In Lecture 2 .. HW1 due 18/8

- Real data: IEEE 754 32 bit Single Precision (s[1], e[8], f[23])
  - Normalized form: e from 1 to 254

$$\left(-1\right)^{s} \times 1.f \times 2^{e-127}$$

  - Denormalized form: e = 0

$$\left(-1\right)^{s} x\, 0.f\; x\, 2^{-126}$$

  - Special cases: e = 255
- Machine instructions: Examples
- Processor: Control, Registers, ALU
- Memory: Mechanisms of remembering

Double Precision: C double

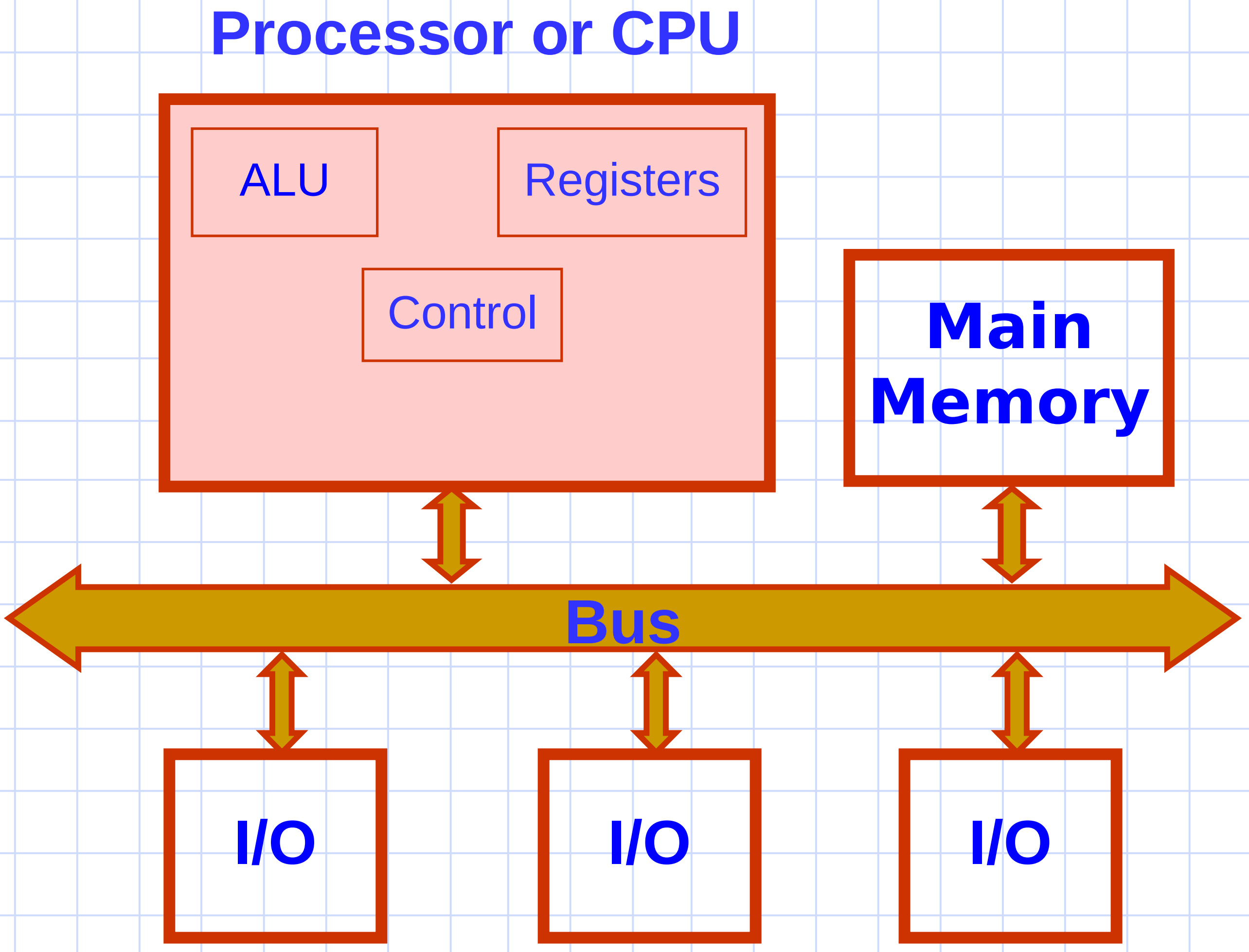
64 bit value with 3 components (*s*,*e*,*f*)

