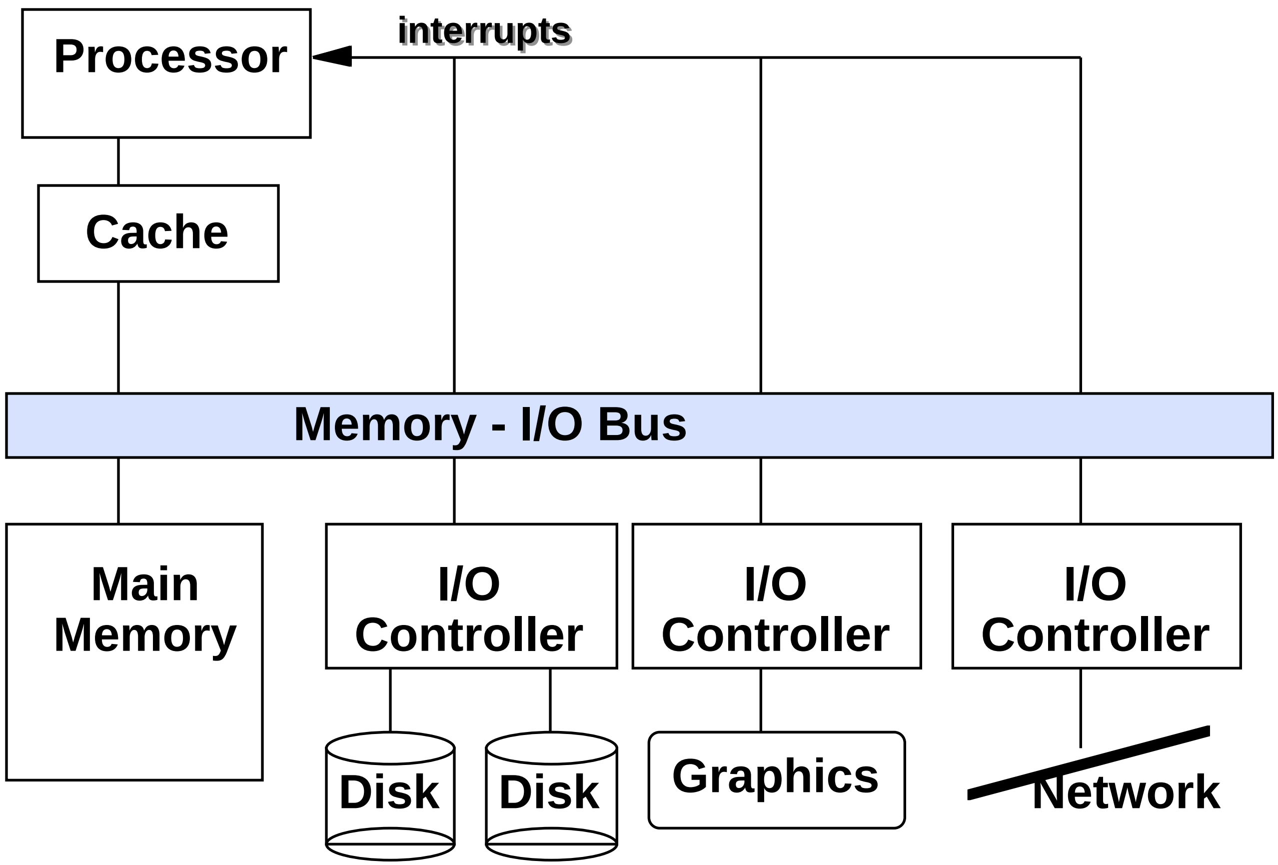
Lecture 21: I/O—A Little Queuing Theory and I/O Interfaces

