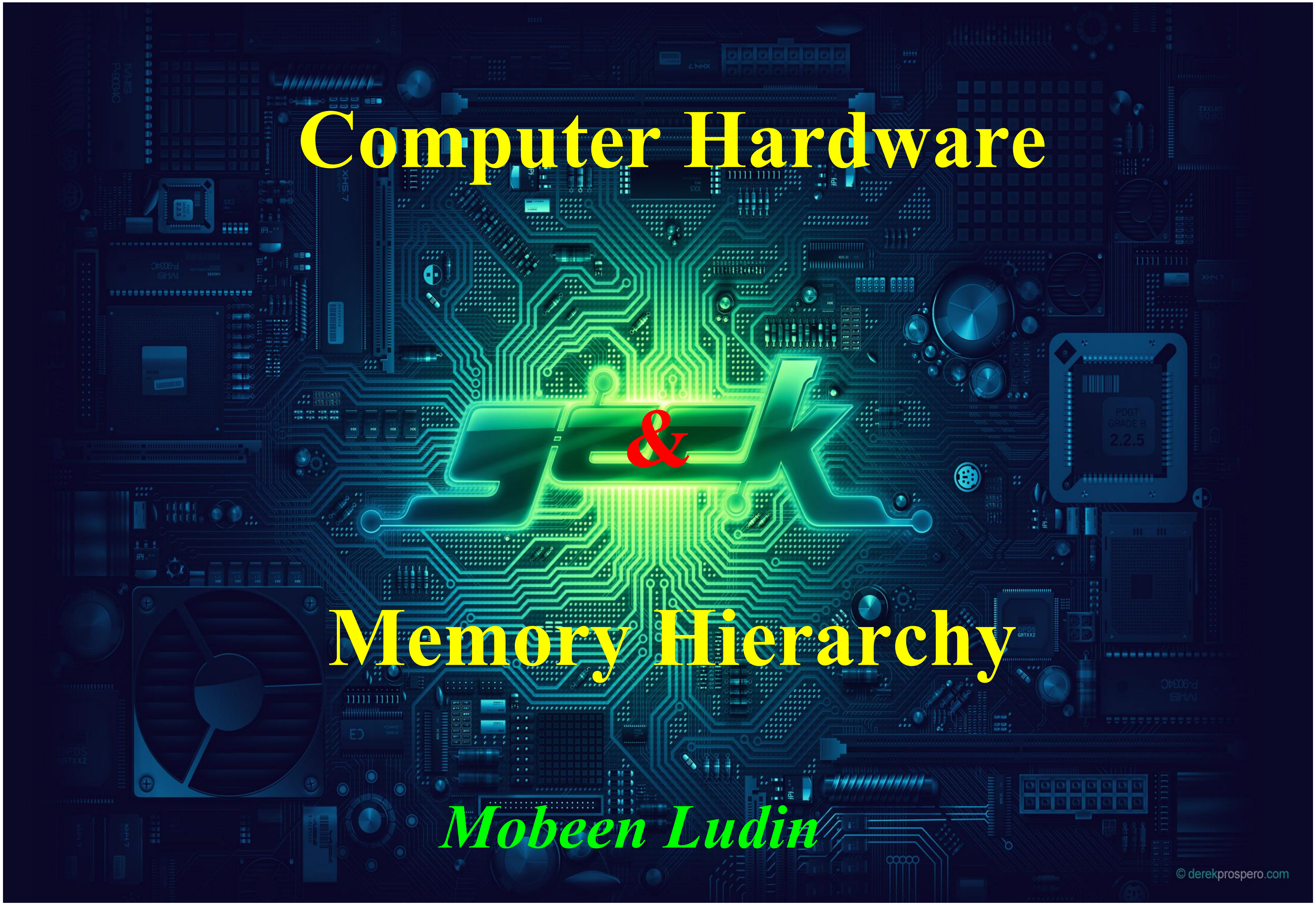


Computer Hardware & Memory Hierarchy

Mobeen Ludin

ssdf

High-level overview of this lecture!!

- We will have deepdive into the cost in hardware, cost of running program on a single processor
  - If you don’t understand the cost in a single processor you are likely not going to understand the cost in parallel processor
- Most application runs at **< 10 % of the “peak”** performance of system
  - **Peak** = Whatever the manufacturer advertises (GHz)
- Most of the 90% waste of performance is time spent moving the data from memories to CPU, so the processor could do some useful things on it.
- **Operations include:**
  - write/read into the very fast memory (registers)
  - arithmetic and logical operations on registers
- **Order specified by program**
  - User level:

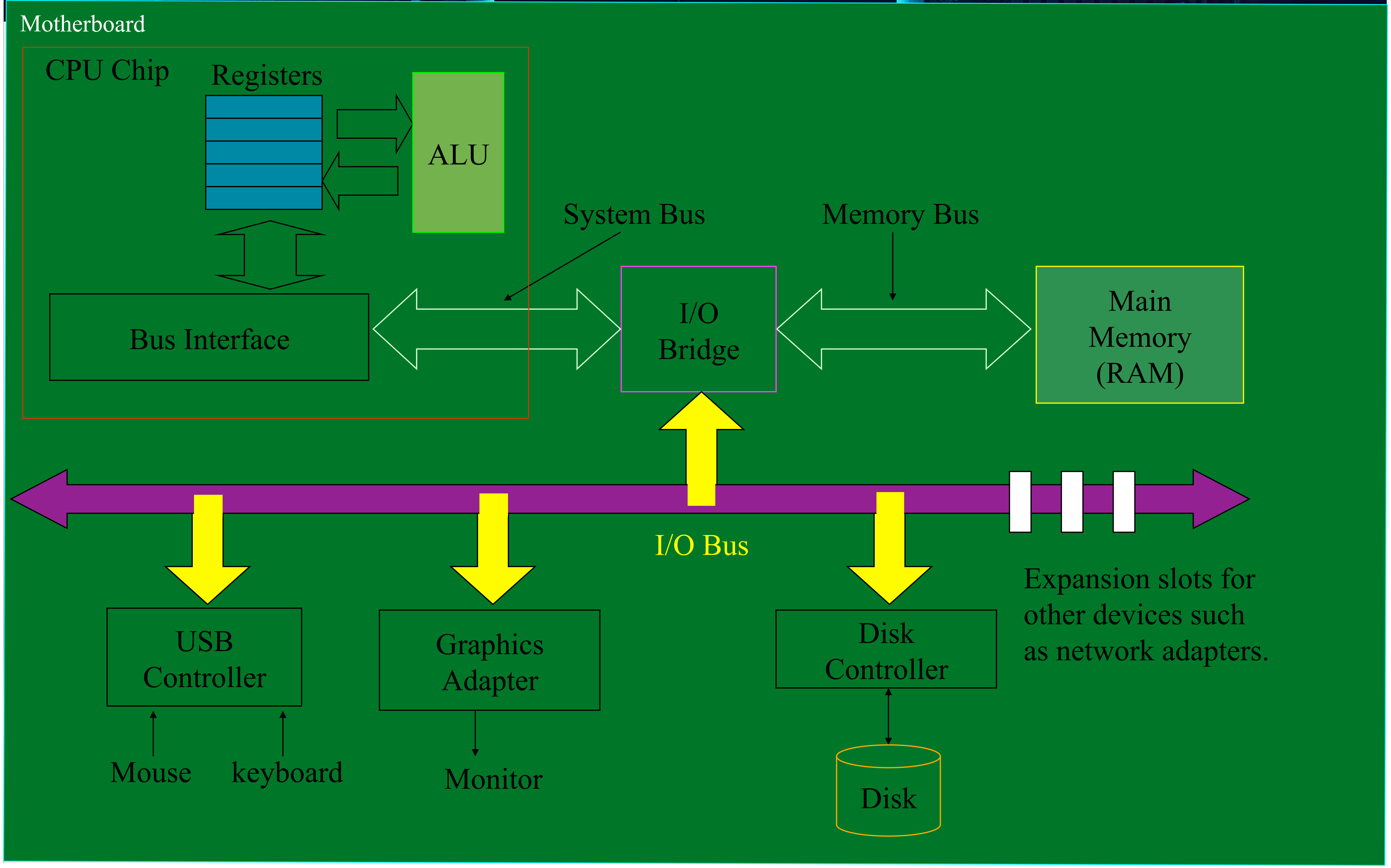
```
A = B + C
```

  - Compiler/CPU level:

```
Read Address(B) to R1
Read address(C) to R2
R3 = R1 + R2
Write R3 to Address(A)
```

