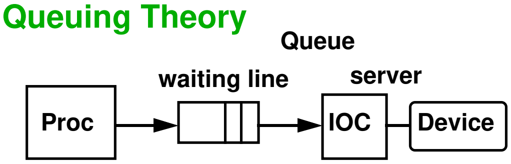
$$avg.\ seek + avg.\ rotational\ \ delay + transfer\ \ time + controler\ \ overhead$$

$$12ms + \frac{0.5}{7200RPM} + \frac{8KB}{4MB/\sec} + 2ms$$

$$12 + 4.17 + 2 + 2ms \approx 20ms$$

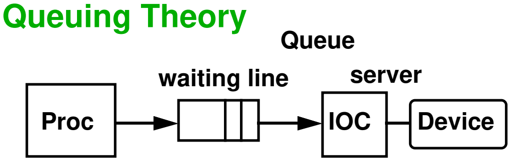
- Advertized seek time assumes no locality: typically 1/4 to 1/3 advertised seek time: 20 ms => 12 ms

A Little Queuing Theory



- Service time completions vs. waiting time for a busy server when randomly arriving event joins a waiting line of arbitrary length when server is busy, otherwise serviced immediately
- A ***single server queue***: combination of a servicing facilty that accomodates 1 customer at a time (***server***) + waiting area (***waiting line***): together called a ***queue***
- Server spends a variable amount of time with customers; ***how characterize variability?***

A Little Queuing Theory

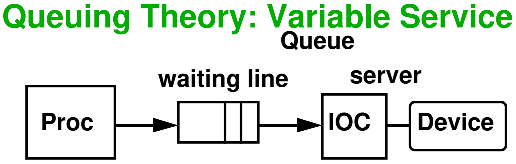


- Server spends a variable amount of time with customers
  - weighted mean ***m1*** = (f1xT1 + f2xT2+...+fnXTn)/F (F=f1+f2+...)
  - ***squared coefficient of variance***: ***C***

    C = variance/m1² → $C = variance/m1^2$

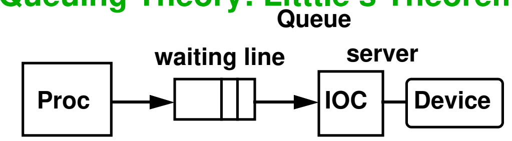
    variance = $(f1xT1^2 + f2xT2^2+...+fnXTn^2)/F - m1^2$
- ***C = 1*** exponential distribution: most short relative to average, few others long; 90% < 2.3 x average, 63% < average
- ***C < 1*** hypoexponential distribution: most close to average, C=0.5 => 90% < 2.0 x average, only 57% < average
- ***C > 1*** hyperexponential distribution: futher from average C=2.0 => 90% < 2.8 x average, 69% < average

A Little Queuing Theory: Variable Service Time



- Server spends a variable amount of time with customers
  - weighted mean ***m1*** = (f1xT1 + f2xT2+...+fnXTn)/F (F=f1+f2+...)
  - squared cooefficient of variance ***C***
- Disk response times ***C*** ≈ ***1.5***
- Yet usually pick ***C = 1.0*** for simplicity
- Another useful value is average time must wait for server to complete task ***m1(z)***
  - not just 1/2 x m1 because doesn't capture variance
  - can derive ***m1(z) = 1/2 x m1 x (1 + C)***
  - ***no variance => C= 0 => m1(z) = 1/2 x m1***

A Little Queuing Theory: Litttle's Theorem



- Queuing models assume state of equilibilirum: input rate = output rate
- Notation:

  r average number of arriving customers/second
  T_s average time to service a customer
  u server utilization (0..1): $u = r \times T_s$
  T_w average time/customer in waiting line
  T_q average time/customer in queue: $T_q = T_w + T_s$
  L_w average length of waiting line: $L_w = r \times T_w$
  L_q average length of queue: $L_q = r \times T_q$

- Little's Law: $r = L_q / T_q = L_w / T_w = u / T_s$
  Mean number customers = arrival rate x mean response time

A Little Queuing Theory: Average Wait Time

- Calculating average wait time T_w
  - if something at server, it takes to complete on average $m1(z)$
  - chance server is busy = u; average delay is u x $m1(z)$
  - Afterward, all customers in line must complete; each avg T_s

