CMSC 313 Lecture 27

- System Performance
- CPU Performance
- Disk Performance

- Announcement:
  Don't use oscillator in DigSim3

Bottlenecks

The performance of a process might be limited by the performance of

- **CPU**
- **Memory system**
- **I/O**

Such a process is said to be *CPU bound, memory bound* or *I/O bound.*

Amdahl's "Law"

- Attributed to George Amdahl in 1967
- A formula that relates the speedup of the overall system resulting from the speedup of a component:

$$S = 1 / [(1 - f) + f / k]$$

S = overall speedup

f = fraction of work performed by faster component

k = speedup of new component

Amdahl's "Law" Example

- Suppose typical jobs on a system spend:

  70% of their time running in the CPU

  30% of their time waiting for the disk

- A processor that is 1.5x faster would result in:

  S = 1 / [(1 - 0.7) + 0.7 / 1.5] = 1.3043...

- A disk drive that is 2.5x faster would result in:

  S = 1 / [(1 - 0.3) + 0.3 / 2.5] = 1.2195...

System Performance

- The overall performance of a computer system depends on the performance of a mixture of processes.
- Some processes might be CPU bound, others memory or I/O bound.
- Cannot represent performance as a single number.
  - Megahertz Myth, MIPS, MFLOPS
- Benchmarks can be and have been manipulated.
  - Whetstone, Linpack, Dhrystone, SPEC
- OTOH, speeding up the CPU & I/O does help.

CPU Performance

CPU time/program =

time/cycle

x cycles/instruction

x instructions/program

- Higher clock rate reduces time/cycle
- RISC machines reduce cycles/instruction
- CISC machines reduce instructions/program

Other CPU Optimizations

- Integrated Floating Point Unit (FPU)
- Parallel execution units
- Instruction pipelining
- Branch prediction
- Speculative execution
- Code optimization

The Case for RISC Architecture

- **Reduced Instruction Set Computer features**

  minimal instructions, load/store architecture, simple addressing modes, instructions with fixed length, lots of registers

- **Instructions can be prefetched**
- **Simple instructions executed quickly**
- **Memory is cheap**
- **Pipelining is easier**

Instruction Frequency

- **Frequency of occurrence of instruction types for a variety of languages. The percentages do not sum to 100 due to roundoff. (Adapted from Knuth, D. E., *An Empirical Study of FORTRAN Programs, Software—Practice and Experience*, 1, 105-133, 1971.)**

Statement	Average Percent of Time
Assignment	47
If	23
Call	15
Loop	6
Goto	3
Other	7

Complexity of Assignments

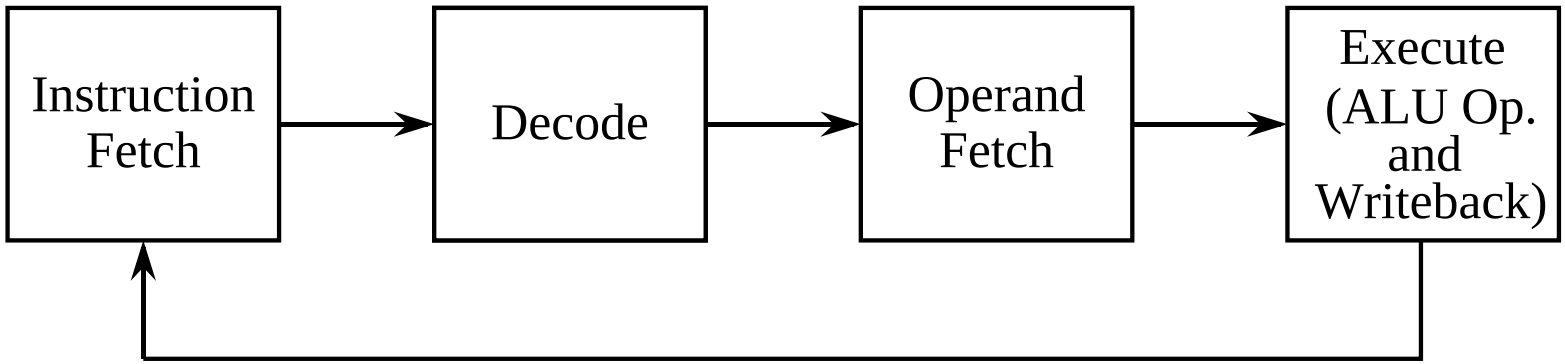
- **Percentages showing complexity of assignments and procedure calls. (Adapted from Tanenbaum, A., *Structured Computer Organization*, 4/e, Prentice Hall, Upper Saddle River, New Jersey, 1999.)**