Professor Randy H. Katz

Computer Science 252

Spring 1996

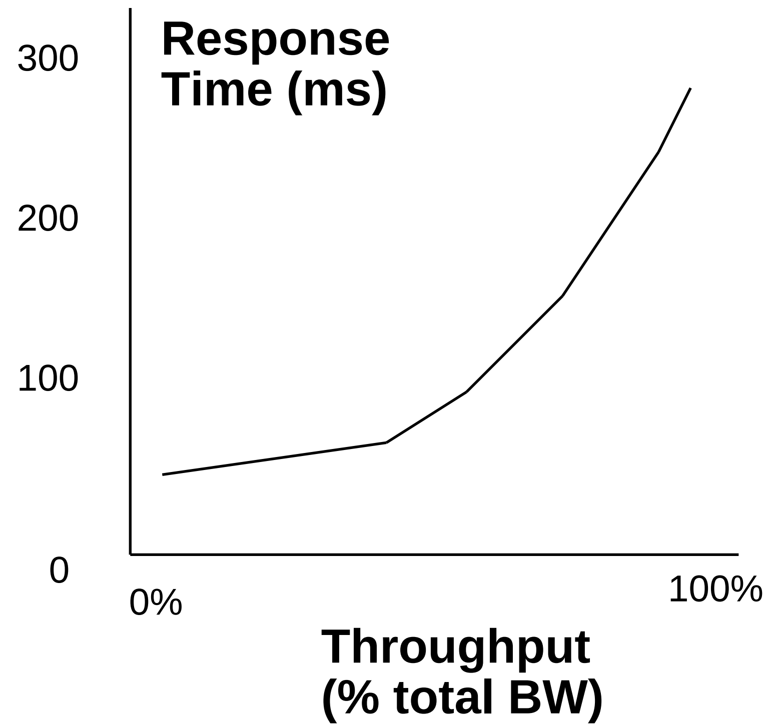
Review—I/O Systems

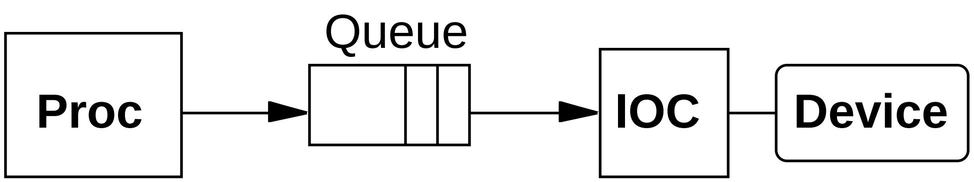


Time(workload) = Time(CPU) + Time(I/O) - Time(Overlap)

Review—Disk I/O Performance

Metrics: Response Time and Throughput





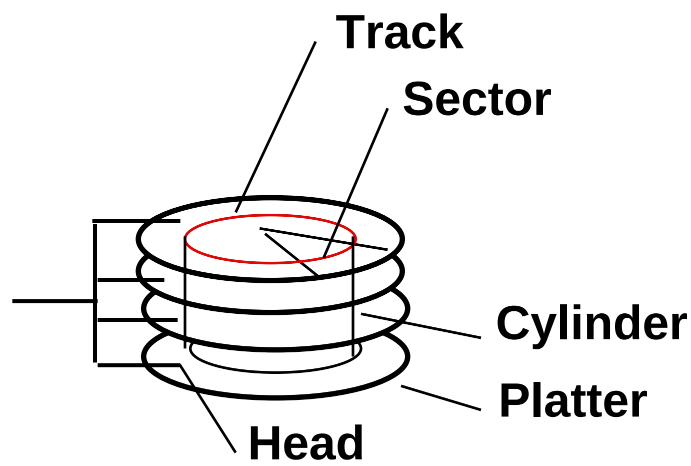
Response time = Queue + Device Service time

Storage System Issues

- Historical Context of Storage I/O
- Storage I/O Performance Measures
- Secondary and Tertiary 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
- Tertiary Storage Systems

Review: 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 (aerodynamics)