1. *s* (1 bit sign)
2. *e* (11 bit exponent)
   - 00000000000 – 11111111111 (i.e., 0 – 2047)
   - Exponent value = e – 1023 “excess 1023”
   - Range of exponent values: -1022 to 1023
3. *f* (52 bit fraction)

represents the value

$$\left(-1\right)^{s} \times 1.f \times 2^{e-1023}$$

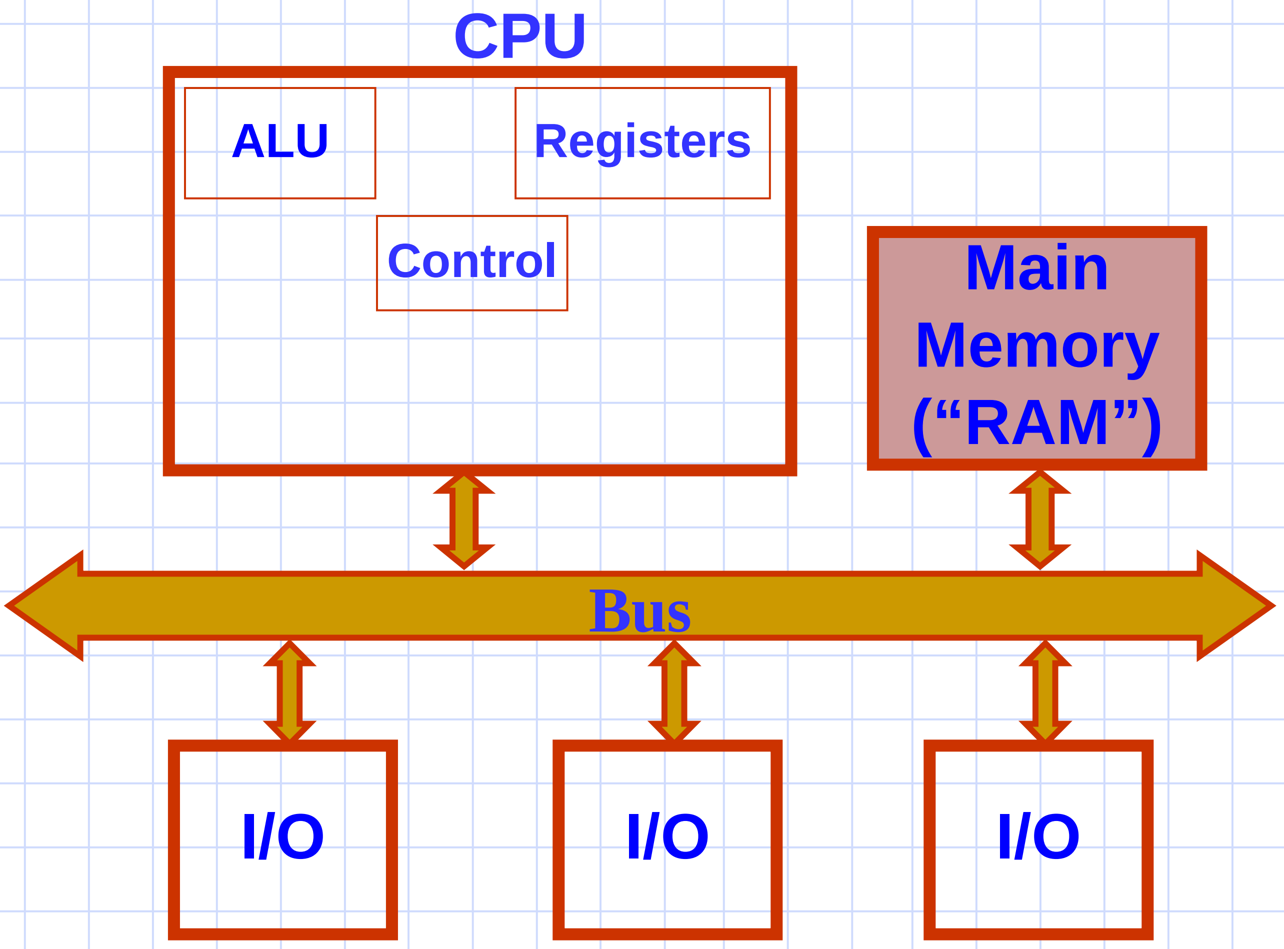
Basic Computer Organization



General Purpose Registers

- Available for use by programmer, possibly for keeping frequently used data
- Why? Since there is a large speed disparity between processor and main memory
  - 2 GHz Processor: 0.5 nanosecond time scale
  - Main memory: ~ 50-100 nsec time scale
- Machine instruction operands can come from registers or from main memory
- But CPUs do not provide a large number of general purpose registers