	Percentage of Number of Terms in Assignments	Percentage of Number of Locals in Procedures	Percentage of Number of Parameters in Procedure Calls
0	–	22	41
1	80	17	19
2	15	20	15
3	3	14	9
4	2	8	7
≥ 5	0	20	8

Pipelining

- Fetch-execute cycle broken down into stages
- Each stage performed by different units
- Next instruction begins before current one finishes
- Ideally, 1 instruction finished per cycle
- Branch instructions mess up the pipeline
- Keeping pipeline filled may be difficult

Four-Stage Instruction Pipeline



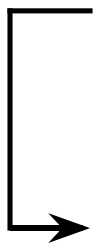
Pipeline Behavior

- **Pipeline behavior during a memory reference and during a branch.**

	Time 1	2	3	4	5	6	7	8
Instruction Fetch	addcc	ld	srl	subcc	be	nop	nop	nop
Decode		addcc	ld	srl	subcc	be	nop	nop
Operand Fetch			addcc	ld	srl	subcc	be	nop
Execute				addcc	ld	srl	subcc	be
Memory Reference						ld		

Filling the Load Delay Slot

- **SPARC code, (a) with a nop inserted, and (b) with `srl` migrated to nop position.**



```
srl   %r3, %r5
addcc %r1, 10, %r1
ld    %r1, %r2
nop
subcc %r2, %r4, %r4
be    label
```

(a)

```
addcc %r1, 10, %r1
ld    %r1, %r2
srl   %r3, %r5
subcc %r2, %r4, %r4
be    label
```

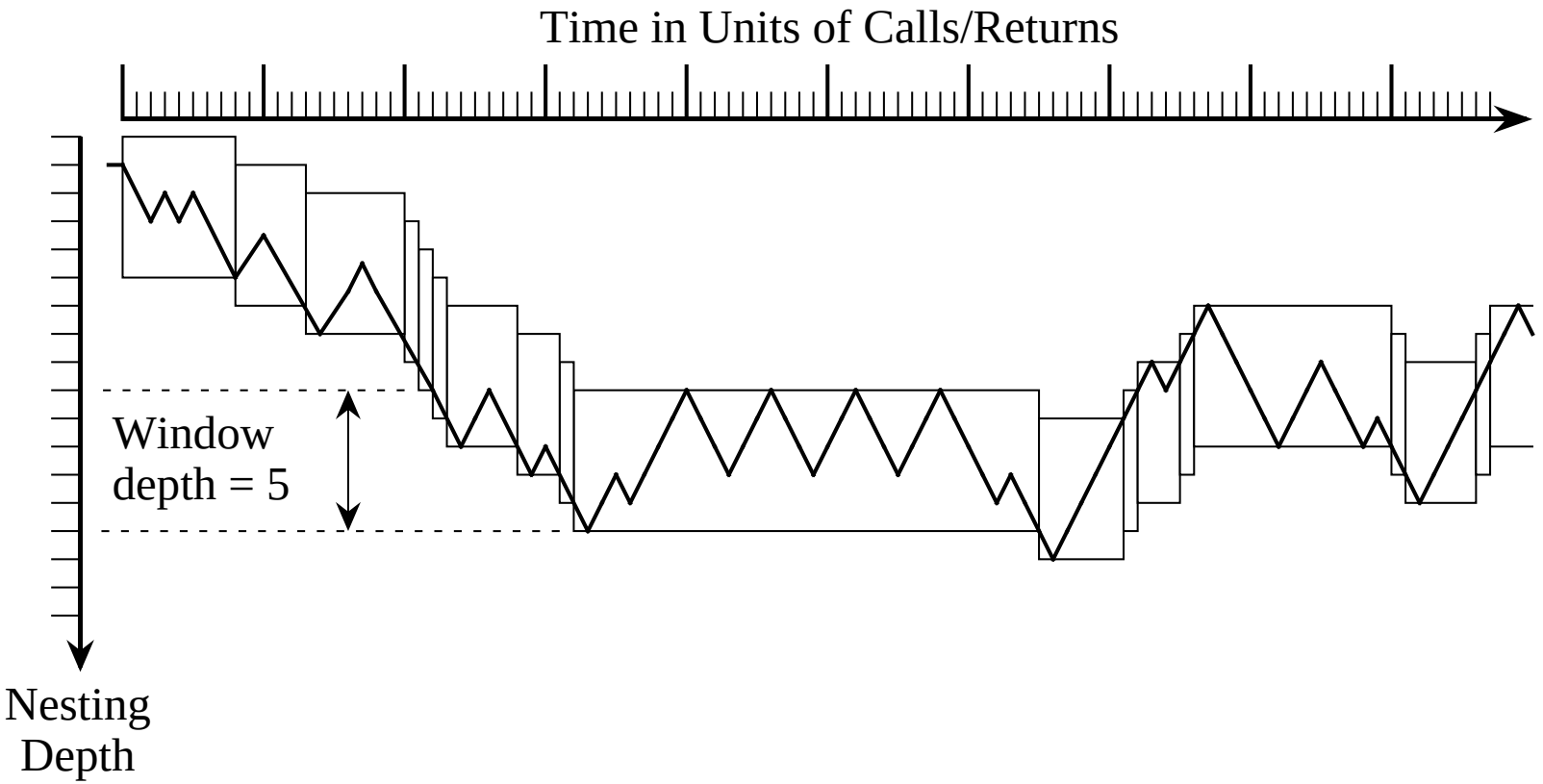
(b)