  $T_w = u \times m1(z) + L_w \times T_s = 1/2 \times u \times T_s \times (1 + C) + L_w \times T_s$
  $T_w = 1/2 \times u \times T_s \times (1 + C) + r \times T_w \times T_s$
  $T_w = 1/2 \times u \times T_s \times (1 + C) + u \times T_w$
  $T_w \times (1 - u) = T_s \times u \times (1 + C) / 2$
  $T_w = T_s \times u \times (1 + C) / (2 \times (1 - u))$

- Notation:

  r average number of arriving customers/second
  T_s average time to service a customer
  u server utilization (0..1): $u = r \times T_s$
  T_w average time/customer in waiting line
  L_w average length of waiting line: $L_w = r \times T_w$

A Little Queuing Theory: M/G/1 and M/M/1

- Assumptions so far
  - equilibirum
  - time between two success arrivals in line are random
  - server can start on next customer immediately after finish prior
  - no limit to the waiting line: works First-In-First-Out
  - Afterward, all customers in line must complete; each avg T_s
- Described Markovianrequest arrival (M for C=1exponentially random), General service distribution (no restrictions), 1 server: *M/G/1 queue*
- When Service times have C =1, *M/M/1 queue*
  $T_w = T_s \times u \times (1 + C) / (2 \times (1 - u)) = T_s \times u / (1 - u)$

  T_s average time to service a customer

  u server utilization (0..1): $u = r \times T_s$

  T_w average time/customer in waiting line
- Some confusion: waiting time = queue delay?

A Little Queuing Theory: An Example

- Suppose processor sends 10 8KB disk I/Os per second, requests exponentially distrib., disk service time = 20 ms
- On average, how utilized is the disk?
  What is the number of requests in the waiting line?
  What is the average time a spent in the waiting line?
  What is the average response time for a disk request?
- Notation:

  r average number of arriving customers/second= 10

  T_s average time to service a customer= 20 ms

  u server utilization (0..1): $u = r \times T_s$= 10/s x .02s = 0.2

  T_w average time/customer in waiting line

  $= T_s \times u / (1 - u)$ = 20 x 0.2/(1-0.2) = 20 x 0.25 = 5 ms

  T_q average time/customer in queue: $T_q = T_w + T_s$= 25 ms

  L_w average length of waiting line: $L_w = r \times T_w$
  = 10/s x .005s = 0.05 requests in wait line

  L_q average length of "queue": $L_q = r \times T_q$= 10/s x .025s
  = 0.25

A Little Queuing Theory: Another Example

- Suppose processor sends 20 8KB disk I/Os per second, requests exponentially distrib., disk service time = 12 ms
- On average, how utilized is the disk?
  What is the number of requests in the waiting line?
  What is the average time a spent in the waiting line?
  What is the average response time for a disk request?
- Notation:

r average number of arriving customers/second= 20

T_s average time to service a customer= 12 ms

u server utilization (0..1): $u = r \times T_s$= 20/s x .012s = 0.24

T_w average time/customer in waiting line

$= T_s \times u / (1 - u)$ = 12 x 0.24/(1-0.24) = 12 x 0.32 = 3.8 ms

T_q average time/customer in queue: $T_q = T_w + T_s$= 16 ms

L_w average length of waiting line: $L_w = r \times T_w$ = 20/s x .0038s = 0.016 requests in wait line

L_q average length of "queue": $L_q = r \times T_q$= 20/s x .016s = 0.32

cs252 21.13 © Patterson 1994

A Little Queuing Theory: An Example

- Suppose processor sends 10 8KB disk I/Os per second, req. squared coef. var. = 1.5, disk service time = 20 ms
- On average, how utilized is the disk?
  What is the number of requests in the waiting line?
  What is the average time a spent in the waiting line?
  What is the average response time for a disk request?
- Notation:

r average number of arriving customers/second= 10

T_s average time to service a customer= 20 ms

u server utilization (0..1): $u = r \times T_s$= 10/s x .02s = 0.2

T_w average time/customer in waiting line

$= T_s \times u \times (1 + C) / (2\times (1 - u))$ = 20 x 0.2(2.5)/2(1 – 0.2) = 20 x 0.32 = 6.25 ms

T_q average time/customer in queue: $T_q = T_w + T_s$= 26 ms

L_w average length of waiting line: $L_w = r \times T_w$ = 10/s x .006s = 0.06 requests in wait line