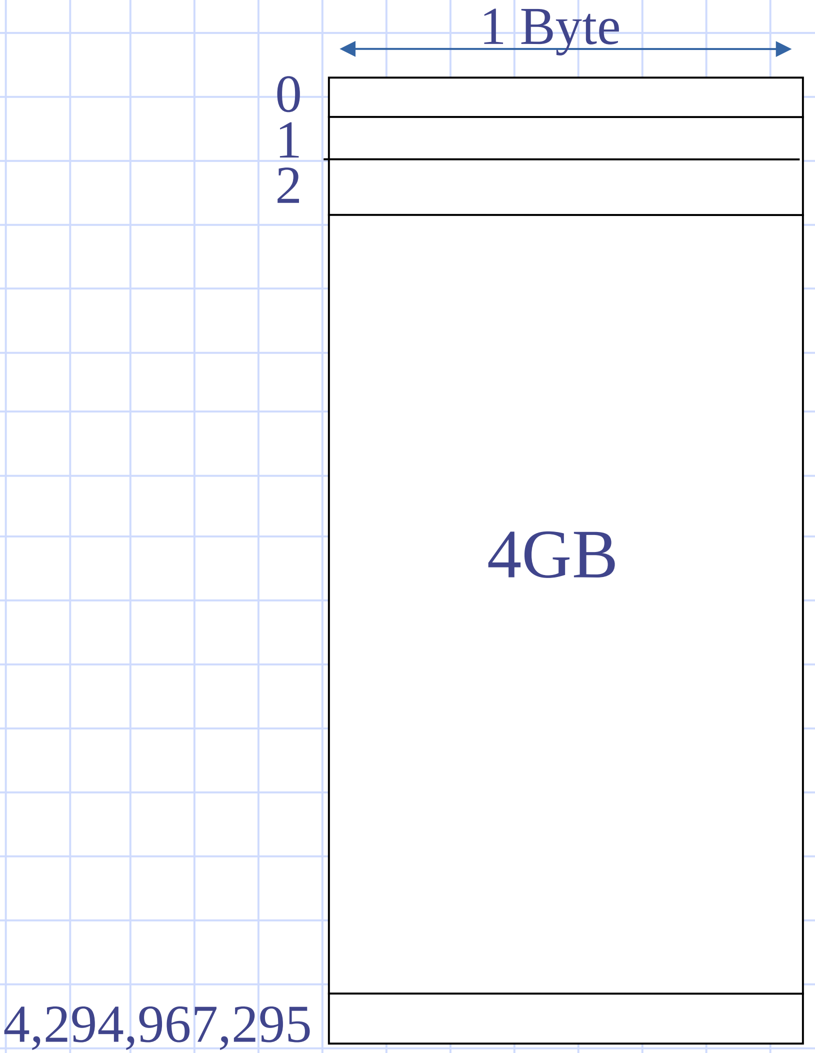
Basic Computer Organization



Main Memory

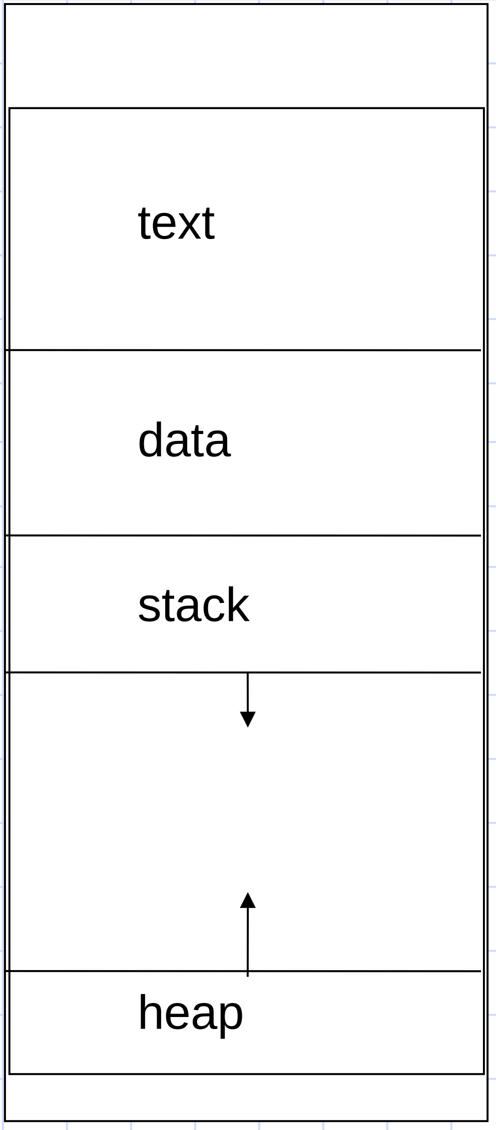
- Holds instructions and data
- We can view it as a sequence of memory locations, each referred to by a unique **memory address**
  - A memory address is an unsigned integer
- If the size of each memory location is 1 Byte, we call the memory **byte addressable**