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

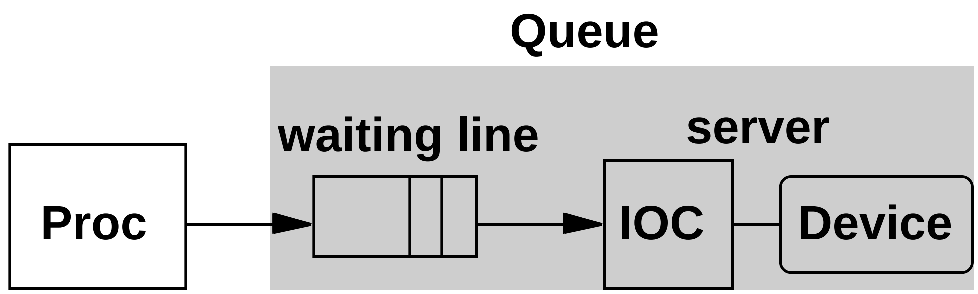
Response time

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

Disk Time Example

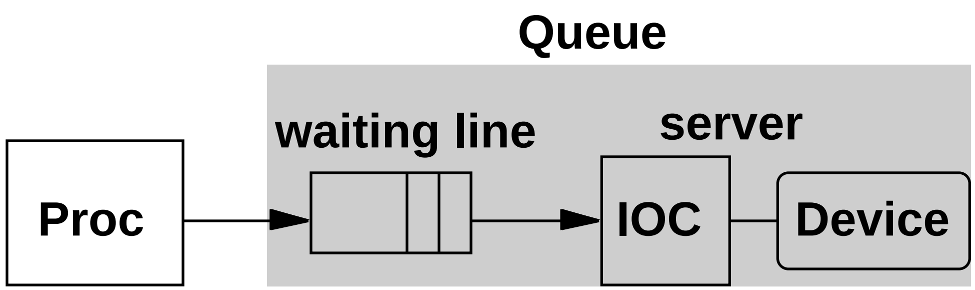
- **Disk Parameters:**
  - **Transfer size is 8K bytes**
  - **Advertised average seek is 12 ms**
  - **Disk spins at 7200 RPM**
  - **Transfer rate is 4 MB/sec**
- **Controller overhead is 2 ms**
- **Assume that disk is idle so no queuing delay**
- **What is Average Disk Access Time for a Sector?**
  - **Ave seek + ave rot delay + transfer time + controller overhead**
  - **12 ms + 0.5/(7200 RPM/60) + 8 KB/4 MB/s + 2 ms**
  - **12 + 4.15 + 2 + 2 = 20 ms**
- **Advertised 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 do you characterize variability?***

A Little Queuing Theory



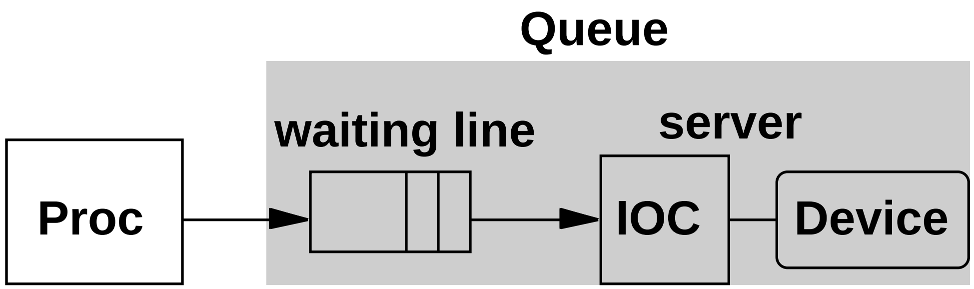
- **Server spends a variable amount of time with customers**
  - **Weighted mean *m1* = (f1 x T1 + f2 x T2 +...+ fn x Tn)/F (F=f1 + f2 +...)**
  - ***Squared coefficient of variance*: *C***

  C = variance/m1² $C = \text{variance}/m1^2$

  variance = **(f1 x T1² + f2 x T2² +...+ fn x Tn²)/F – m1²** $\text{variance} = (f1 \times T1^2 + f2 \times T2^2 + \ldots + fn \times Tn^2)/F - m1^2$

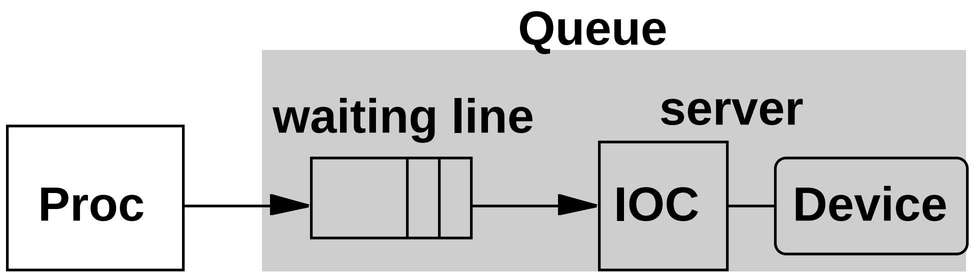
- **Exponential distribution *C = 1* : most short relative to average, few others long; 90% < 2.3 x average, 63% < average**
- **Hypoexponential distribution *C < 1* : most close to average, C=0.5 => 90% < 2.0 x average, only 57% < average**
- **Hyperexponential distribution *C > 1* : further from average 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 coefficient of variance *C*
- Disk response times *C ≈ 1.5* (majority seeks < average)
- 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 equilibrium: 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 service 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**

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

- **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:**
  - System in equilibrium
  - Time between two successive arrivals in line are random
  - Server can start on next customer immediately after prior finishes
  - No limit to the waiting line: works First-In-First-Out
  - Afterward, all customers in line must complete; each avg T_s
- **Described "memoryless" Markovian request arrival (M for C=1 exponentially 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

- **Note distinction between waiting time and queue delay**

A Little Queuing Theory: An Example

- **Suppose processor sends 10 x 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 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 x 8KB disk I/Os per sec, 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**