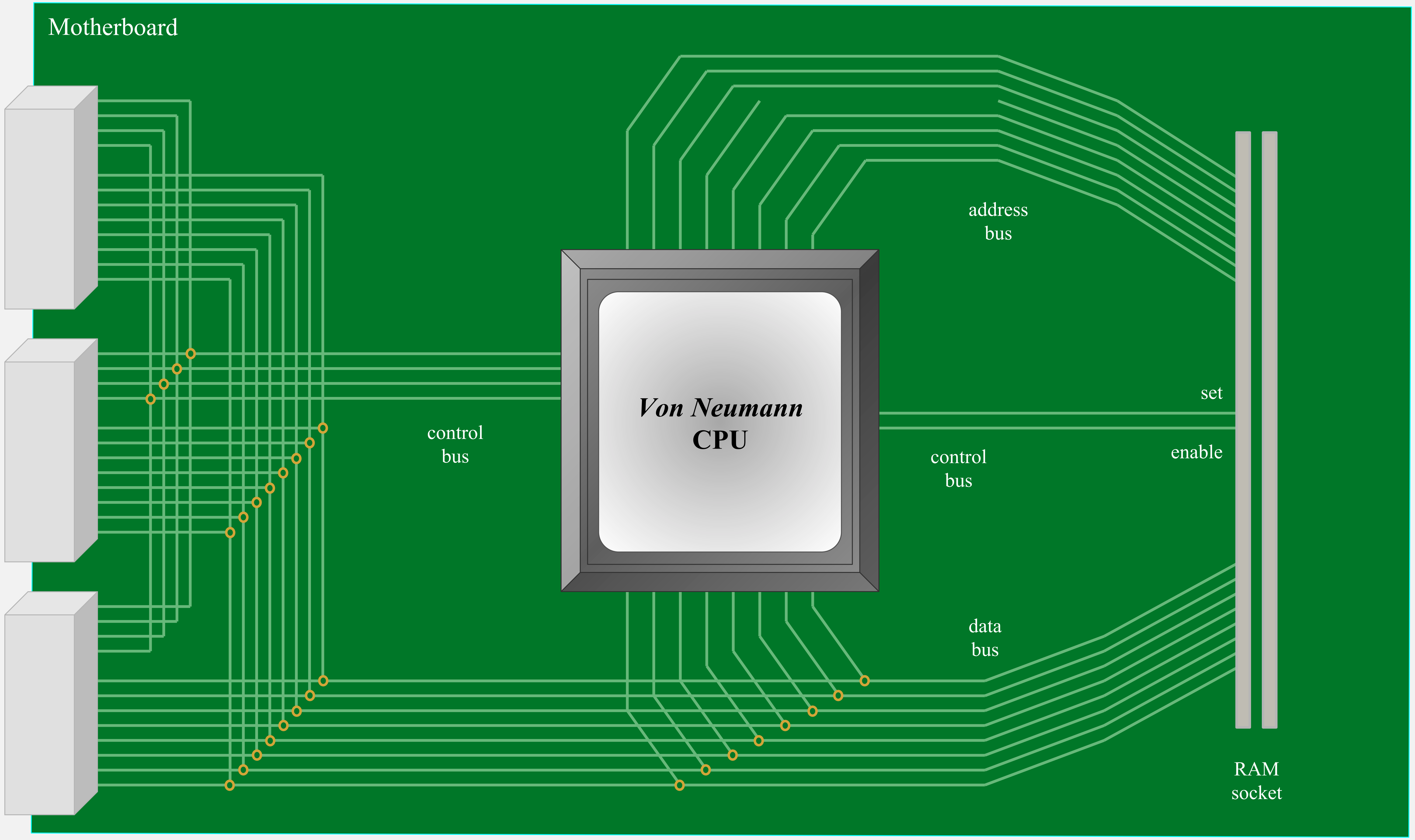
ssdf

Typical Bus Structure Computer



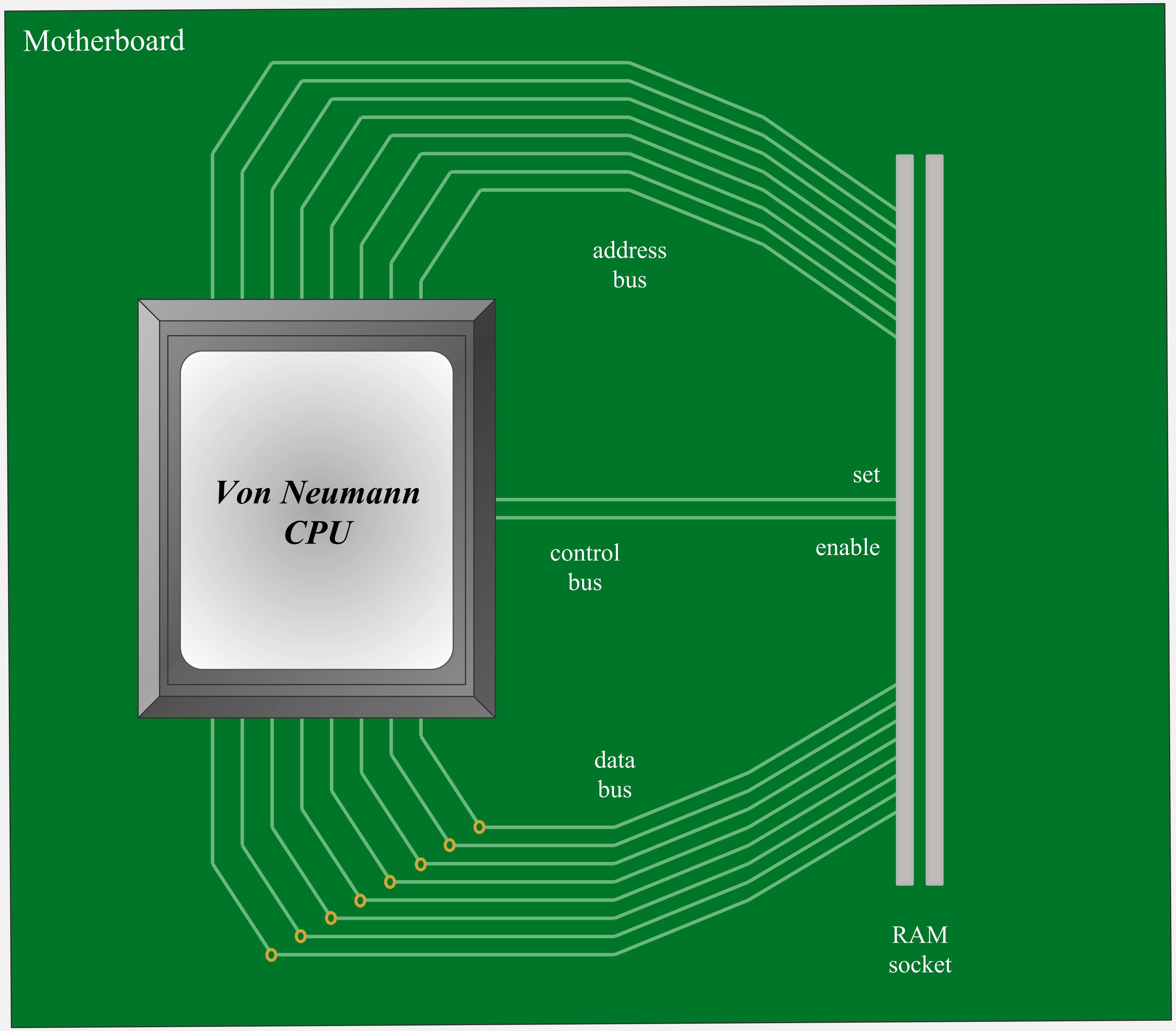
ssdf

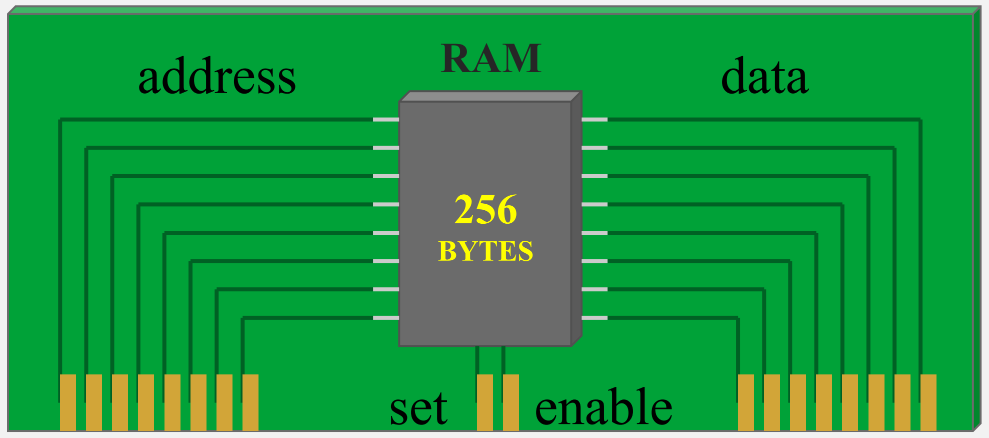
Von Neumann Architecture



ssdf

RAM Structure

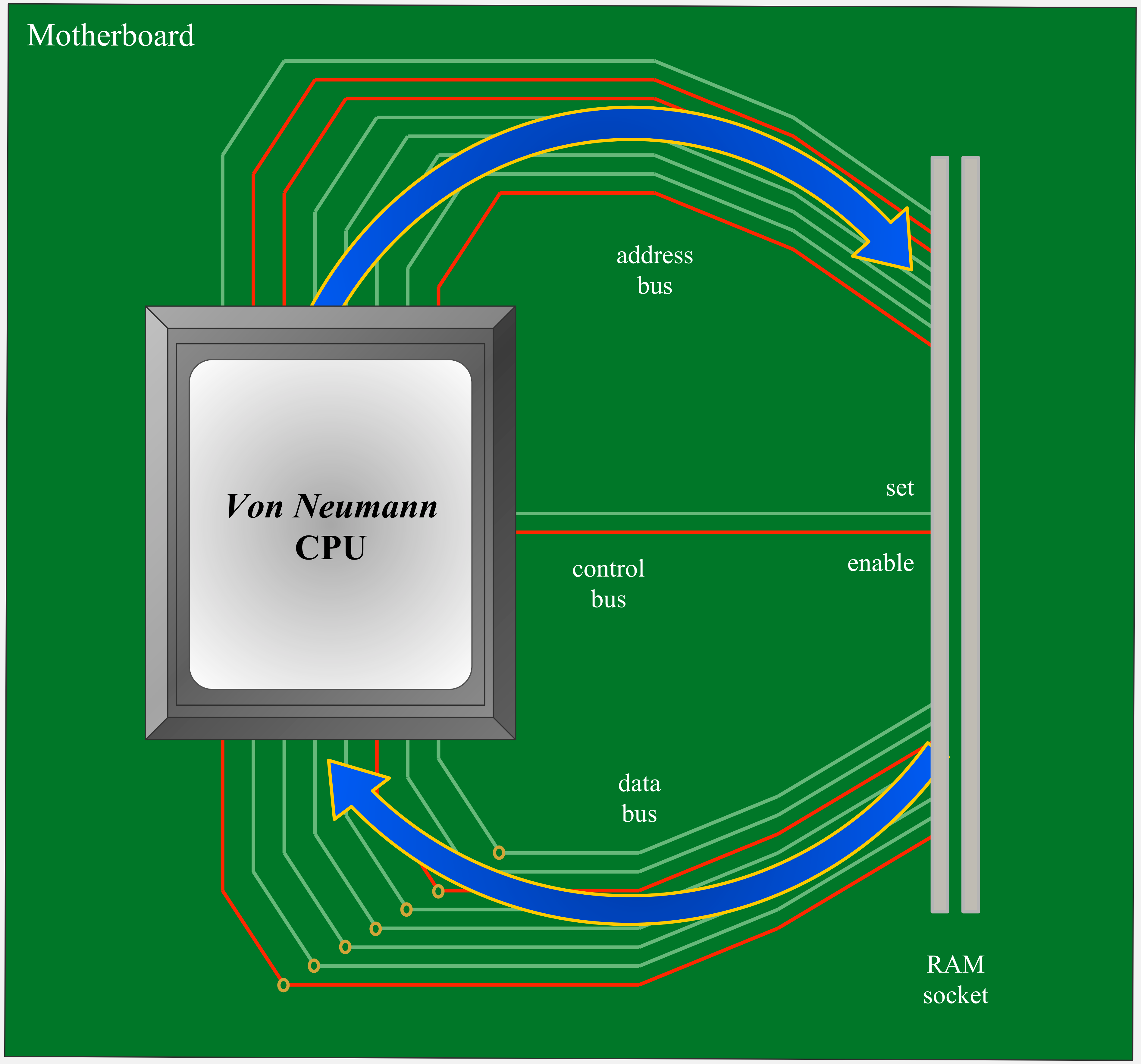


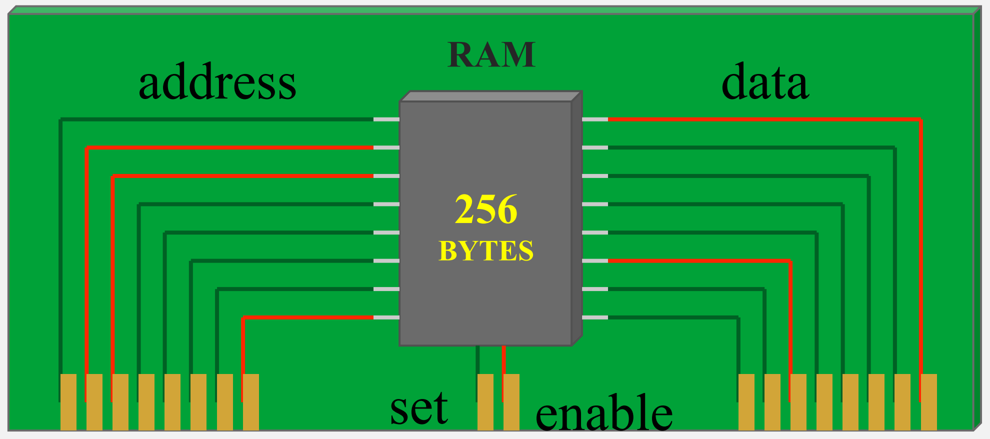


…	…
01100001	00100001
01100010	00001001
01100011	01110000
01100100	00001111
01100101	11110001
01100110	01010010
01100111	11100001
01101000	01111011
01101001	00011000
01101010	01000111
…	…