Register Windows

- Used in SPARC machines, not popular
- Use registers for parameter passing
- CPU has more registers than program "sees"
- Parameters passed by register
- Function call depth does not vary that much

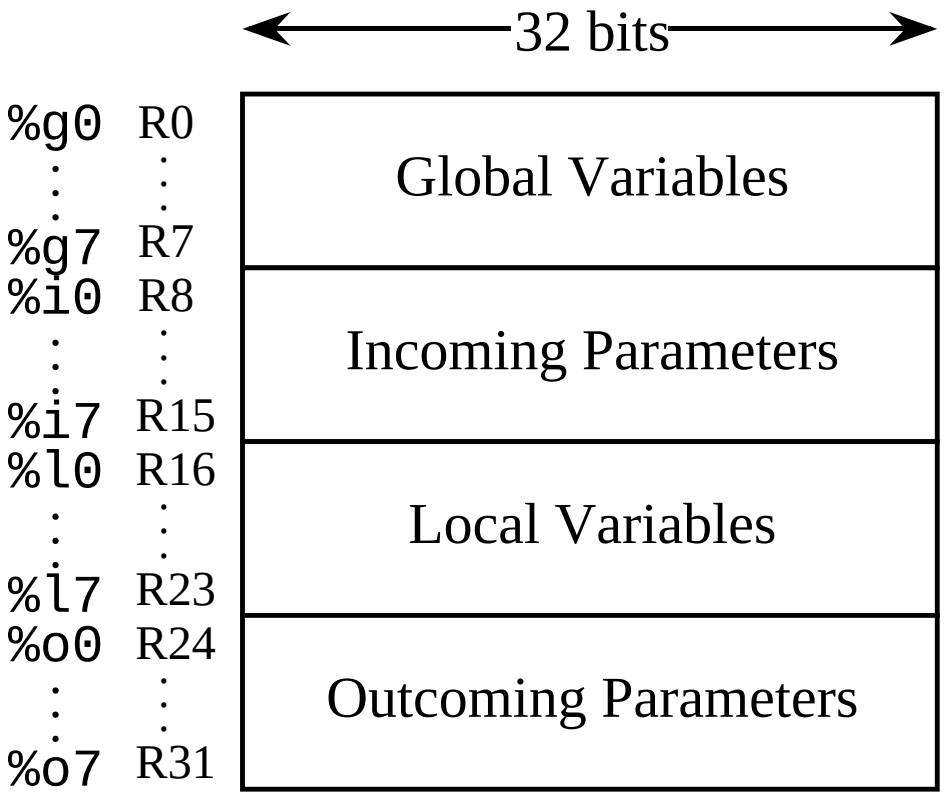
Call-Return Behavior

- **Call-return behavior as a function of nesting depth and time (Adapted from Stallings, W., *Computer Organization and Architecture: Designing for Performance*, 4/e, Prentice Hall, Upper Saddle River, 1996).**