A Little Queuing Theory: Yet Another Example

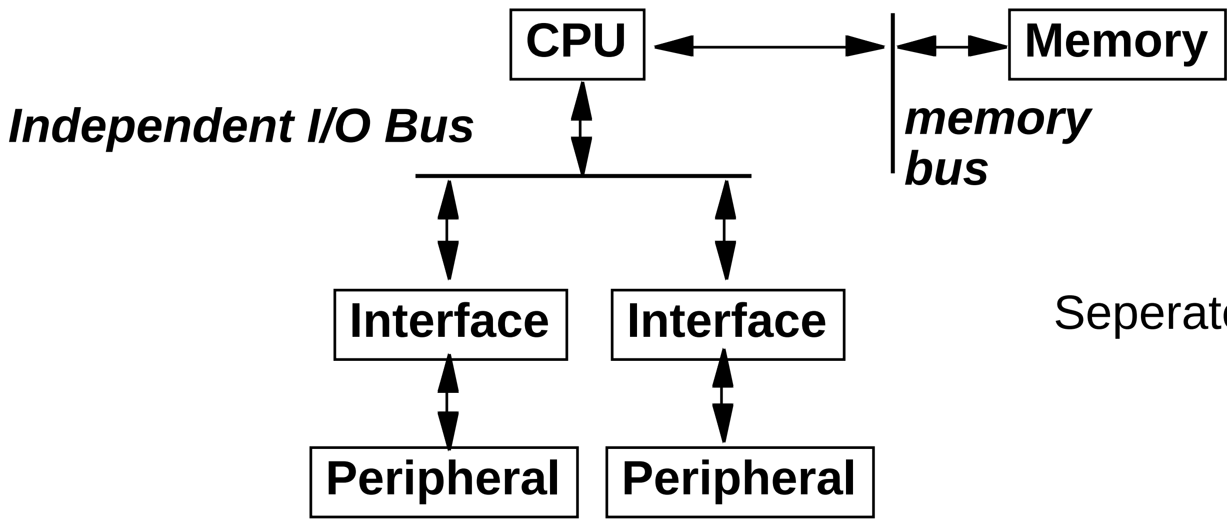
- **Suppose processor sends 10 x 8KB disk I/Os per second, req. squared coef. var. = 1.5, disk service time = 20 ms**
- **On average, how utilized is the disk?**
  - **What is the number of requests in the waiting line?**
  - **What is the average time a spent in the waiting line?**
  - **What is the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

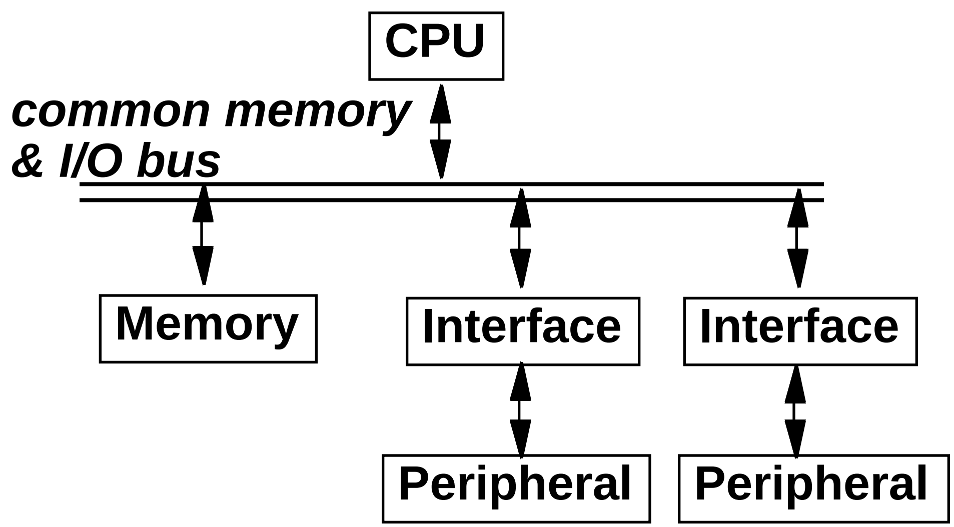
Processor Interface Issues

- **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



Seperate I/O instructions (in,out)