ssdf

CPU Read from RAM



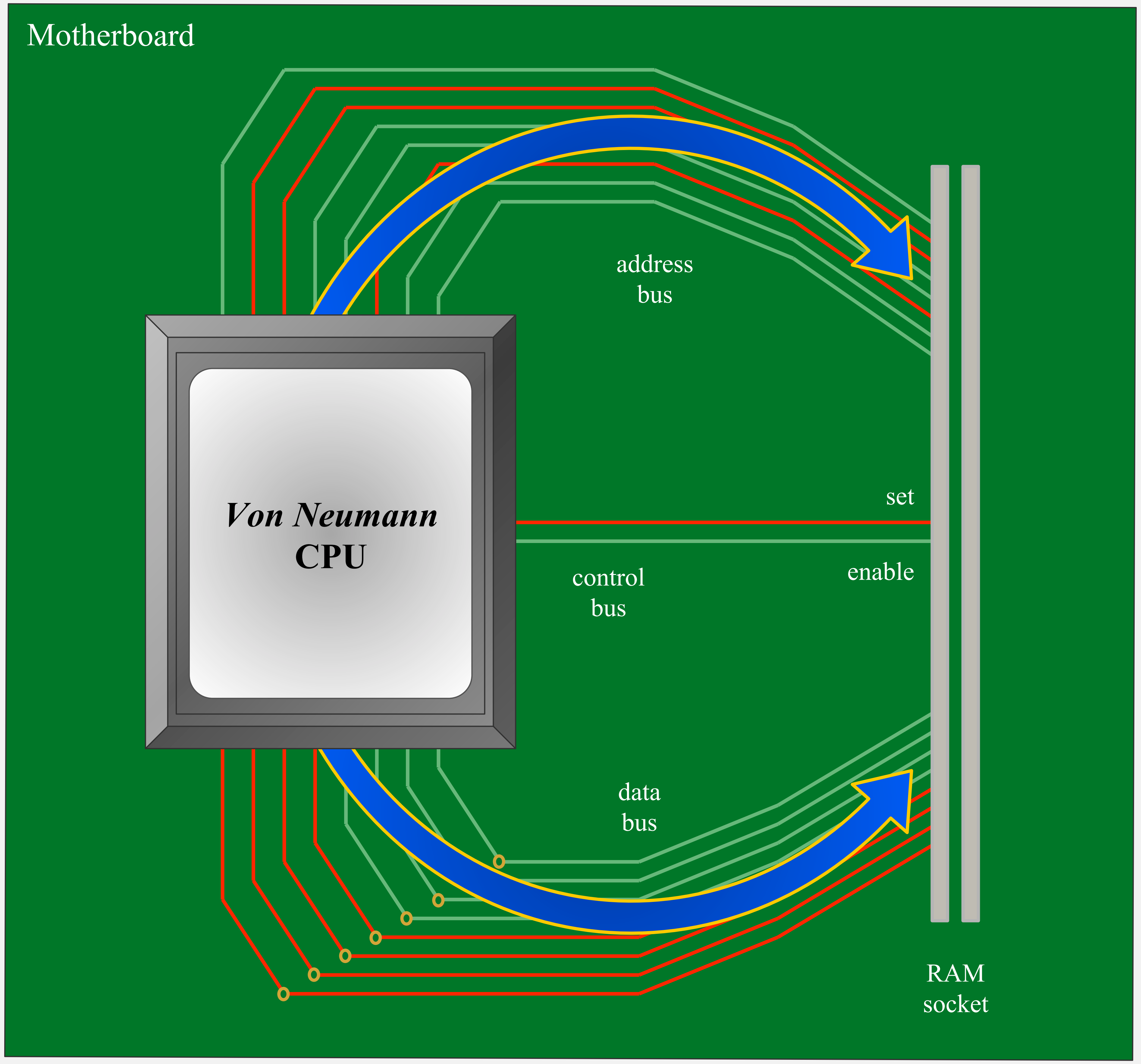


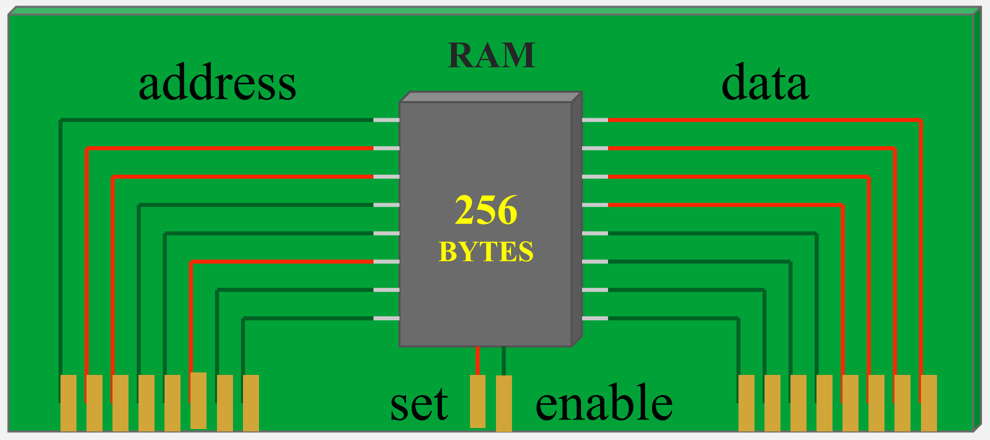
01100001

00100001

ssdf

CPU Writing to RAM



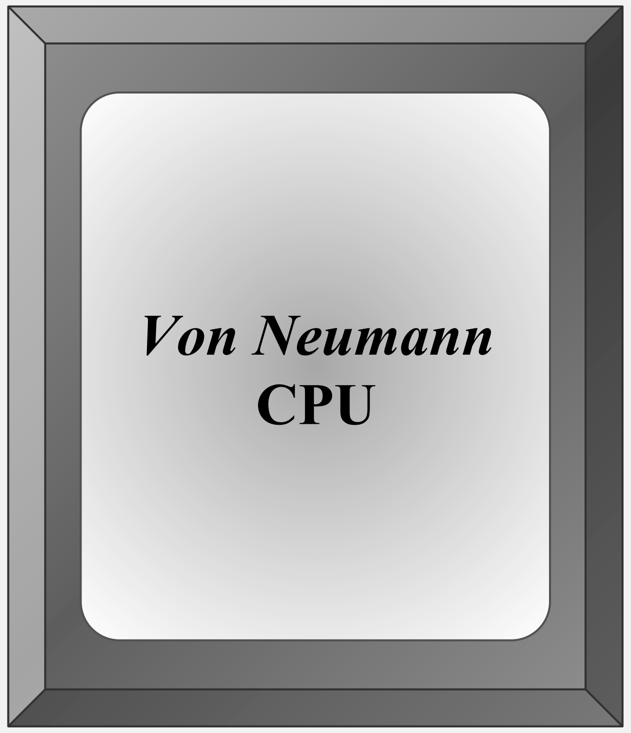


01100100

00001111

ssdf

Instruction Set



Instruction Set

LOAD a number from RAM into the CPU

ADD two numbers together

STORE a number from the CPU back out to RAM

COMPARE one number with another

JUMP IF *Condition* to another address in RAM

JUMP to another address in RAM

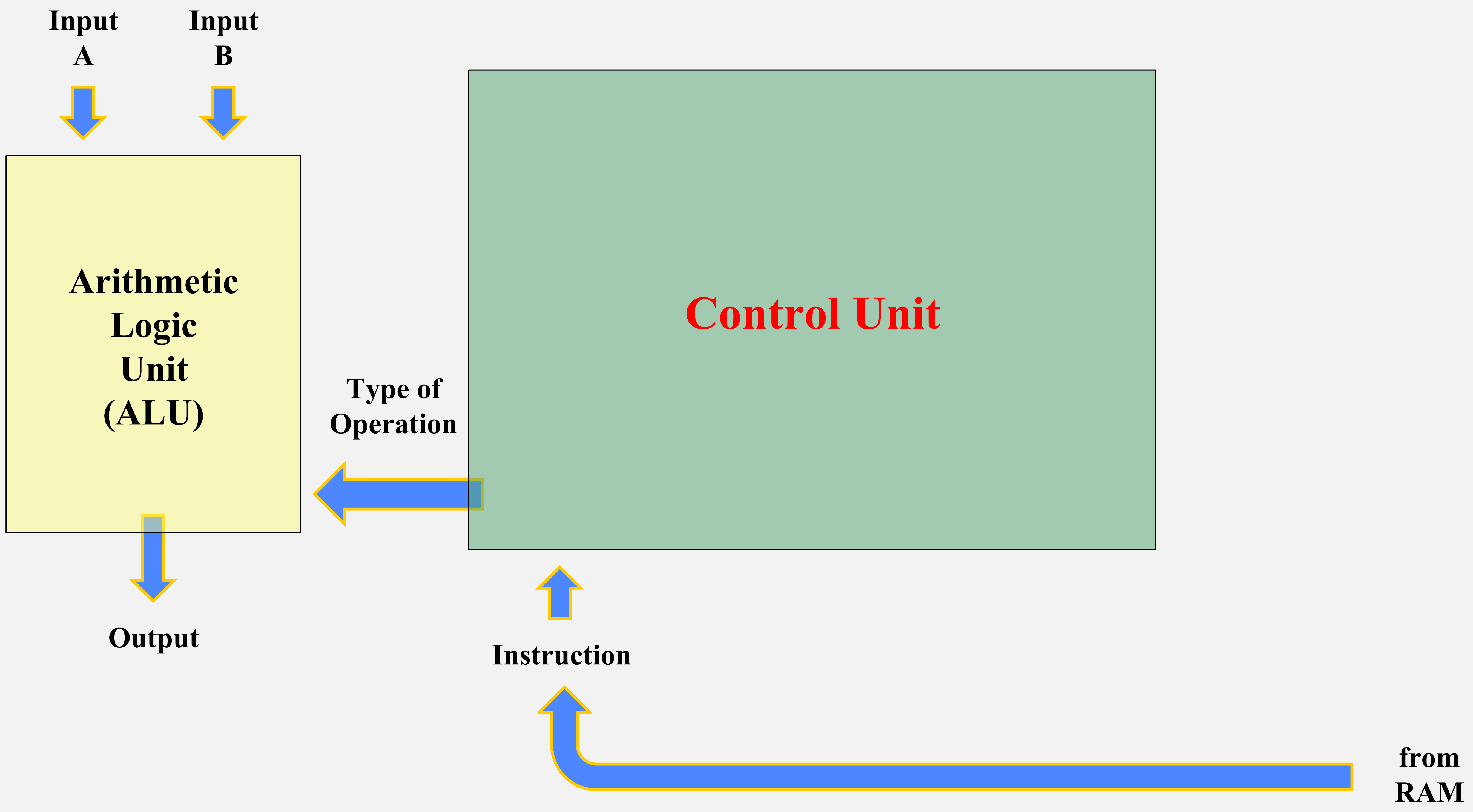
OUTPUT to a device such as a monitor

INPUT from a device such as a keyboard

…	…
01100001	**LOAD**
01100010	**9**
01100011	**IN**
01100100	**Keyboard**
01100101	**COMPARE**
01100110	**JUMP IF =**
01100111	**10100001**
01101000	**OUT**
01101001	**Monitor**
01101010	**"G"**
…	…