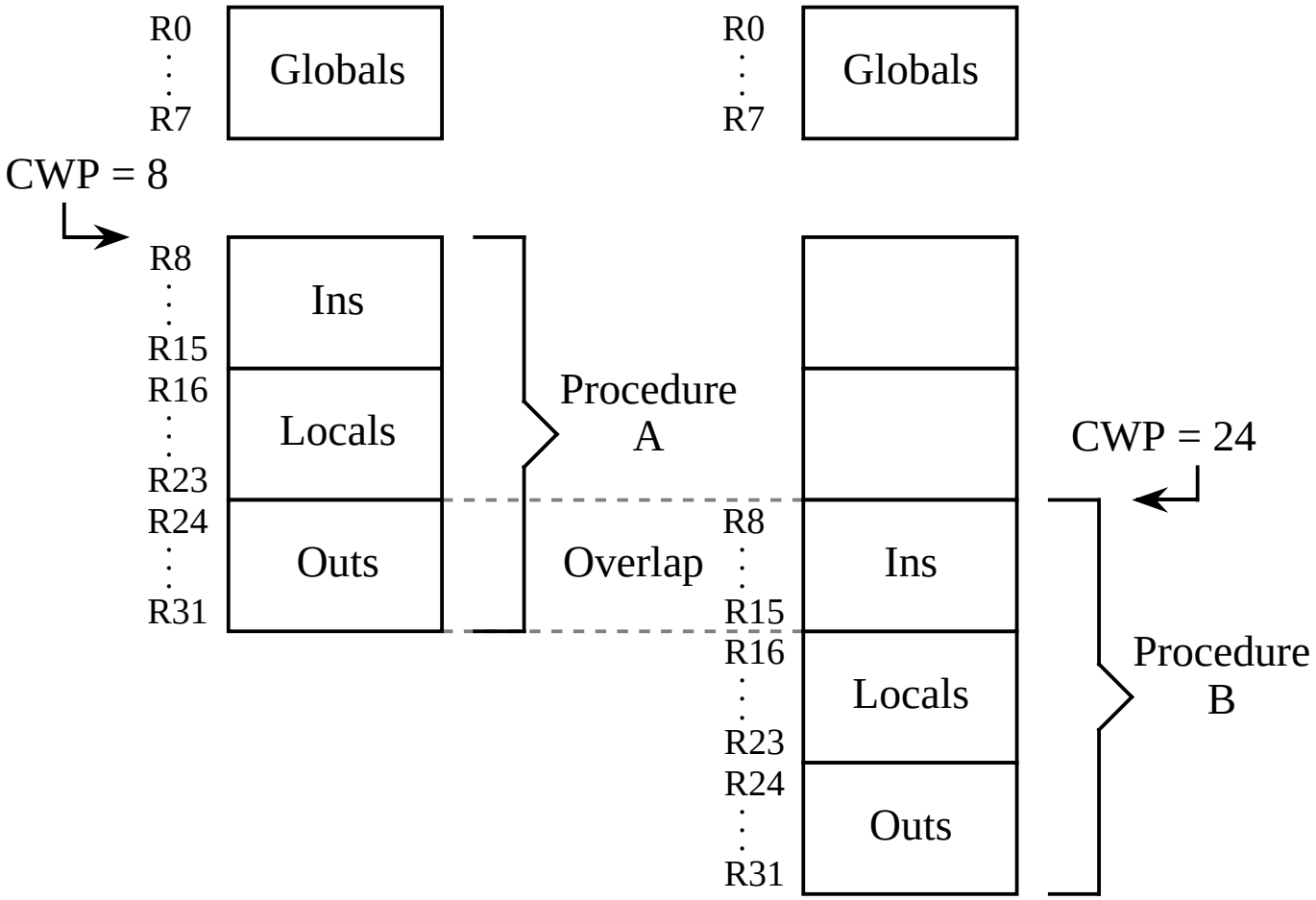


SPARC Registers

- **User view of RISC I registers.**



Overlapping Register Windows

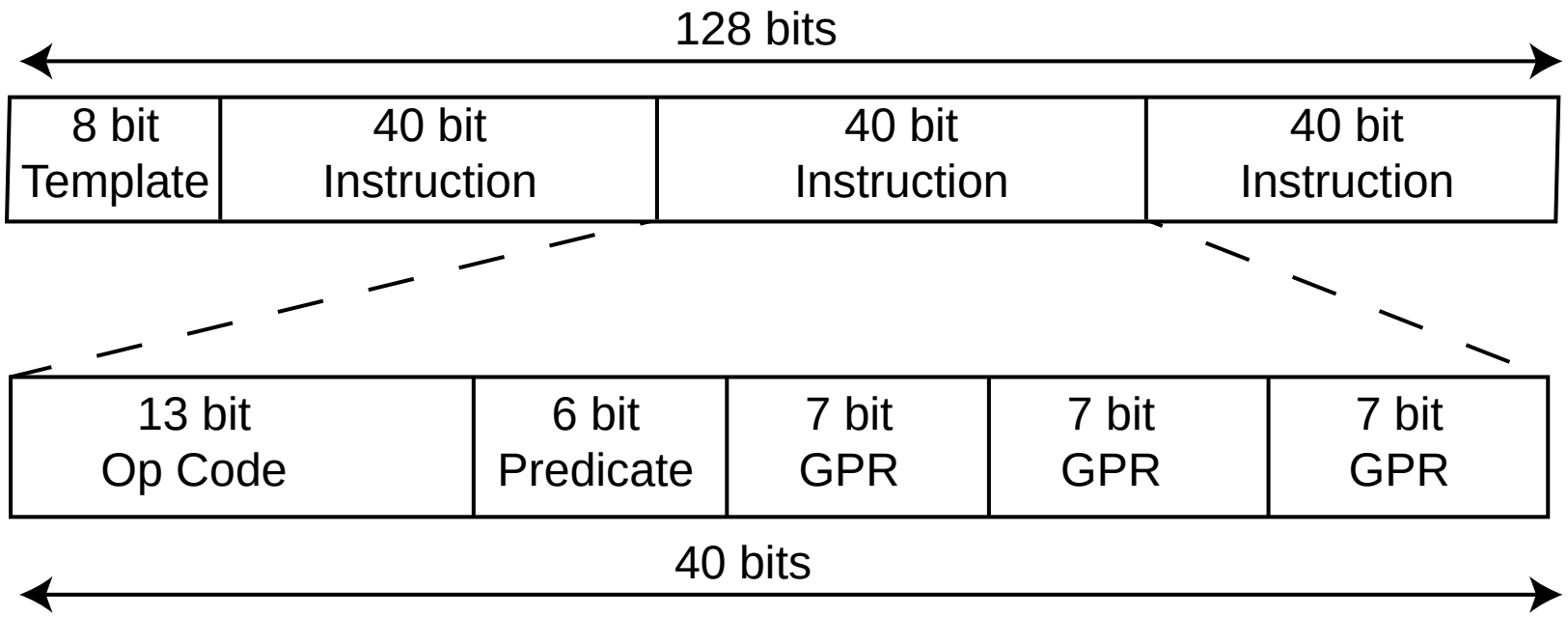


Superscalar Architecture

- Parallel execution units (e.g., duplicate ALUs)
- Instruction-Level Parallelism

  (versus Program-Level Parallelism)
- Data dependencies
- Out-of-order execution
- Dynamic scheduling
- Static scheduling
  - ◇ VLIW = Very Long Instruction Word
  - ◇ EPIC = Explicitly Parallel Instruction Computer
    = VLIW with variable length instructions