Lines distinguish between
I/O and memory transfers

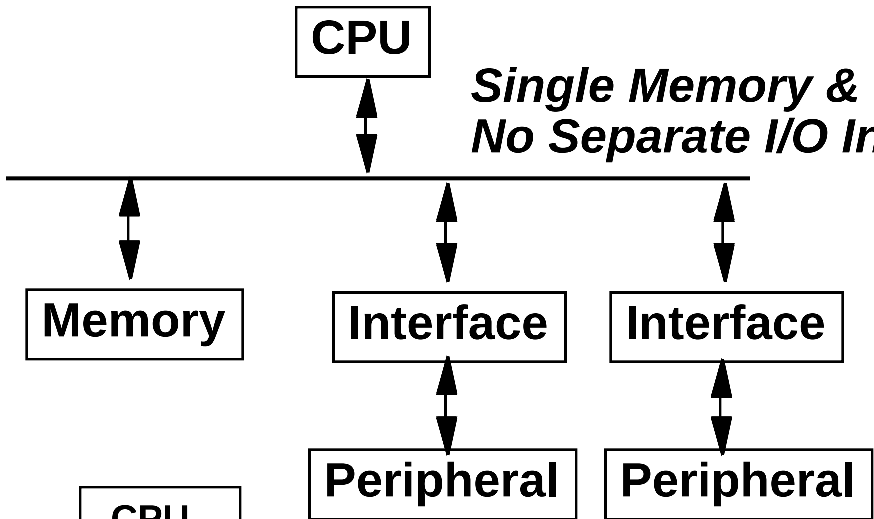
VME bus
Multibus-II
Nubus

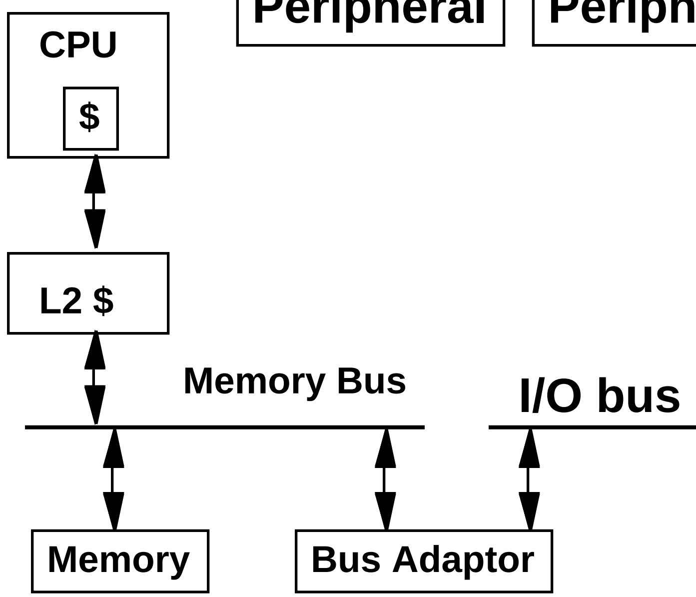
40 Mbytes/sec
optimistically

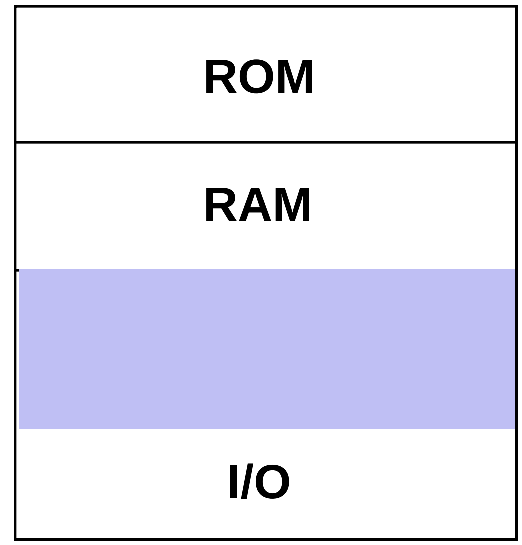
10 MIP processor
completely
saturates the bus!

Memory Mapped I/O

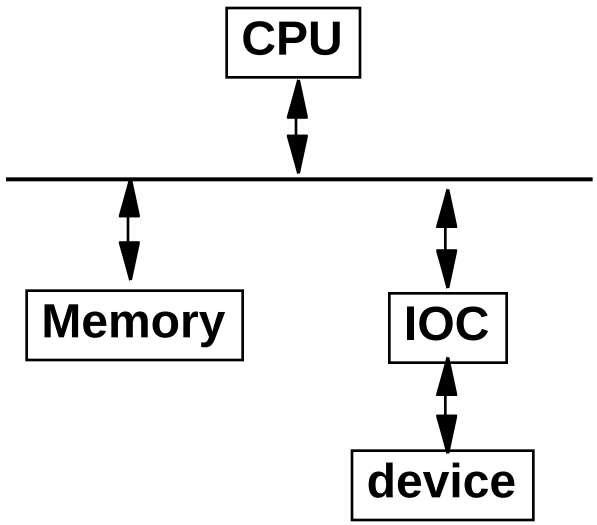
Single Memory & I/O Bus
No Separate I/O Instructions