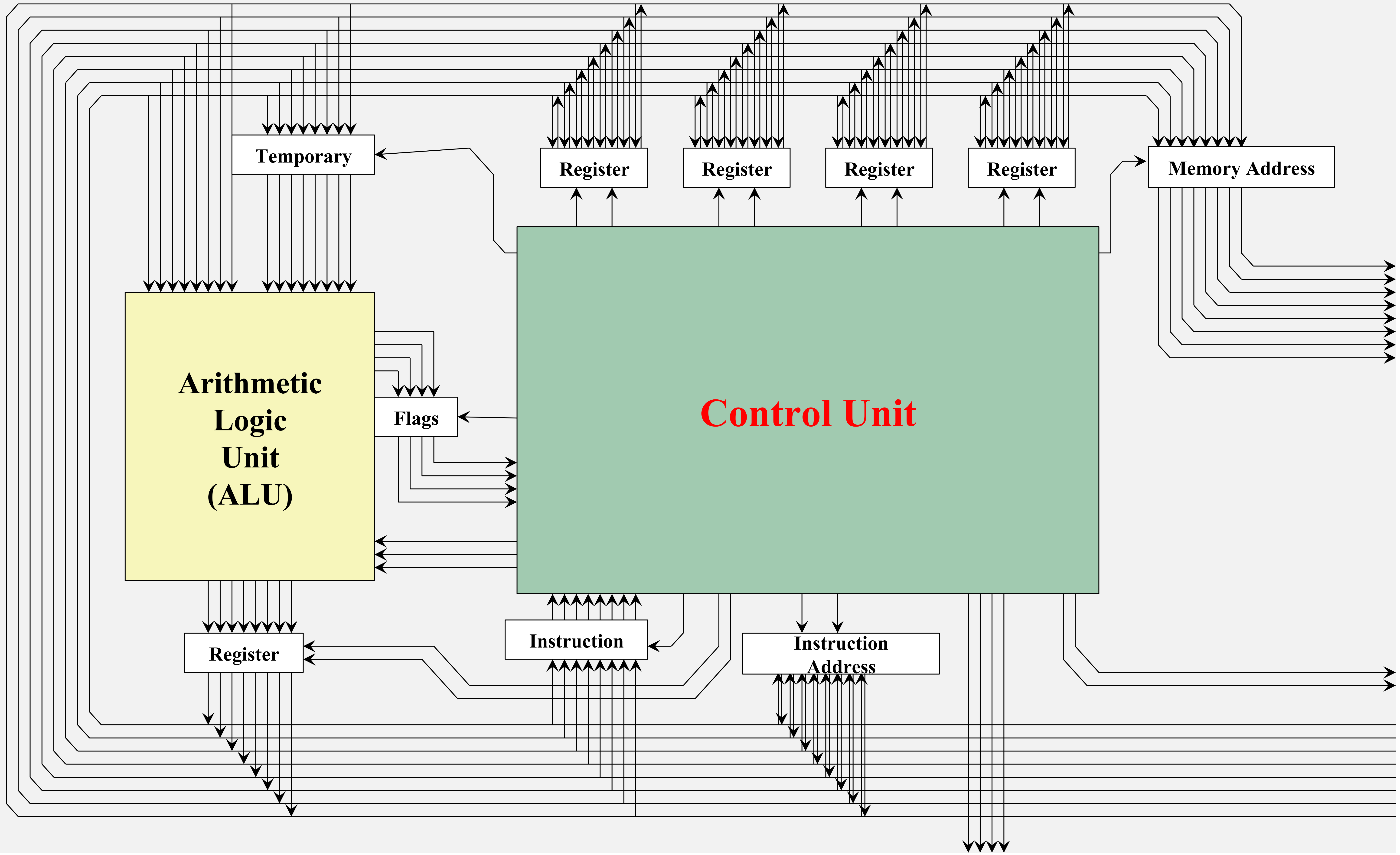
ssdf

Top Level View of Sequential



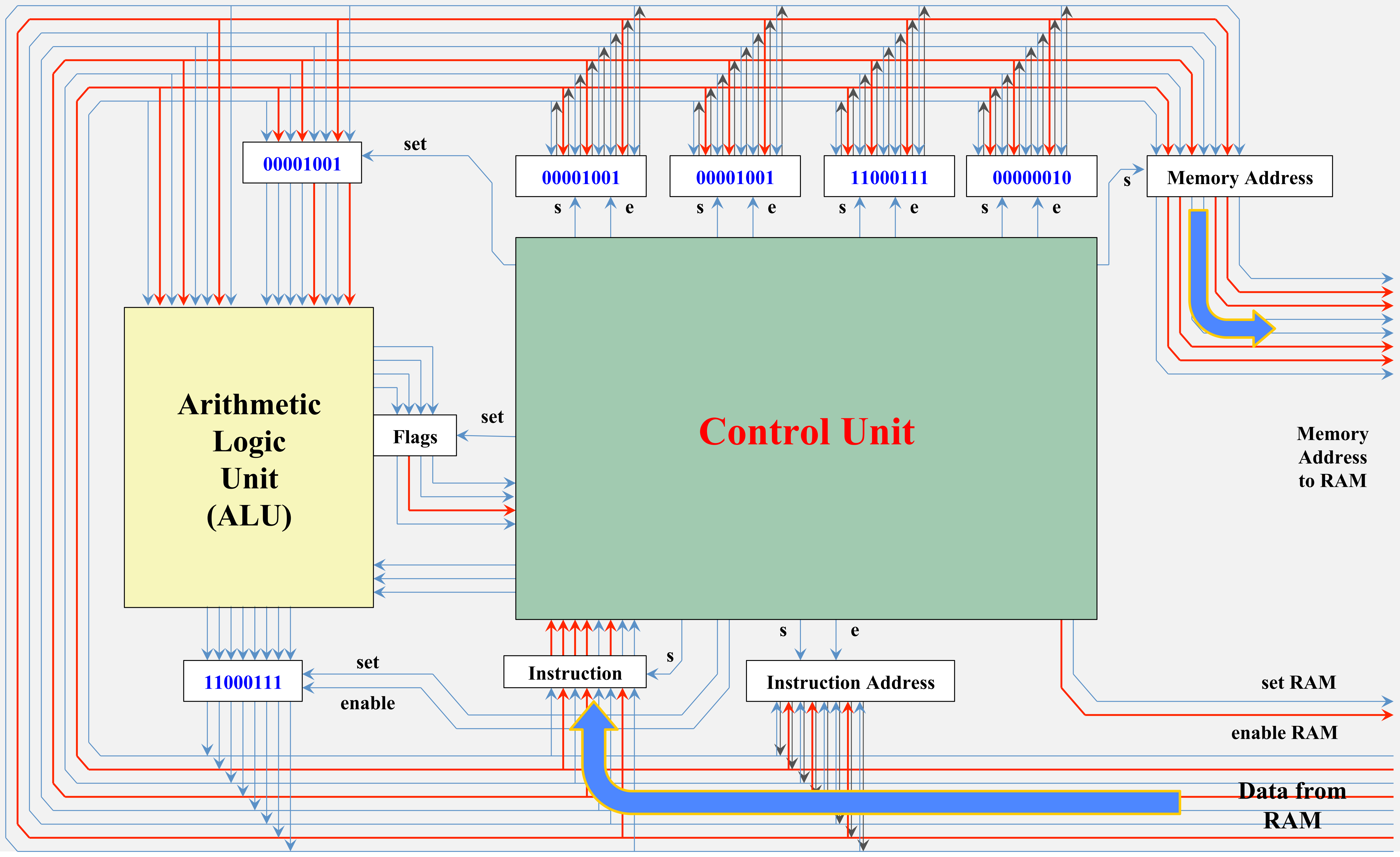
ssdf

Zoom into a Sequential Processor



ssdf

Zoom into a single CPU



ssdf

Advances & Challenges in Computer Systems

- **Central Processing Unit (CPU)**
  - Past decade with tremendous advances in micro-architecture
    - Clock rate increased from 40MHz to over 3.0 GHz
    - Over 1.5 billion transistors in relatively same chip size (about 45nm)
  - Processors can now also execute multiple instructions per clock cycle
  - Memory controller & coherent interconnects moved into the processor chip.

- **Memory System**
  - 1982, with it 10-megabyte disk storage to 200,000+ times larger disk storage by 2014

  - Programmers demand an infinite amount of fast memory

ssdf

Processor and Memory speed gap

❖ **Its not only that the processor performance matters, Feeding data by RAM too.**

➢ DRAM access time have increased at much slower rate than CPU clock rate

➢ To help the processor speed and **memory latency** engineers have created **hierarchy of successively faster memory devices called cache that relay on locality of data to improve performance.**

➢ Another issue is **bandwidth** rate at which the data is pumped from memory to CPU.

ssdf

Defining Performance

Fast, fewer capacity, expensive

Slow, large capacity, cheaper

- What does it mean to say ***x*** is faster than ***y***?
- **Supersonic Jet**
  - 2,485 miles on hour
  - 1-2 max passengers
- **Boeing 777**
  - 440 max passengers
  - 559 miles per hour
- **Latency vs. Bandwidth**
  - **Latency:** how long does it take for a block of memory from level k+1 to be transferred to level k?
  - **Bandwidth:** the rate at which the data can be pumped from the memory to the processor determines the bandwidth of memory system.

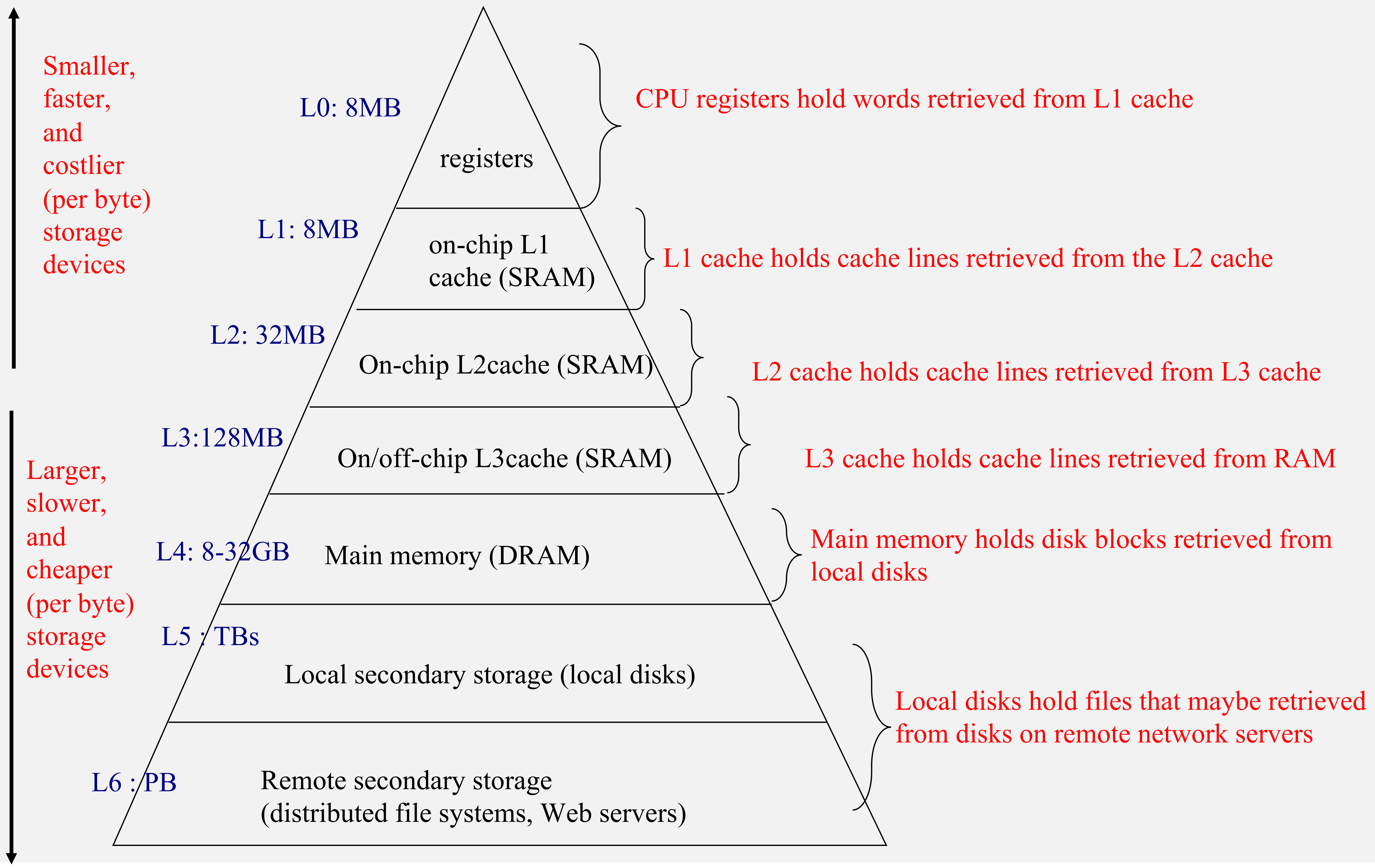
ssdf

Types of Memory

- **Volatile/Power-On Memory**
  - Random-Access Memory (RAM)
    - Static RAM (SRAM)
    - Dynamic RAM (DRAM)
- **Nonvolatile/Power-Off Memory**
  - Variety of Nonvolatile Memory exits
    - Read only Memory (ROM)
    - Programmable ROM (PROM)
    - Erasable Programmable ROM(EPROM)
    - Electrically EPROM
      - Cell phones, SSD, Flash Memory, PS3, XBOX, etc…

ssdf

Memory Hierarchy

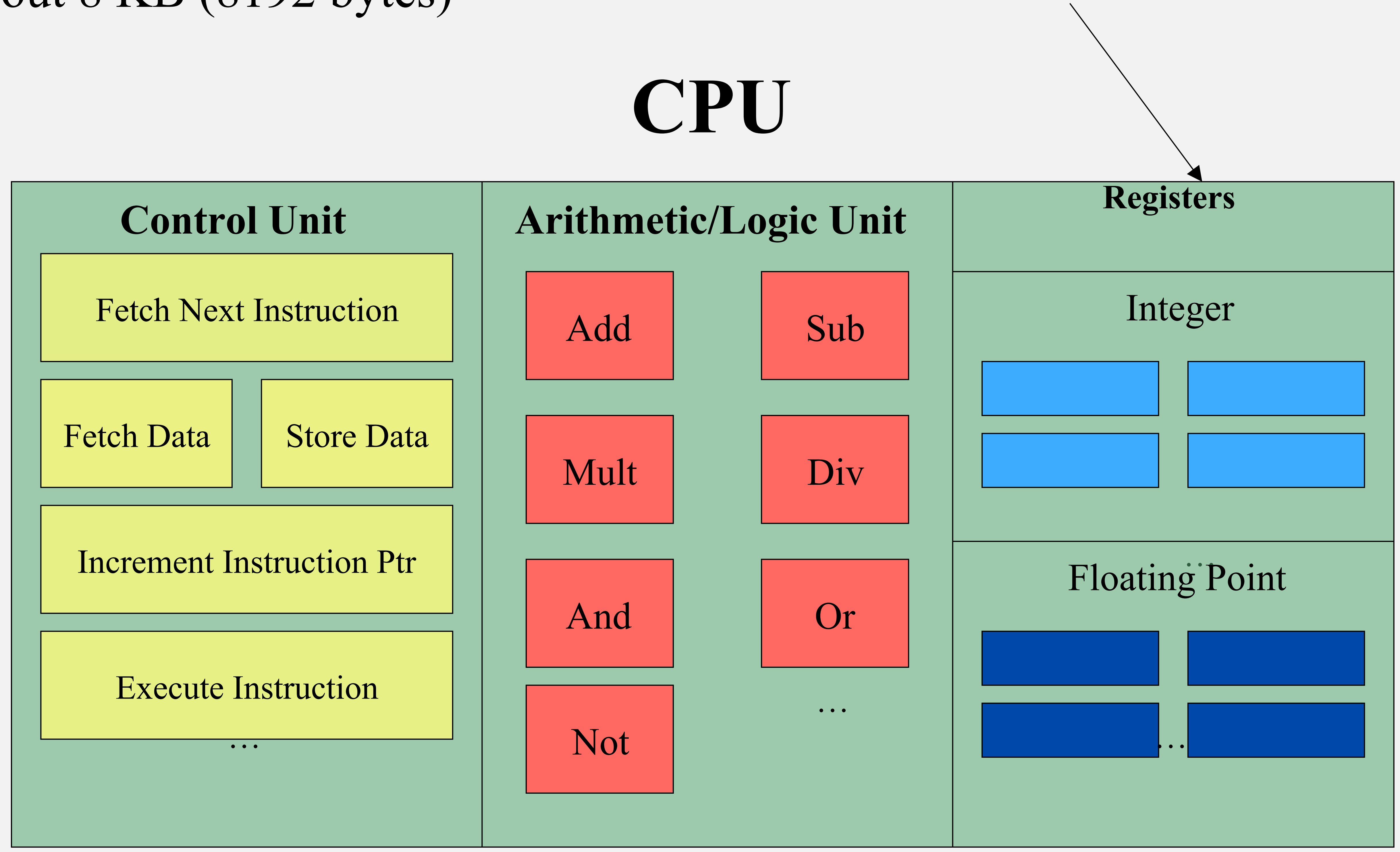

Smaller,
faster,
and
costlier
(per byte)
storage
devices
Larger,
slower,
and
cheaper
(per byte)
storage
devices
L0: 8MB
registers
L1: 8MB
on-chip L1
cache (SRAM)
L2: 32MB
On-chip L2cache (SRAM)
L3:128MB
On/off-chip L3cache (SRAM)
L4: 8-32GB
Main memory (DRAM)
L5 : TBs
Local secondary storage (local disks)
L6 : PB
Remote secondary storage
(distributed file systems, Web servers)
CPU registers hold words retrieved from L1 cache
L1 cache holds cache lines retrieved from the L2 cache
L2 cache holds cache lines retrieved from L3 cache
L3 cache holds cache lines retrieved from RAM
Main memory holds disk blocks retrieved from local disks
Local disks hold files that maybe retrieved from disks on remote network servers

ssdf

What are Registers?

Registers are memory-like locations inside the Central Processing Unit that hold data that are **being used right now** in operations.