128-Bit IA-64 Instruction Word

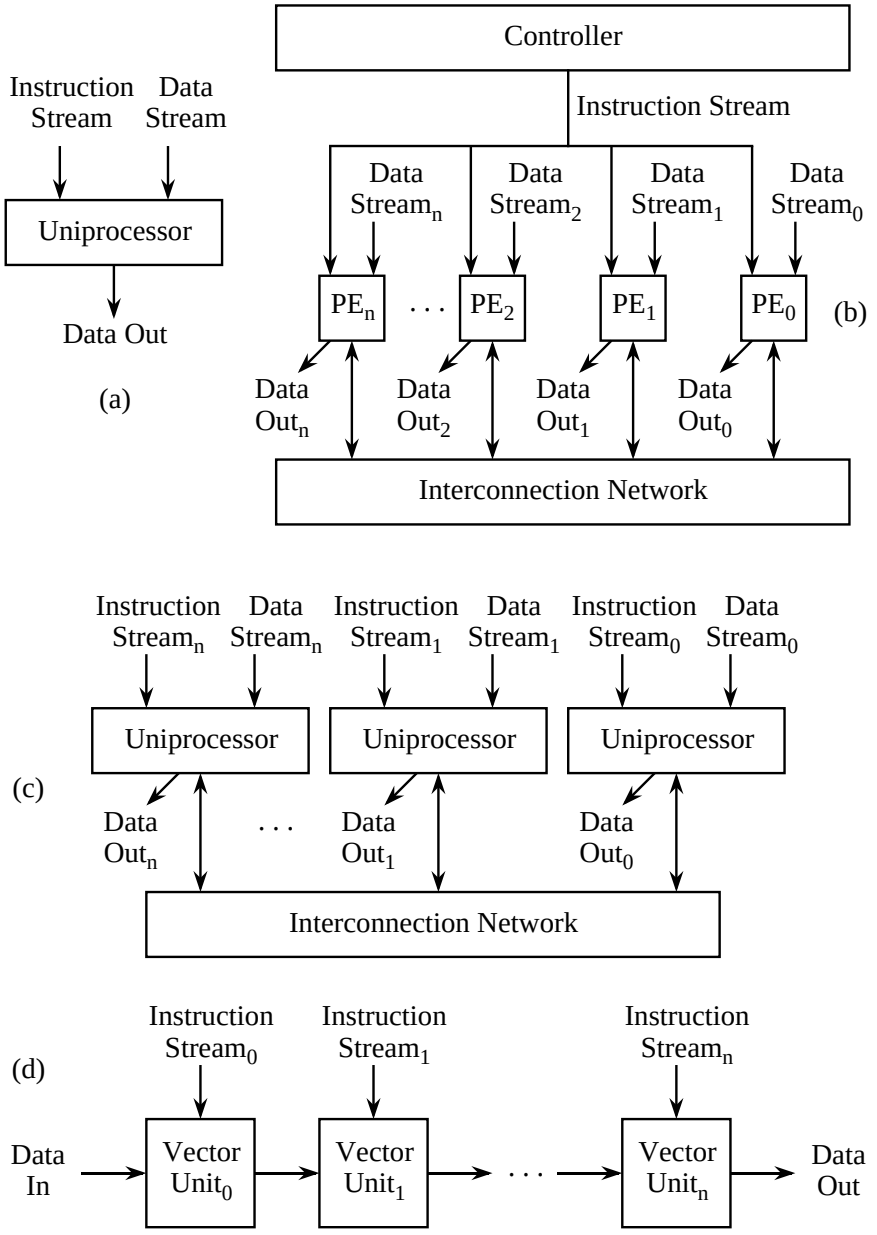


Parallel Processing

- Many processors connected
- Program-Level Parallelism
- Shared memory versus message passing
- Process must be "parallelizable"
- Hard to keep processors busy
- How do we program these things???

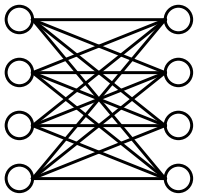
Flynn Taxonomy

- **Classification of architectures according to the Flynn taxonomy: (a) SISD; (b) SIMD; (c) MIMD; (d) MISD.**

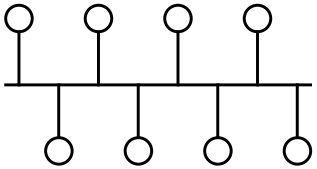


Network Topologies

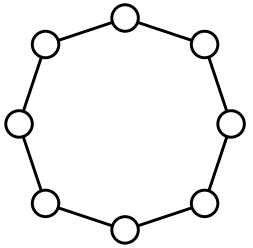
- **Network topologies: (a) crossbar; (b) bus; (c) ring; (d) mesh; (e) star; (f) tree; (g) perfect shuffle; (h) hypercube.**



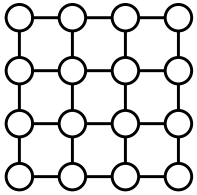
(a)



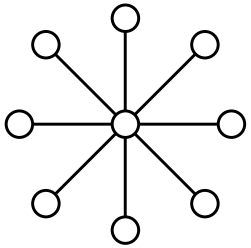
(b)



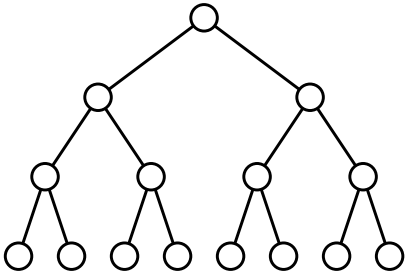
(c)



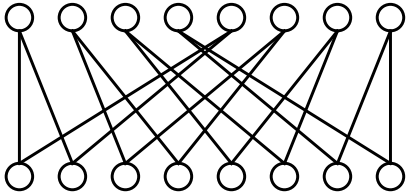
(d)



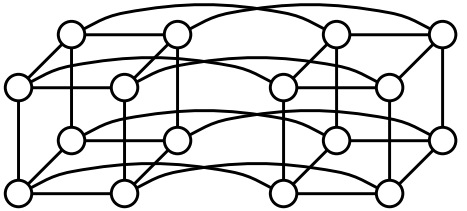
(e)



(f)



(g)