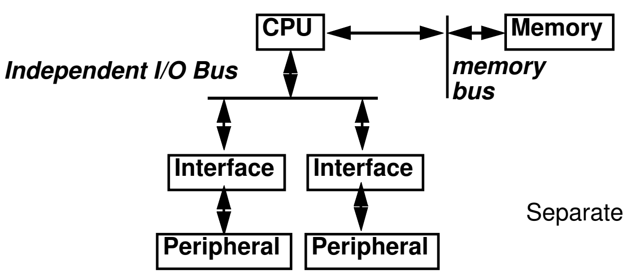
L_q average length of "queue": $L_q = r \times T_q$= 10/s x .026s = 0.26

cs252 21.14 © Patterson 1994

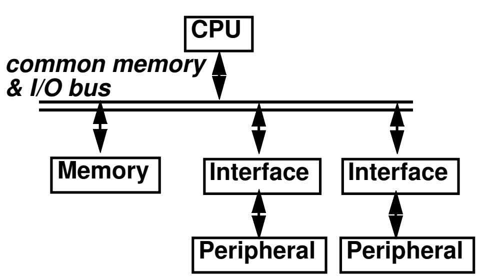
Processor Interface Issues

- **Interconnections**
  - **busses**
- **Processor interface**
  - **interrupts**
  - **memory mapped I/O**
- **I/O Control Structures**
  - **polling**
  - **interrupts**
  - **DMA**
  - **I/O Controllers**
  - **I/O Processors**
- **Capacity, Access Time, Bandwidth**

I/O Interface



Separate I/O instructions (in,out)



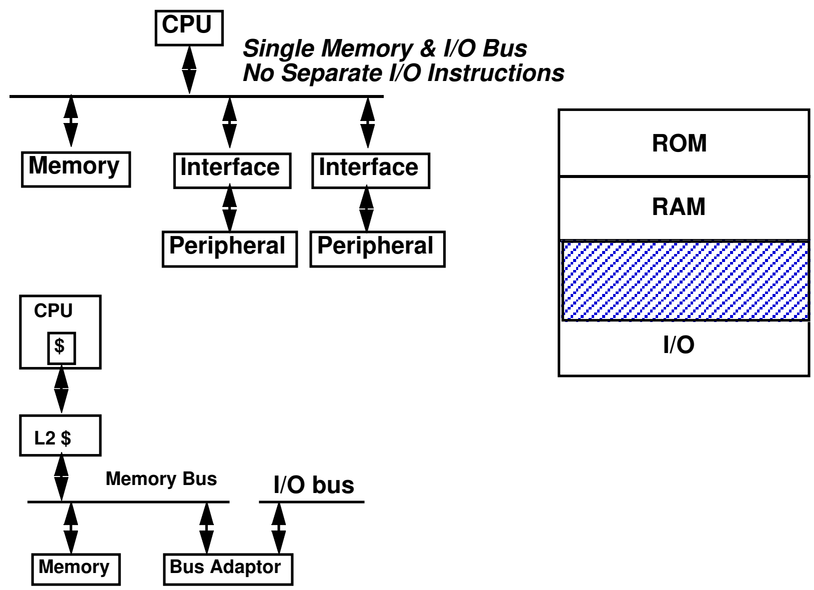
Lines distinguish between
I/O and memory transfers

VME bus
Multibus-II
Nubus

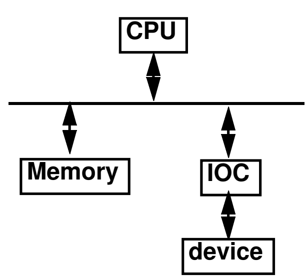
40 Mbytes/sec
optimistically

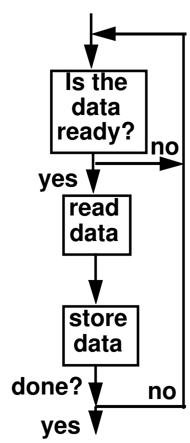
10 MIP processor
completely
saturates the bus!

Memory Mapped I/O



Programmed I/O (Polling)

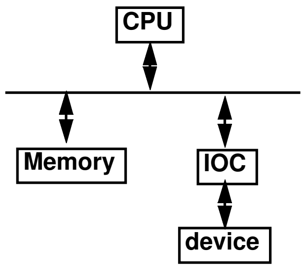


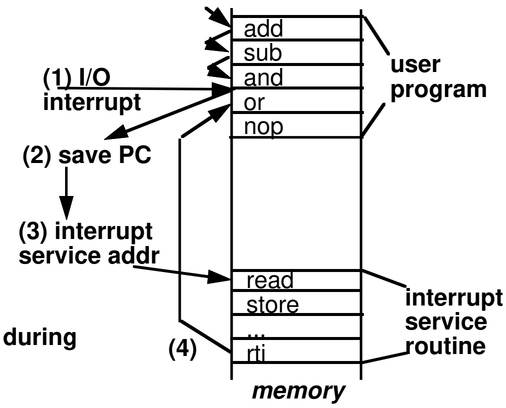


busy wait loop not an efficient way to use the CPU unless the device is very fast!