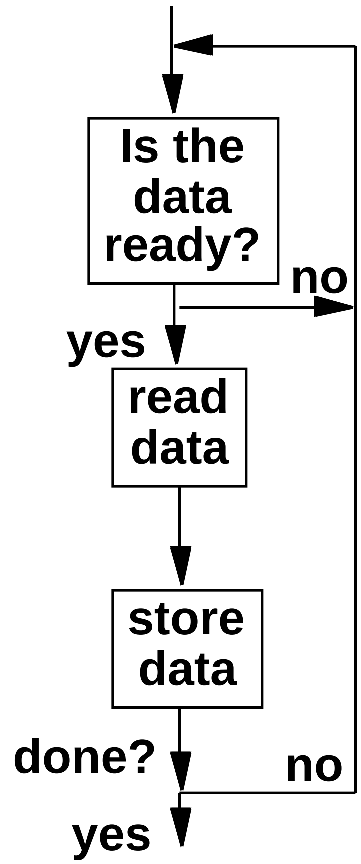






Programmed I/O (Polling)

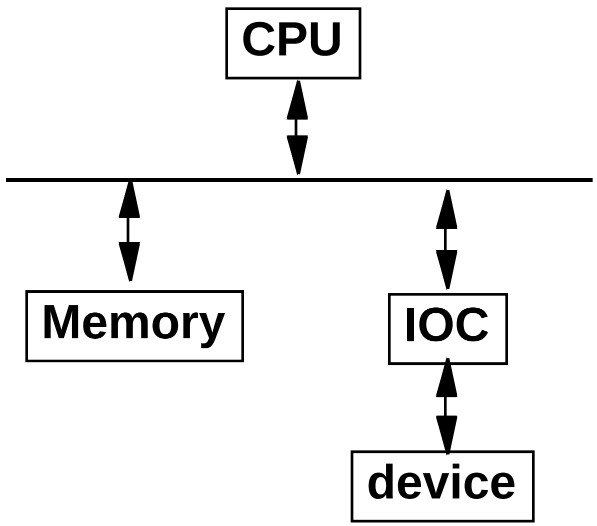


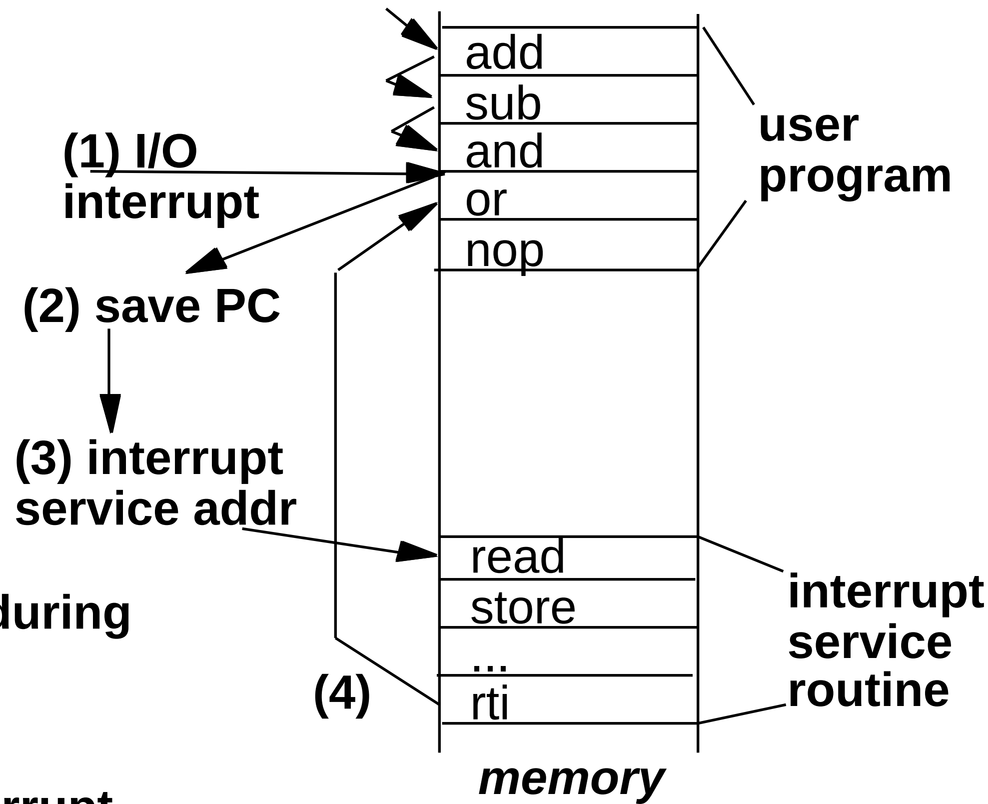


busy wait loop not an efficient way to use the CPU unless the device is very fast!