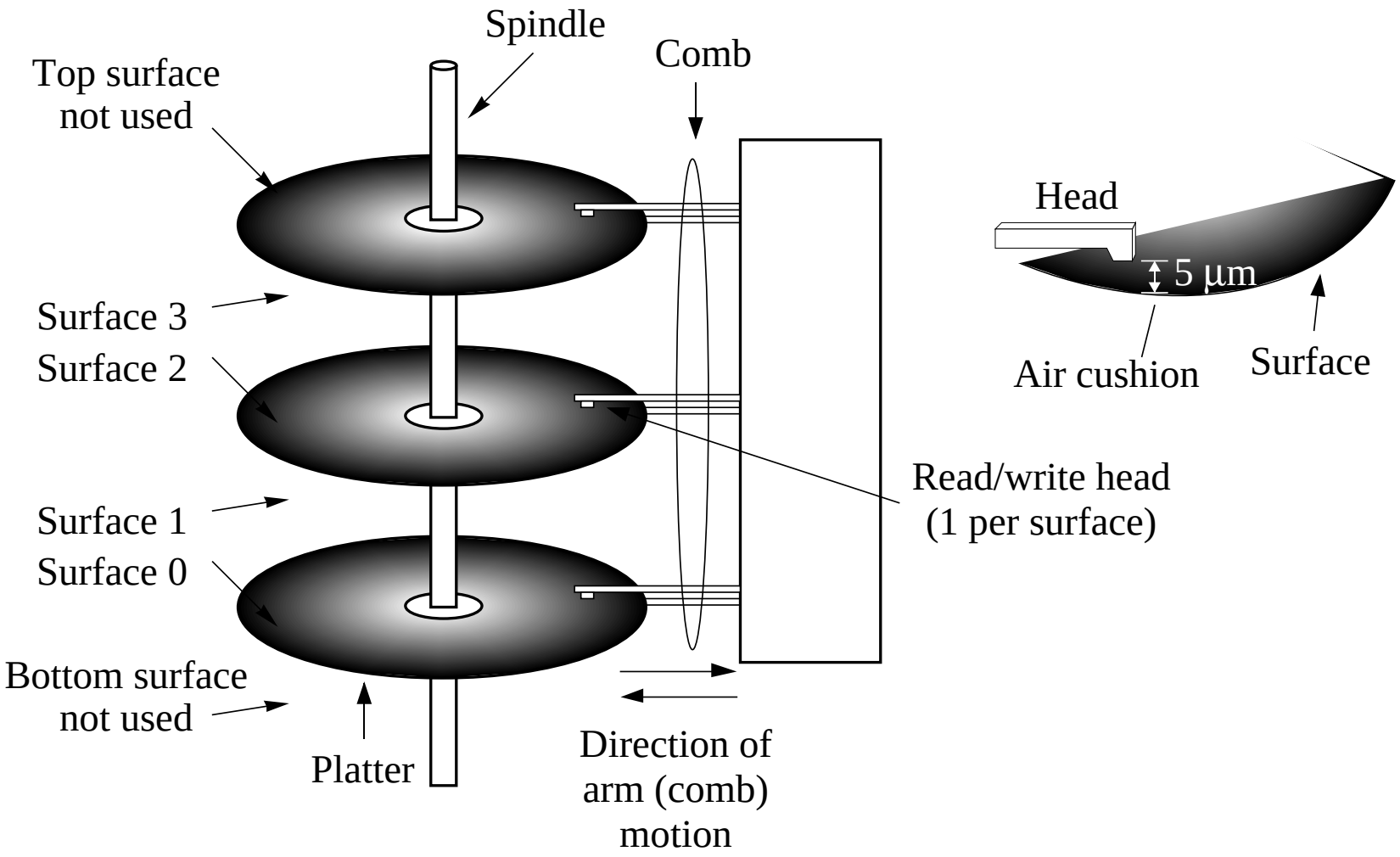


(h)