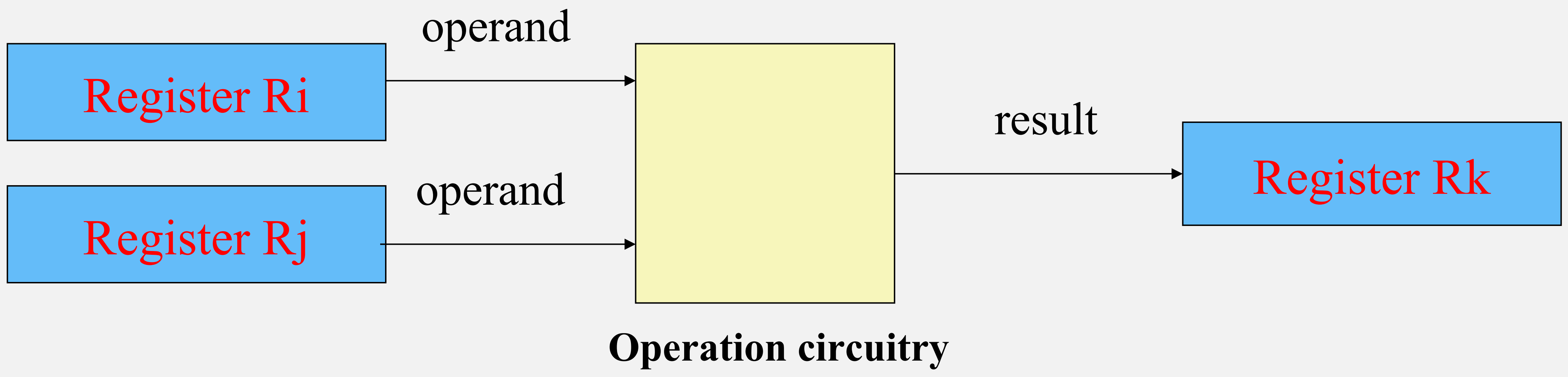
About 8 KB (8192 bytes)



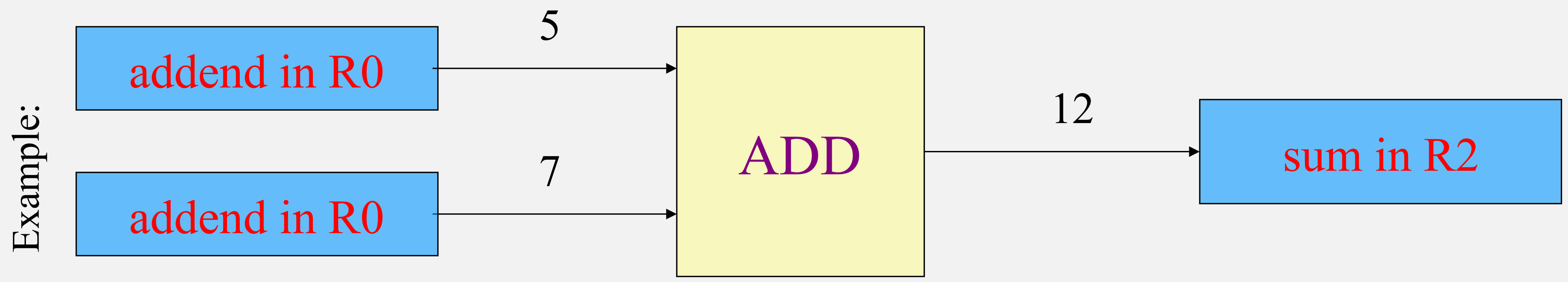
ssdf

How Registers works?

- Every arithmetic or logical operation has one or more operands and one result.
- Operands are contained in source registers.
- A “yellowish box” of circuits performs the operation.
- The result goes into a destination register.



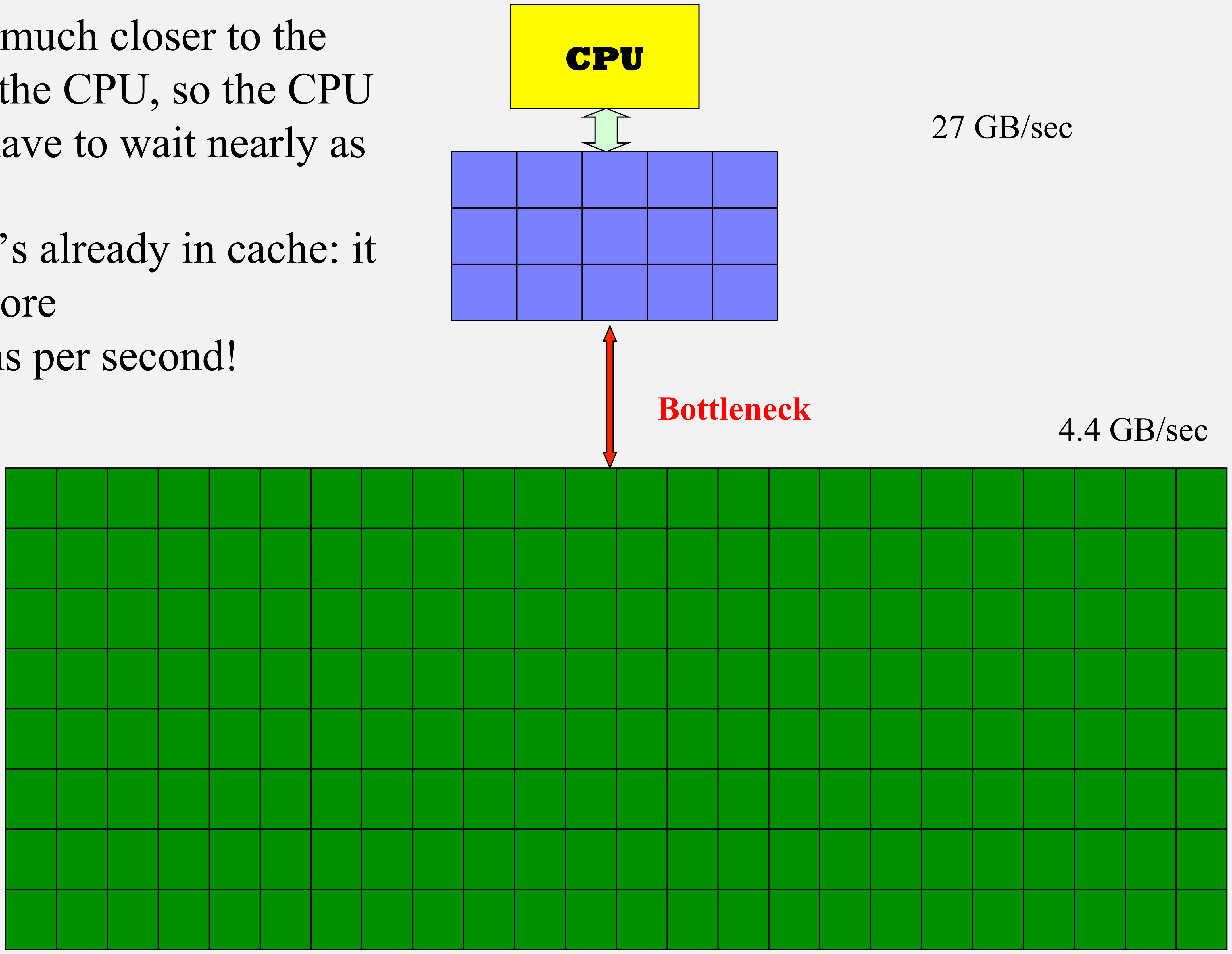
Operation circuitry



ssdf

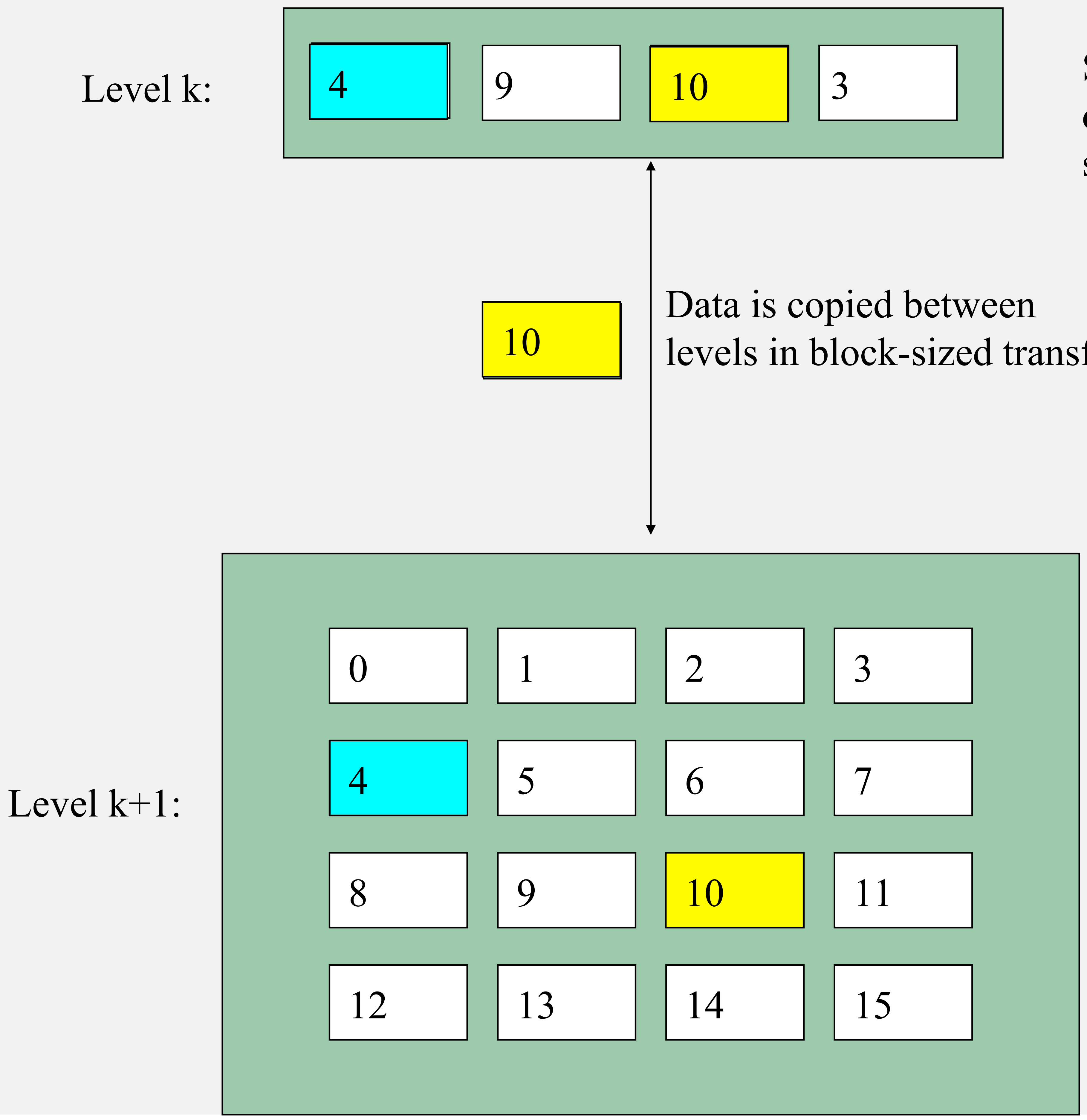
Why Have Cache?

Cache is much closer to the speed of the CPU, so the CPU doesn’t have to wait nearly as long for
stuff that’s already in cache: it can do more
operations per second!



ssdf

Caching in a Memory Hierarchy?



Smaller, faster, more expensive device at level k caches a subset of the blocks from level k+1

Data is copied between levels in block-sized transfer units

Larger, slower, cheaper storage device at level k+1 is partitioned into blocks.

ssdf

Principal of locality in cache memory

- In Principal that an access of one location is followed by an access of nearby location is often called **locality**

- Usually when access one memory location (instruction/data) typically access:
  - Nearby location (**spatial locality**)
  - Near future (**temporal locality**)

- Example is the user of arrays:

```
float z[1000];
…
sum = 0.0;
      for (i=0; i<1000; i++)
              sum +=z[i];
```

- Arrays are allocated as block of contiguous memory locations
- Using wider busses you can access blocks of memory locations
  - Cache-block / cache-line

ssdf

Effects of Memory Latency on performance

- **Suppose:**
  - Memory block = 1 word
  - processor peak = 1 GHz
  - DRAM Latency = 100ns
  - multiply-add units = 2
  - number of instruction executed per cycle (1ns) = 4