- This is quite typical, as smallest data (character) is represented in 1 Byte
- Larger data items are stored in contiguous memory locations, e.g., a 4Byte integer would occupy 4 consecutive memory locations

Use of Main Memory by a Program



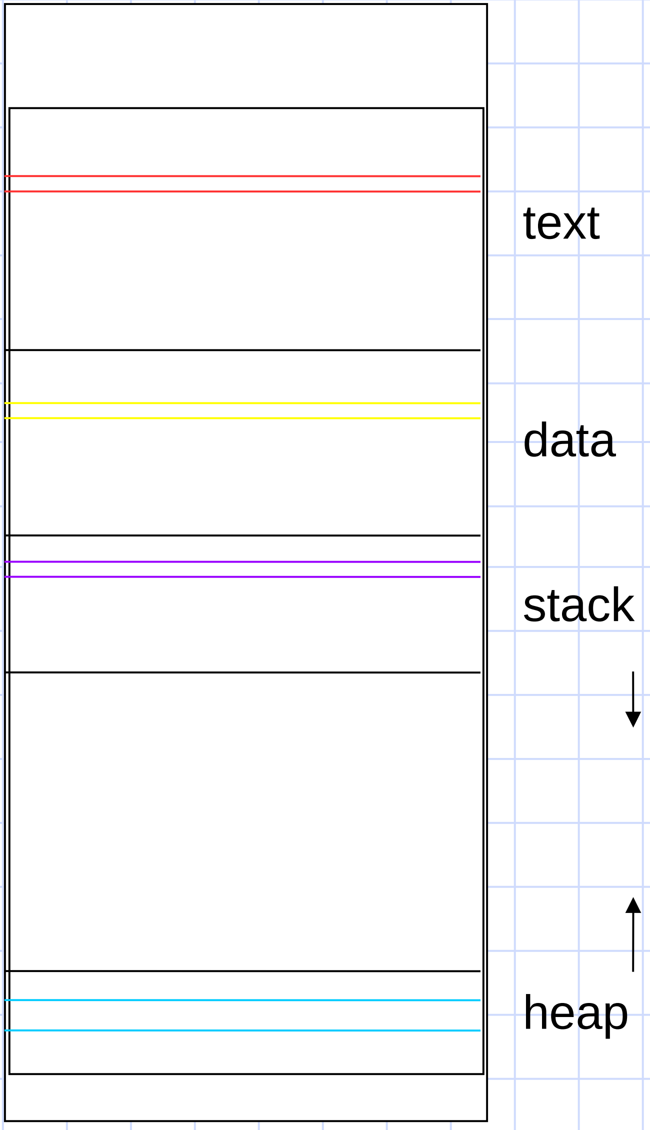
Use of Main Memory by a Program

- Instructions (code, text)
- Data used in different ways
  - Stack allocated
  - Heap allocated
  - Statically allocated