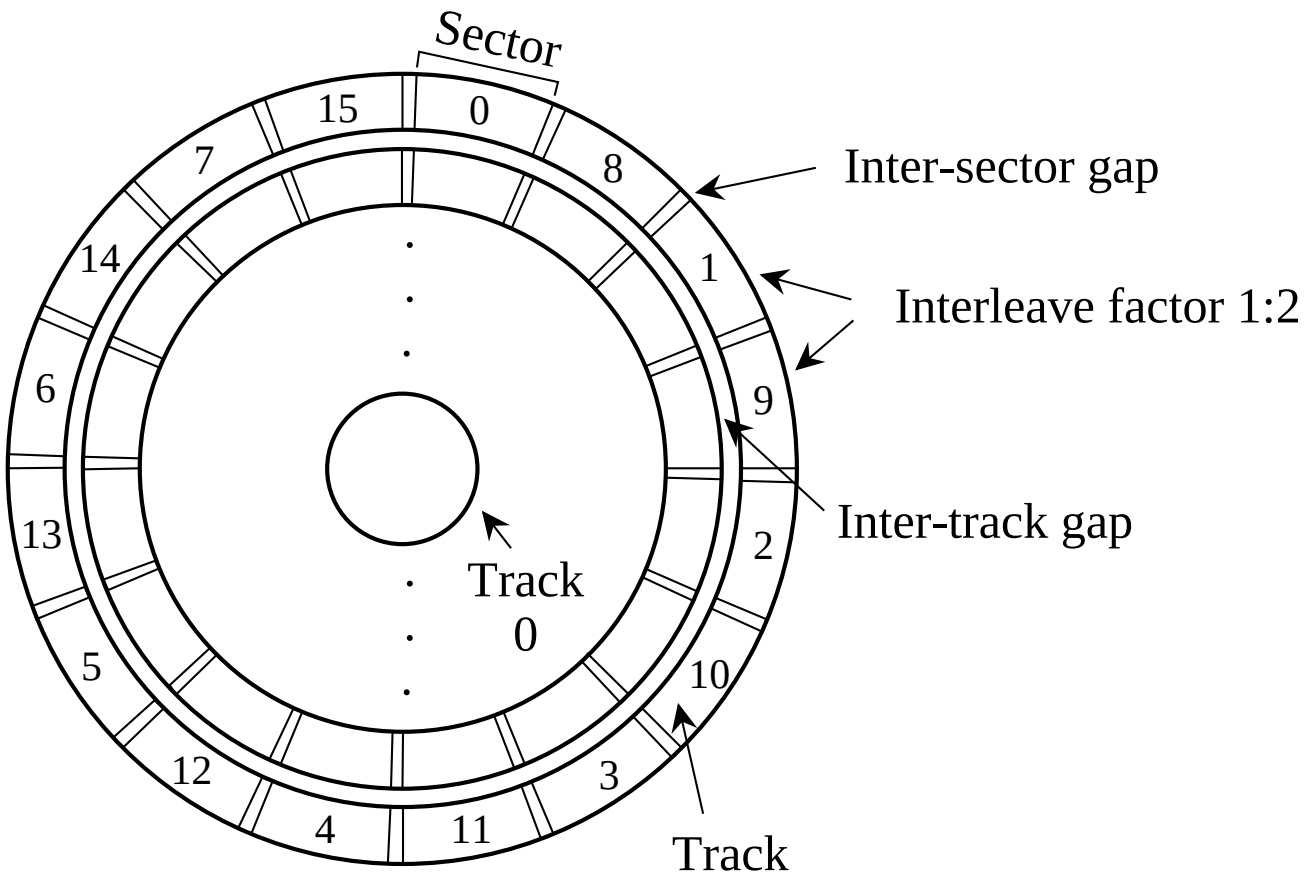
Disk Performance

- Seek time = time to move disk arm to track
- Rotational delay = time for sector to be positioned
- Access time = seek time + rotational delay
- Transfer time = time to read/write a sector
- Capacity of hard disks increased dramatically
- Average access time (~8ms) hasn't improved much
- When several processes use the same disk, scheduling becomes important.

A Magnetic Disk with Three Platters



Organization of a Disk Platter with a 1:2 Interleave Factor



Seek Time

- The seek time is the time necessary to move the arm from its previous location to the correct track.
- Industry standard is *average seek time* computed by averaging all possible seeks.
- Average seek time tends to overstate actual seek times because of locality of disk references.

Rotational Delay

- Time for requested sector to rotate under head
- Typical rotation speeds 3,600 - 7,200 rpm
  - ◇ disks rotated at 3,600rpm in 1975.
- On average disks rotate half way to desired sector
  - ◇ 6,000rpm => 0.5/ (6,000rpm / 60,000 ms/min) = 5ms
- Some disks can read data out of order into a buffer so a full track can be transferred in one rotation

Transfer Time

- Time to read or write a sector
- Approximately equal to time for a sector to pass fully under the head
- Function of block size, rotation speed, recording density and speed of electronics
  - 6,000rpm, 64 sectors per track, 512 bytes per sector
  - 6,000rpm = 100 revs per second => 10ms per rev
  - 64 sectors => 10ms/64 = 0.156ms for sector to pass under head
  - 512 bytes => 1KB takes 0.312ms => 3.2 KB/ms = 3.2 MB/s

Next Time

- Semester overview
- Sample Final Exam