- **What do you think the actual peak performance of the processor is?**

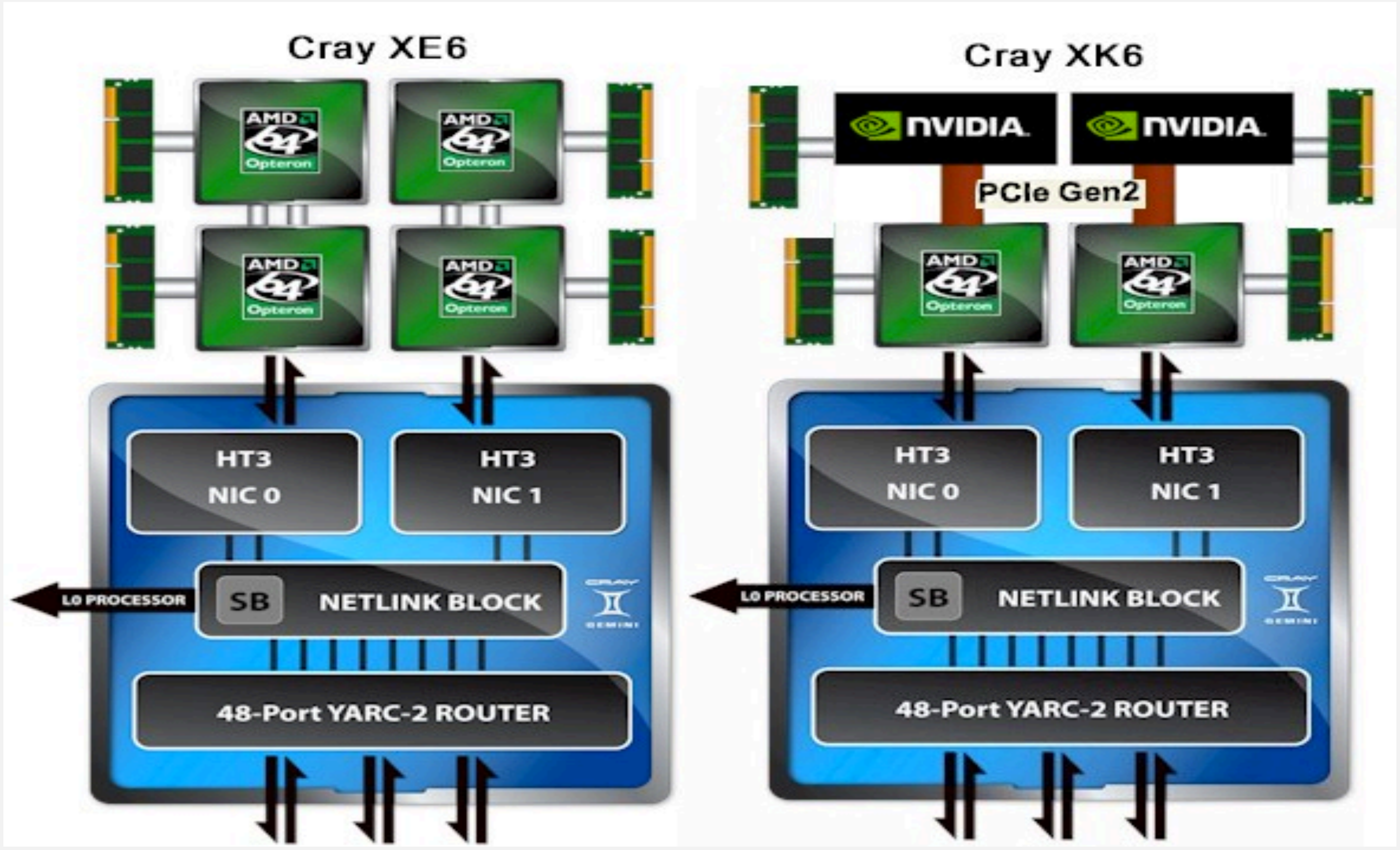
ssdf

Performance of computing dot-product

- If we were computing the dot-product of two vectors on this platform a dot product performs one multiply-add on a single pair of vector elements.
  - Each floating point operation requires one data fetch
  - So the peak speed of this computing is limited to one floating point operation every 100ns or a speed of 10 MFLOPS. A very small fraction of the peak performance of the processor
- Solution to this problem:
  - **Add cache to CPU**
  - **Increase the bus lines** (4 words)

ssdf

Cray Node layout



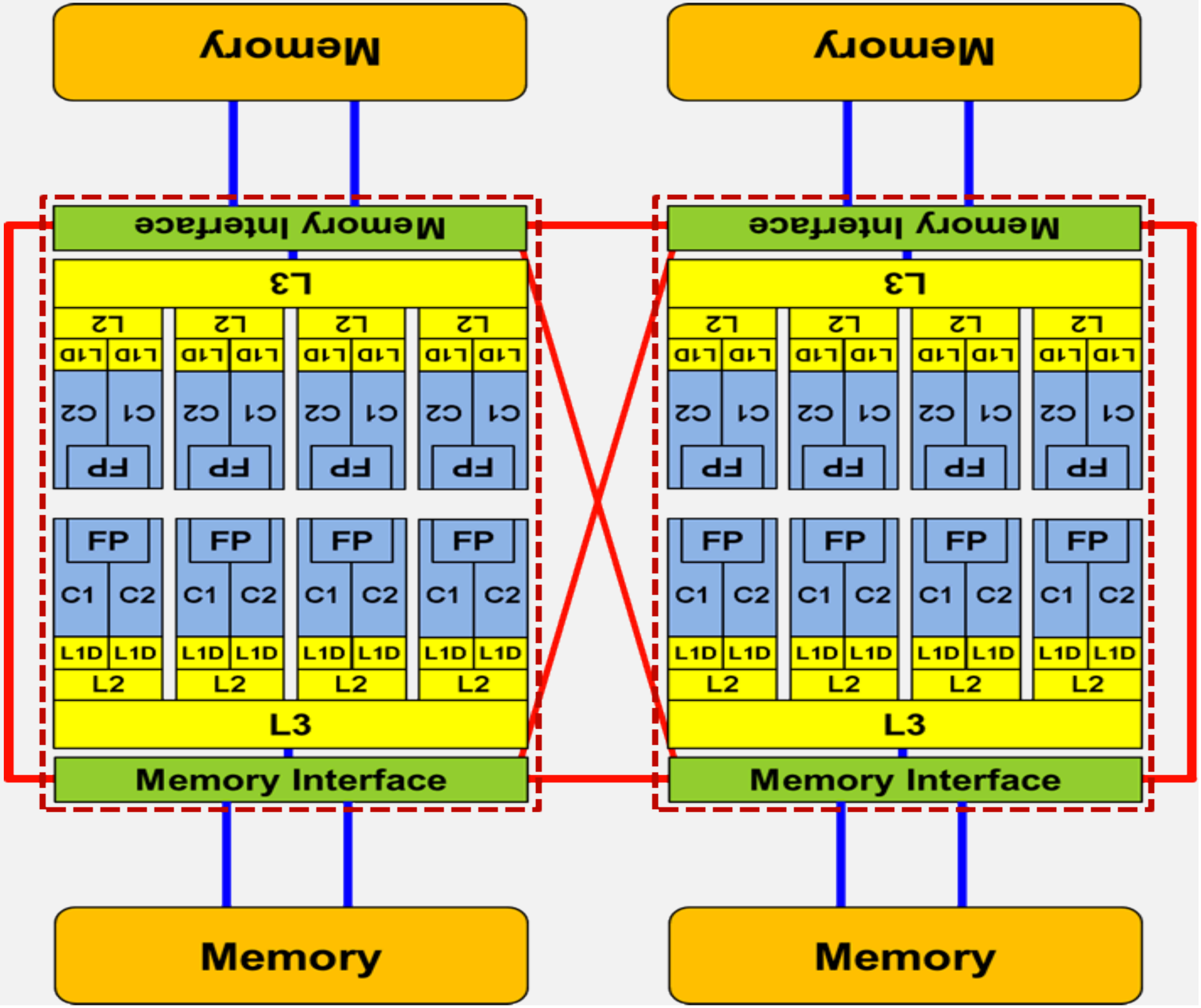
ssdf

Cary XE Cache layout

L1 ~ 4 CPU Cycles
L2 ~ 20 CPU Cycles
L3 ~ 45 CPU Cycles

8x64 KB L1 instruction cache, 16x16 KB L1 data cache, 8x2 MB L2 cache per processor core, 2x8 MB shared L3 cache

Memory Bandwidth: Up to 102.4 GB/s per node



Hard Disk

ssdf

Why Hard Disks are Slow?

- Your hard disk is **much much** slower than main memory (factor of 10-1000). **Why?**

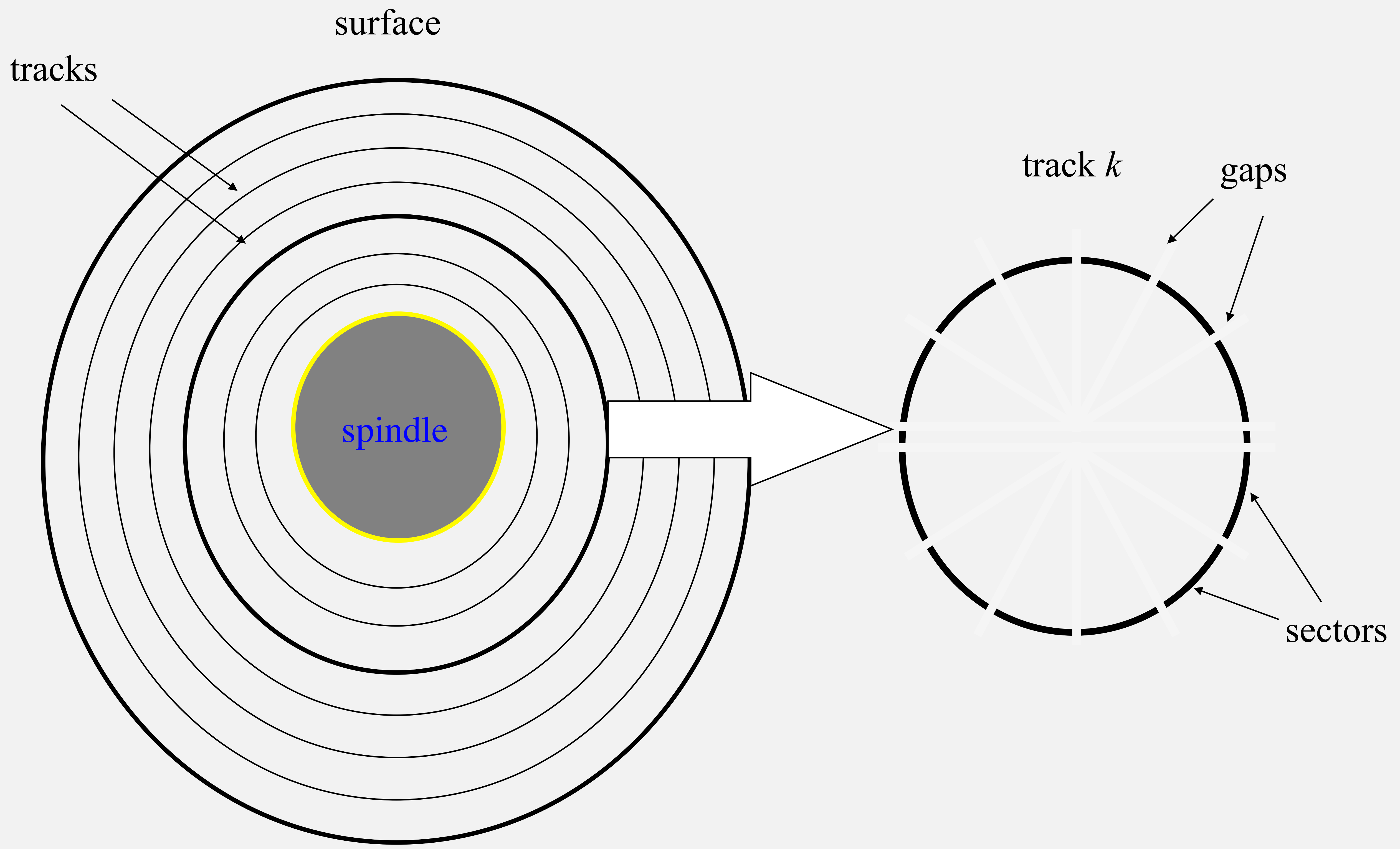
- Well, accessing data on the hard disk involves physically moving:
  - **the disk platter**
  - **the read/write head**

- hard disk is very slow because **objects** move much slower than **electrons**

ssdf

Disk Geometry

- ❖ Disks consist of platters, each with two surfaces
- ❖ Each surface consists of concentric rings called tracks
- ❖ Each track consists of sectors separated by gaps