Use of Main Memory by a Program

- add x, y, z
- Data used in different ways
  - Stack allocated: int x
  - Heap allocated: double w
  - Statically allocated: float t



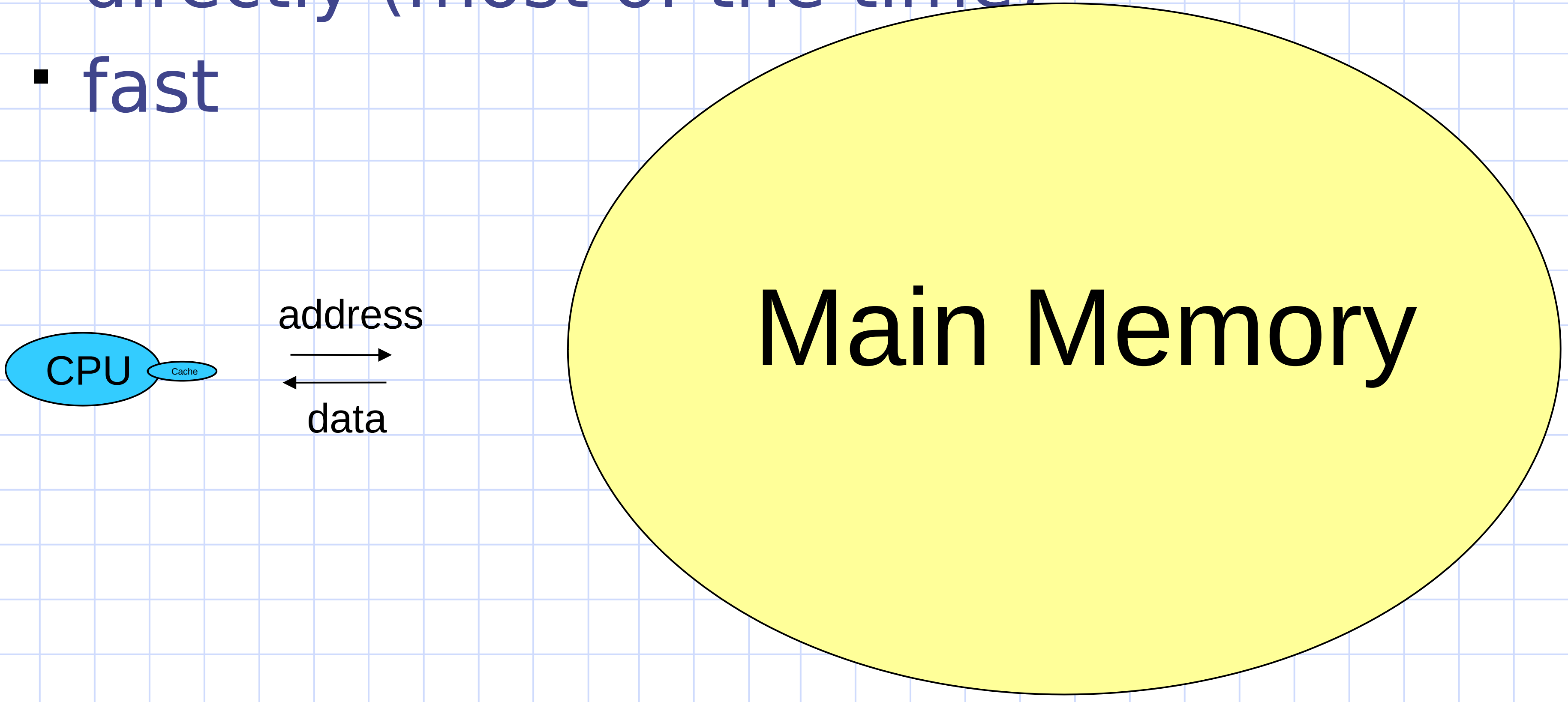
Problem: Slow Speed of Main Memory

- Main Memory is much slower (around 100x) than the CPU and only a few CPU registers
  - CPU will be waiting for data most of the time
- Solution: Cache Memory
  - Fast memory that is part of CPU
  - Design principle: Locality of Reference
  - Temporal locality: least recently accessed memory locations are least likely to be referenced in the near future
  - Spatial locality: neighbours of recently accessed memory locations are most likely to be referenced in the near future