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

Interrupt Driven Data Transfer





User program progress only halted during
  actual transfer

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

Device xfer rate = 10 MBytes/sec => 0 .1 x 10^{-6} sec/byte => 0.1 µsec/byte
  => 1000 bytes = 100 µsec

1000 transfers x 100 µsecs = 100 ms = 0.1 CPU seconds

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

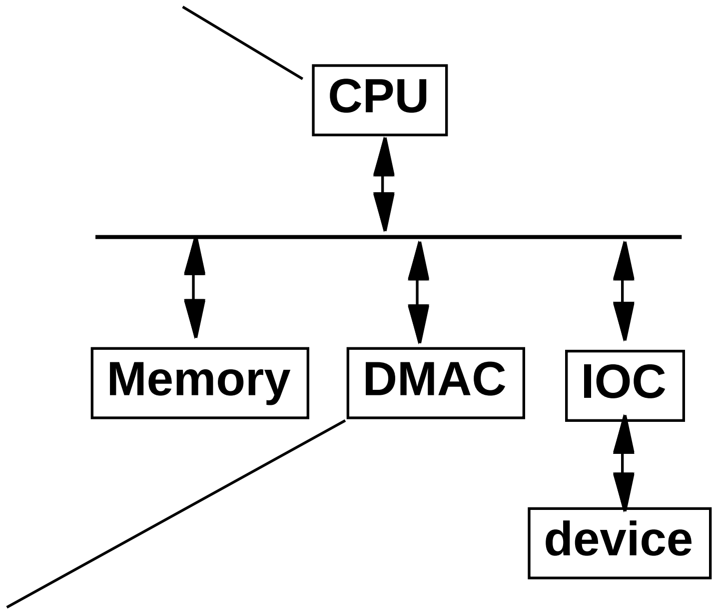
Direct Memory Access

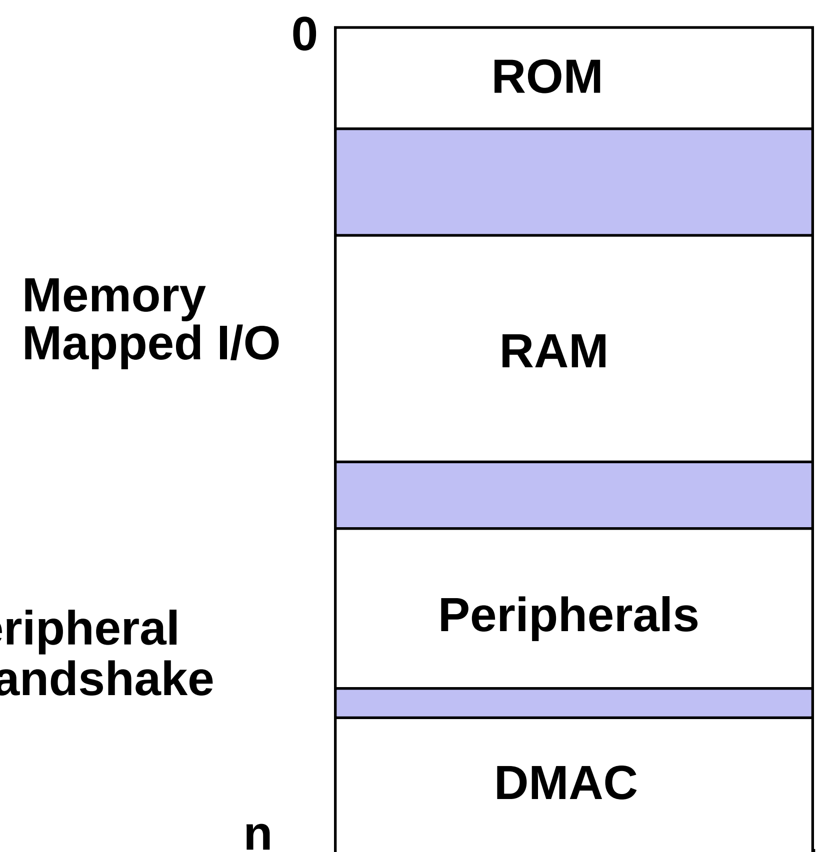
Time to do 1000 xfers at 1 msec each:

1 DMA set-up sequence @ 50 μsec
1 interrupt @ 2 μsec
1 interrupt service sequence @ 48 μsec

.0001 second of CPU time

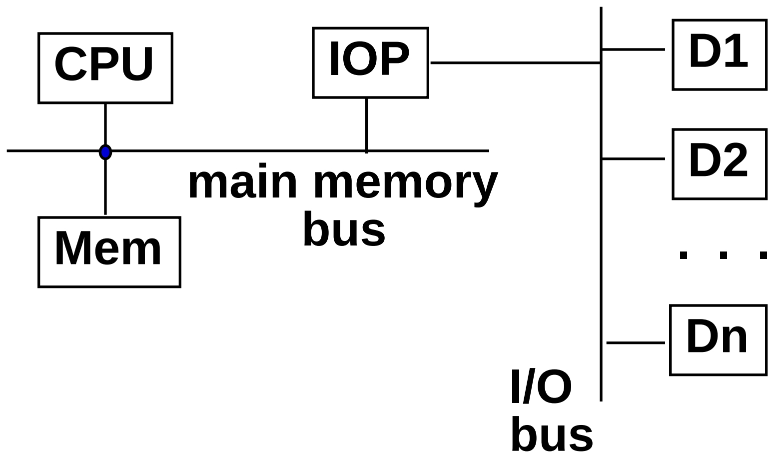
CPU sends a starting address, direction, and length count to DMAC. Then issues "start".

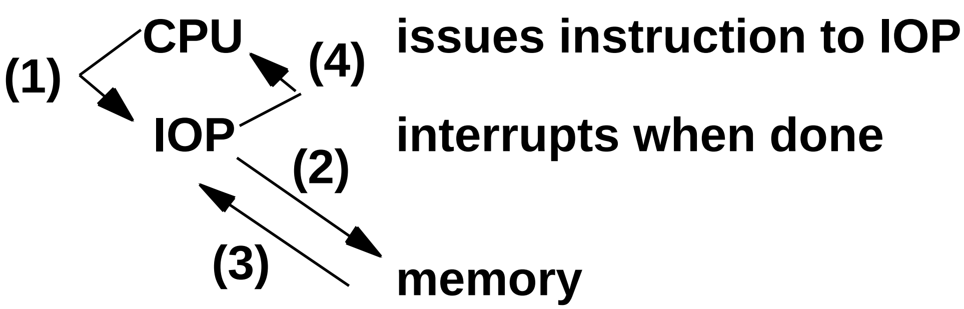


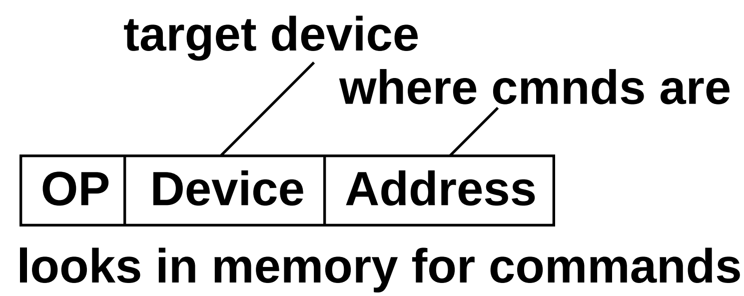


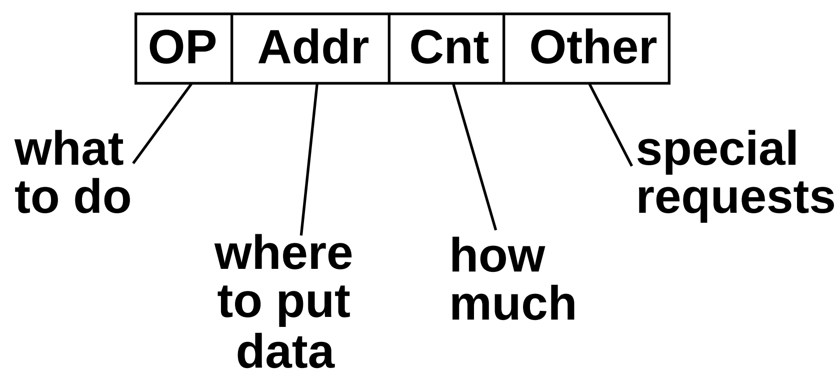
DMAC provides handshake signals for Peripheral Controller, and Memory Addresses and handshake signals for Memory.

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 largely disappeared**
- **Interrupt vectors have been replaced by jump tables**
  PC <- M [IVA + interrupt number]
  PC <- IVA + interrupt number
- **Interrupts:**
  - **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**

Relationship to Processor Architecture

- **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**