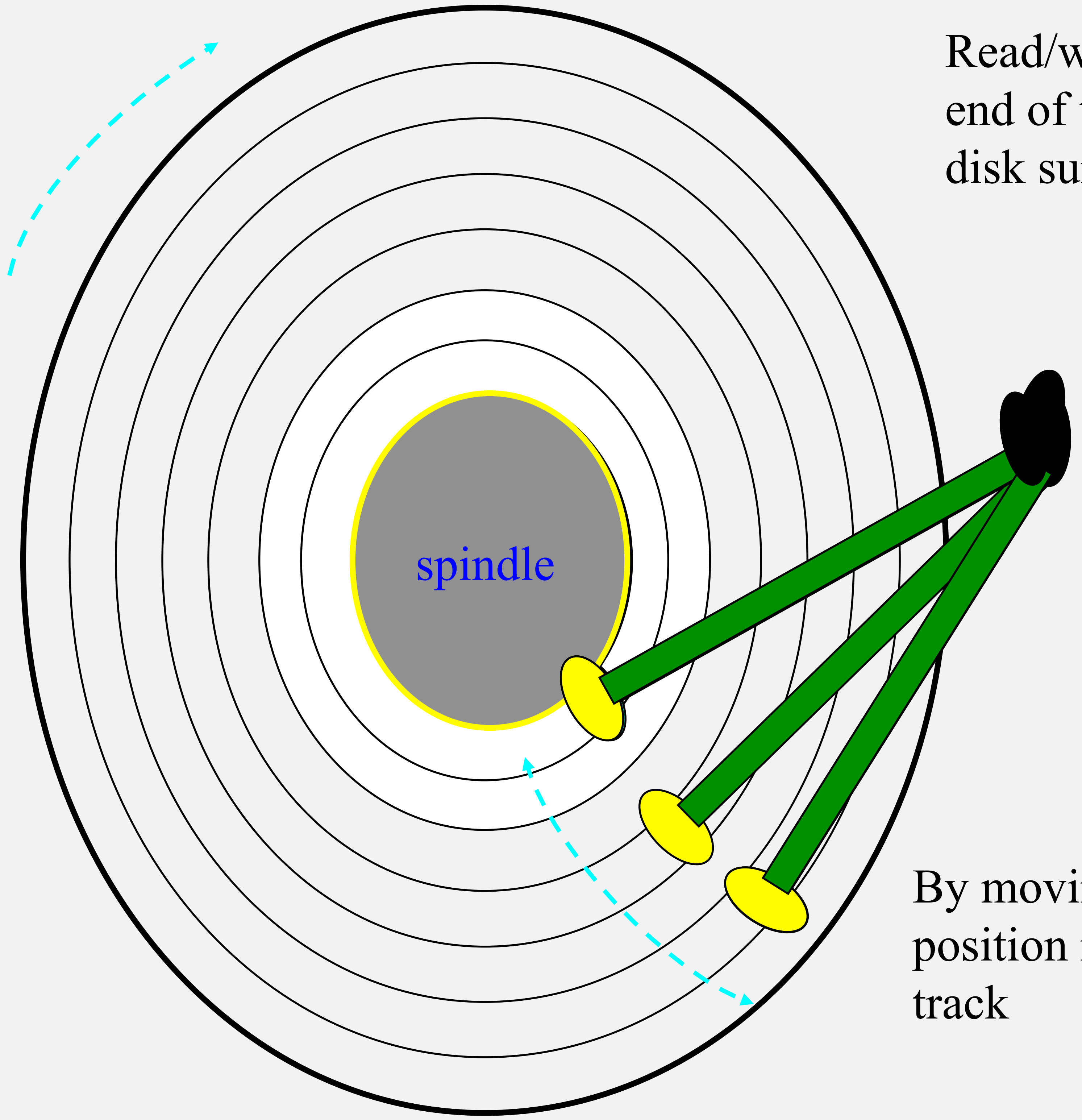


ssdf

Disk Operation (Single-Platter View)

The disk surface spins at a fixed rotational rate

Read/write *head* is attached to end of the *arm* and flies over disk surface on thin cushion of air

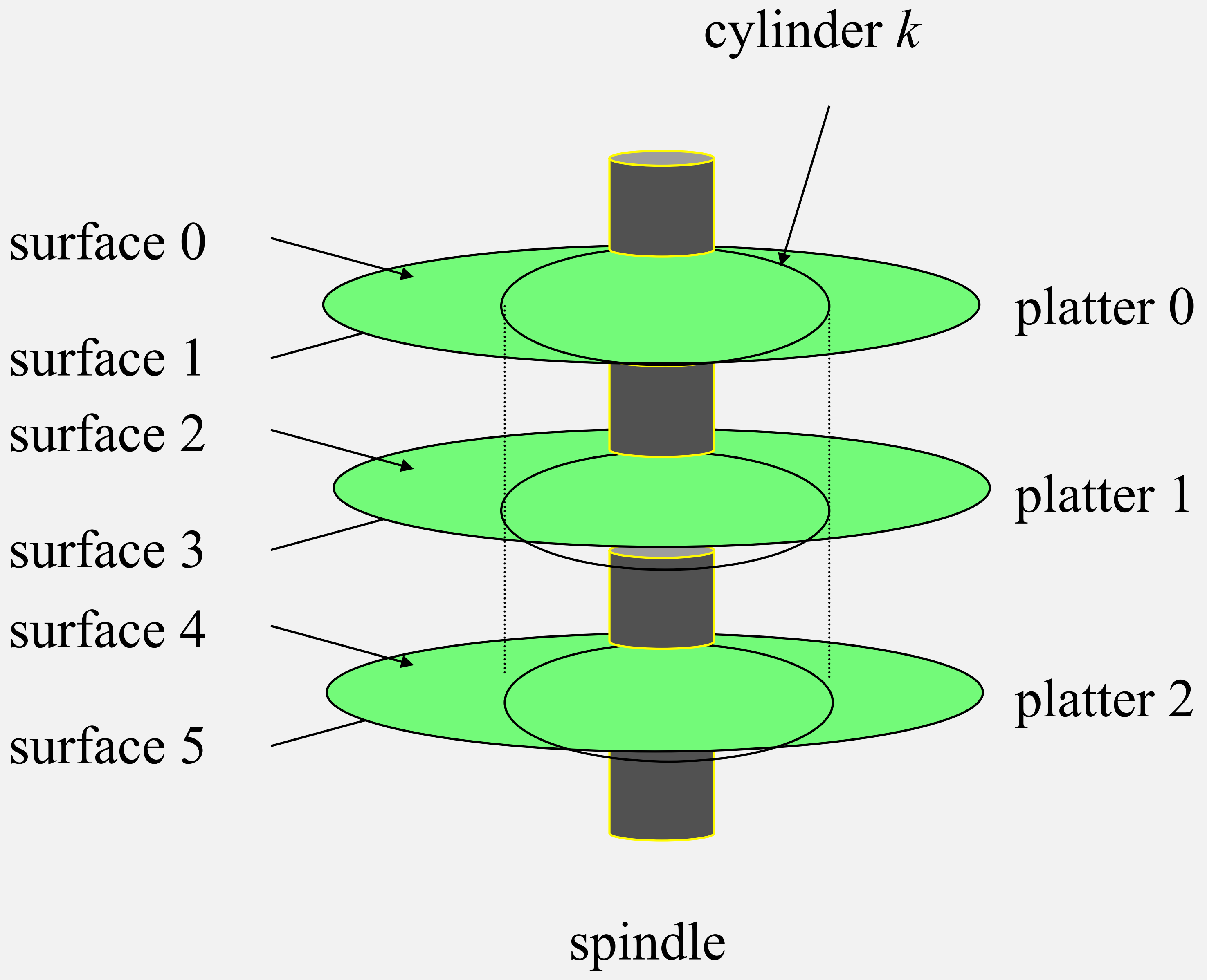


By moving radially, arm can position read/write head over any track

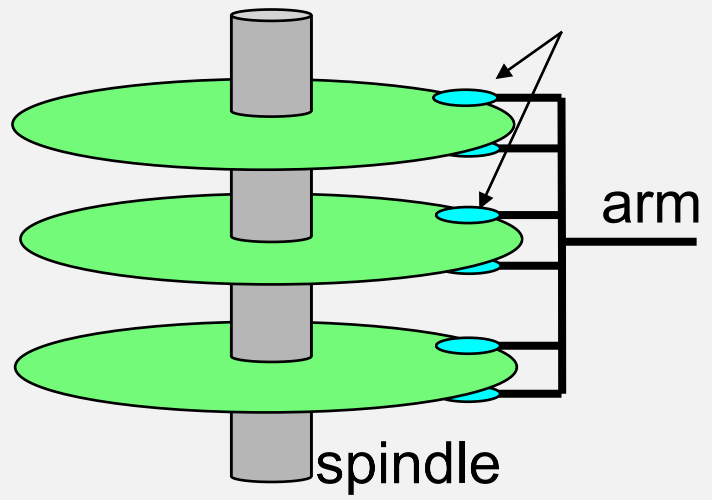
ssdf

Disk Geometry (Muliple-Platter View)

- Aligned tracks form a cylinder



read/write heads
move in unison
from cylinder to cylinder



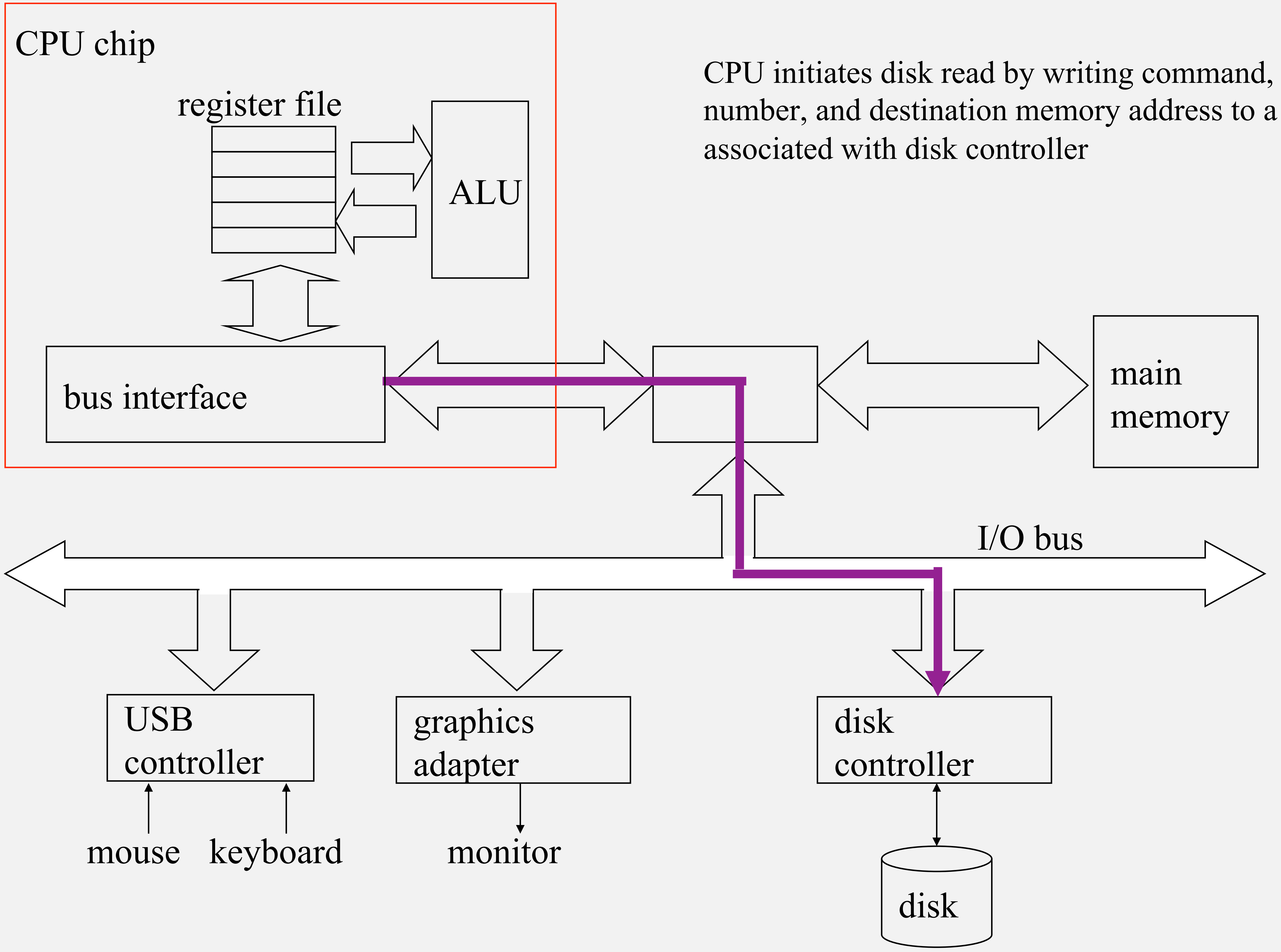
ssdf

Disk Access Time

- ❖ Average time to access some target sector approximated by :
  - ➢ **Taccess = Tavg seek + Tavg rotation + Tavg transfer**
- ❖ **Seek time** (**Tavg seek**)
  - ➢ Time to position heads over cylinder containing target sector
  - ➢ Typical Tavg seek = 9 ms
- ❖ **Rotational latency** (**Tavg rotation**)
  - ➢ Time waiting for first bit of target sector to pass under r/w head
  - ➢ Tavg rotation = 1/2 x 1/RPMs x 60 sec/1 min
- ❖ **Transfer time** (**Tavg transfer**)
  - ➢ Time to read the bits in the target sector.
  - ➢ Tavg transfer = 1/RPM x 1/(avg # sectors/track) x 60 secs/1 min

ssdf

Reading a Disk Sector (1)