but checks for I/O completion can be dispersed among computation intensive code

Interrupt Driven Data Transfer





User program progress only halted during
actual transfer

1000 transfers at 1 ms each:
- 1000 interrupts @ 2 usec per interrupt
- 1000 interrupt service @ 98 usec each = 0.1 CPU seconds

device xfer rate = 10 MBytes/sec => $.1 \times 10^{-6}$ sec/byte => .1 usec/byte
=> 1000 bytes = 100 usec
1000 transfers x 100 usecs = 100 ms = 0.1 CPU seconds

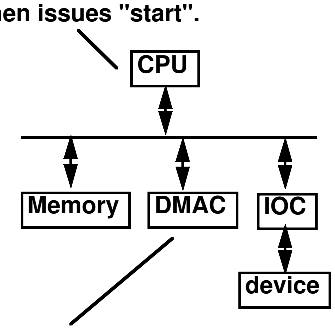
Still far from device transfer rate!1/2 in interrupt overhead

Direct Memory Access

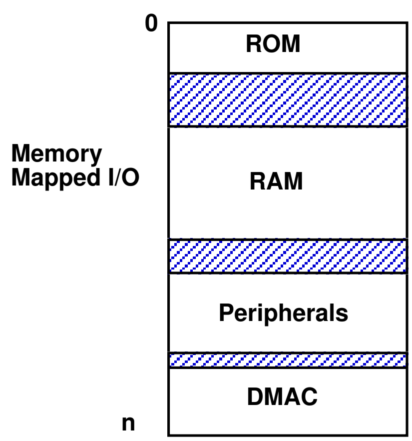
Time to do 1000 xfers at 1 msec each:
- 1 DMA set-up sequence @ 50 usec
- 1 interrupt @ 2 useq
- 1 interrupt service sequence @ 48 usec

.0001 second of CPU time

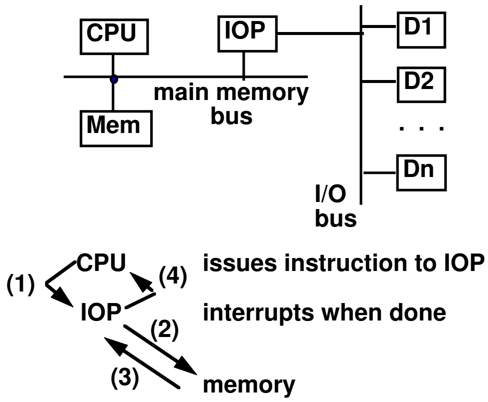
CPU sends a starting address, direction, and length count to DMAC.
Then issues "start".

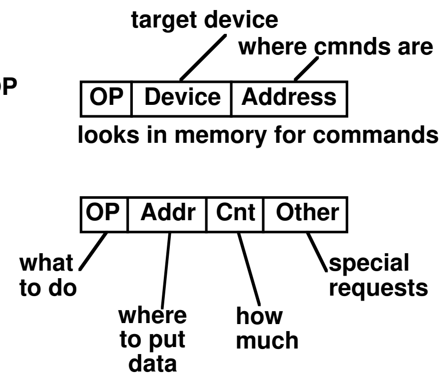


DMAC provides handshake signals for Peripheral Controller, and Memory Addresses and handshake signals for Memory.



Input/Output Processors





Device to/from memory transfers are controlled by the IOP directly.

IOP steals memory cycles.

Relationship to Processor Architecture

- I/O instructions and busses have disappeared
- Interrupt vectors have been replaced by jump tables
  PC <- M [IVA + interrupt number]
  PC <- IVA + interrupt number
- Interrupt stack replaced by shadow registers
  handler saves registers and re-enables higher priority int's
- Interrupt types reduced in number
  handler must query interrupt controller
- Caches required for processor performance cause problems for I/O
  - flushing is expensive, I/O polutes cache
  - solution is borrowed from shared memory multiprocessors "snooping"
- virtual memory frustrates DMA
- load/store architecture at odds with atomic operations
  - load locked, store conditional
- stateful processors hard to context switch