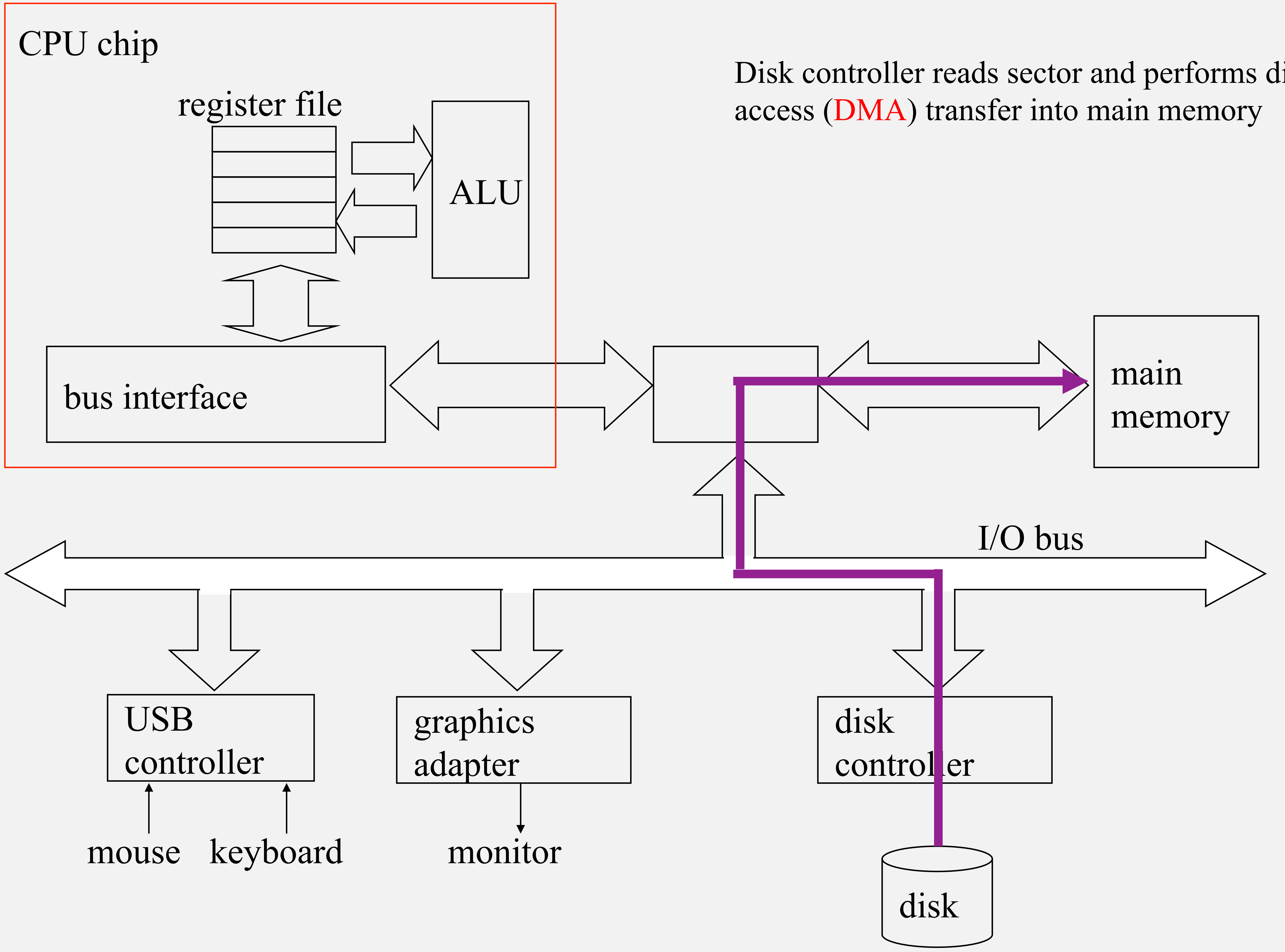


CPU initiates disk read by writing command, logical block number, and destination memory address to a port (address) associated with disk controller

ssdf

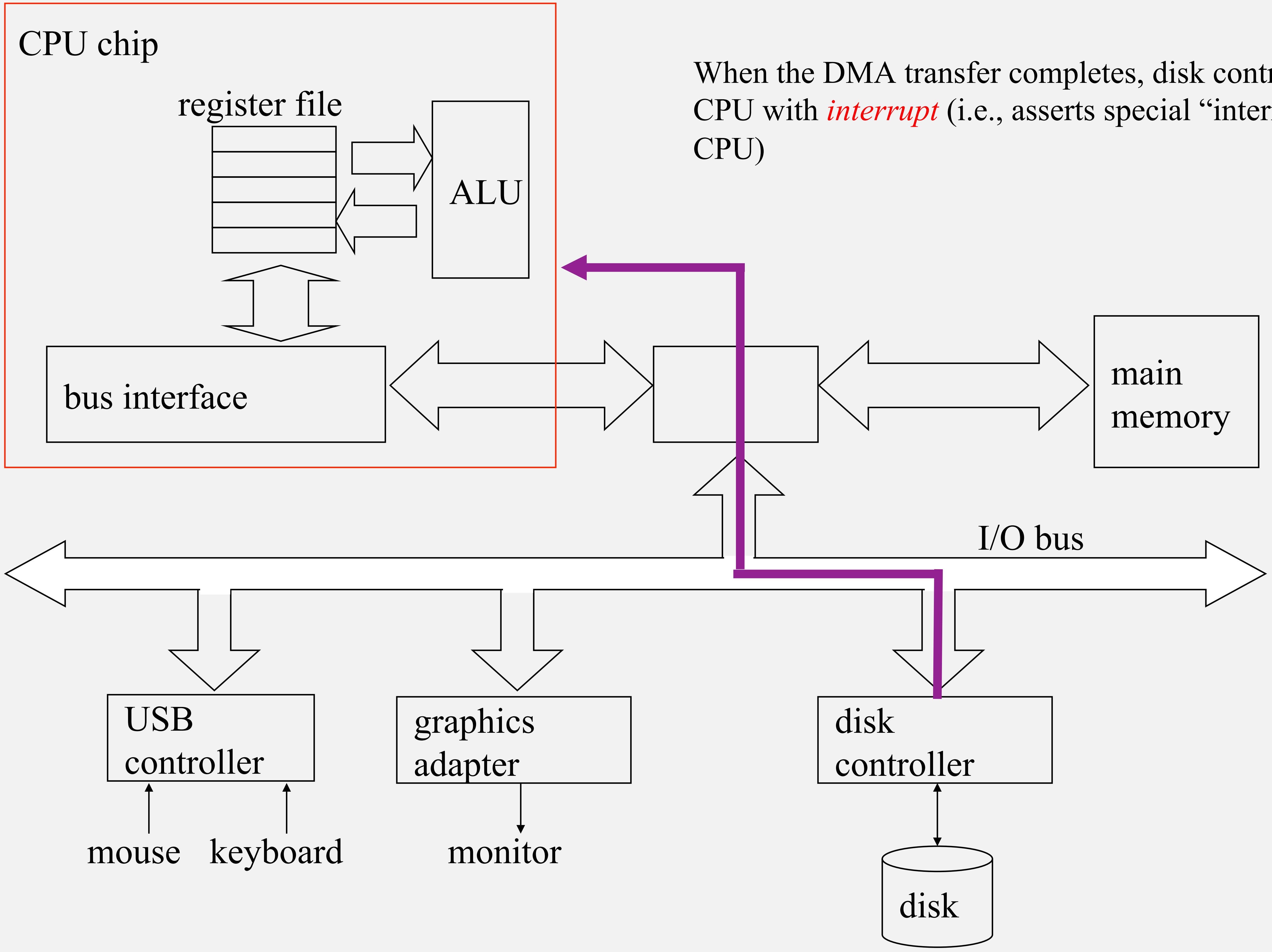
Reading a Disk Sector (2)

Disk controller reads sector and performs direct memory access (DMA) transfer into main memory



ssdf

Reading a Disk Sector (3)

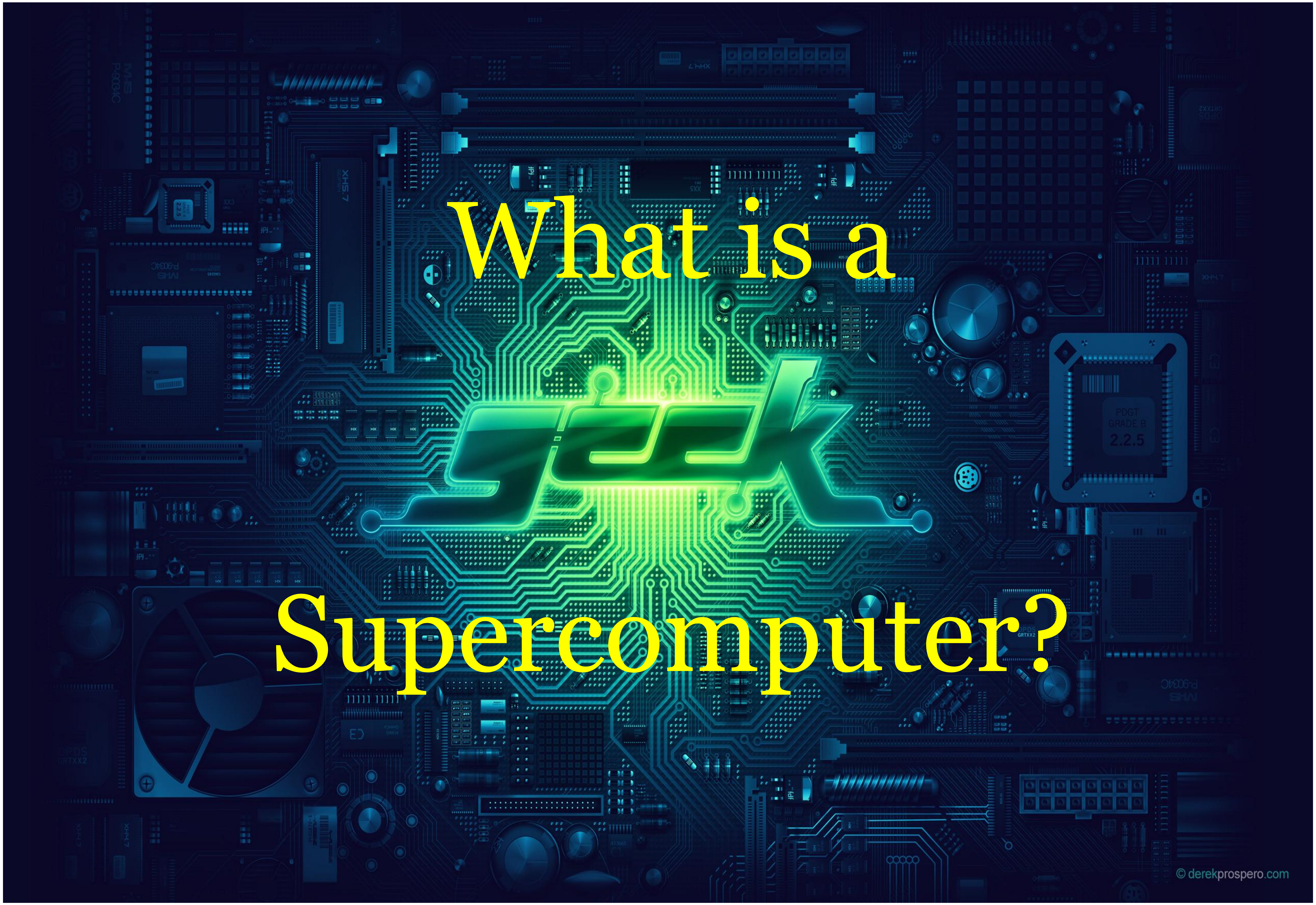


When the DMA transfer completes, disk controller notifies CPU with *interrupt* (i.e., asserts special “interrupt” pin on CPU)

ssdf

Programming strategies for memory usage

- **Register reuse**: do a lot of work on the same data before working on new data.
- **Cache reuse**: the program is much more efficient if all of the data and instructions fit in cache; if not, try to use what's in cache a lot before using anything that isn't in cache (e.g., tiling).
- **Data locality**: try to access data that are near each other in memory before data that are far.
- **I/O efficiency**: do a bunch of I/O all at once rather than a little bit at a time; don't mix calculations and I/O.



What is a Supercomputer?

ssdf

What is a supercomputer?

- ❖ *"A state of the art computer with highest processing speed possible at any given time, and is use for solving scientific & engineering problems"* Encarta Dictionary
  - ➢ Measured in Petaflops (1,000,000,000,000,000,000)
  - ➢ Top500 list (top500.org)
  - ➢ Customized compute nodes (blades)
    - ■ Multiple Nodes on a blade (CPU, GPUs)
  - ➢ Connected via proprietary interconnects (optical)

Top Supercomputer wish to come in top10-20s next year list.

ssdf

Top 5 Supercomputers (top500.org)

1. **Tianhe-2 (China)**
   - ➢39.9 Petaflops on 3,120,000 Cores
   - ➢Mix of Intel Xeon Phi and Xeon E5 Custom processor
   - ➢Run custom version of Ubuntu
2. **Titan (USA) [Cray]**
   - ➢ 17.6 Petaflops on 561,000 Cores
   - ➢AMD Opteron Processor and Nvidia GPUs
3. **Sequoia (USA) [#1 in June 2012] [IBM]**
   - ➢ 17.2 Petaflops on 1.6 million cores
4. **K Computer (JAPAN) [Fujitsu]**
   - ➢10.5 Petaflops on 705,000 cores

10. **SuperMUC (Germany) [4th in June] [IBM]**
    - ➢ Performance at 2.9 Petaflops on 150,000 processing cores
    - ➢ Water cooled (water takes 4000 times more heat than air)

ssdf

Sources

Cray XE 6: http://www.cray.com/Assets/PDF/products/xe/CrayXE6Brochure.pdf
Cray XE6 (Garnet) User Guide: http://www.erdc.hpc.mil/docs/garnetUserGuide.html
Best Practice User Guide – Cray XE6 (V2.2):
http://www.prace-ri.eu/html/cray_guide/T7.3c_cray_best_practice.html
http://www.hector.ac.uk/cse/documentation/Phase2b/
AMDZone: http://www.amdzone.com/phpbb3/viewtopic.php?f=532&t=138591

But How Do It Know: http://www.buthowdoitknow.com/