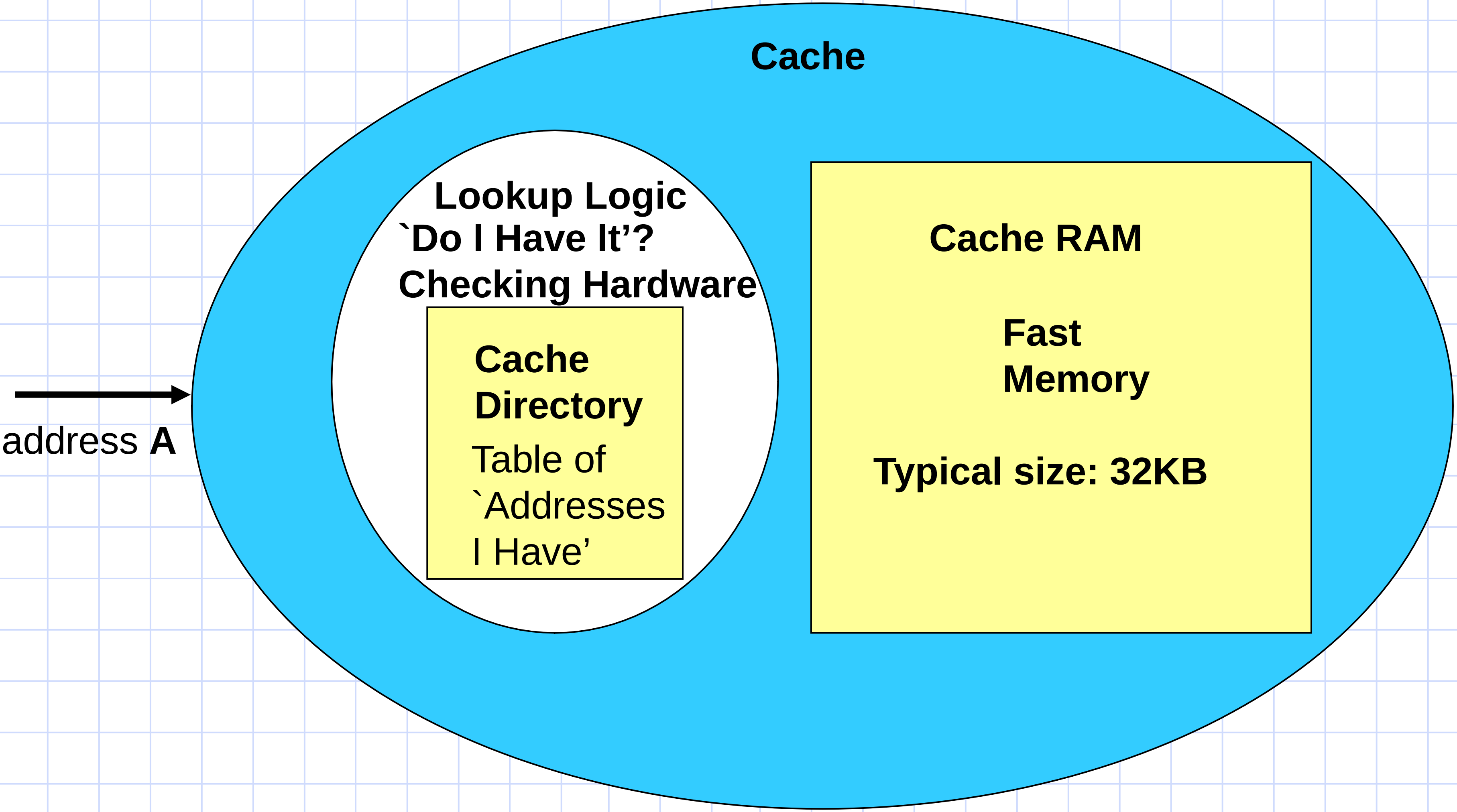
Cache exploits locality

Cache: Hardware structure that provides memory contents the processor wants to access

- directly (most of the time)
- fast



Cache Design



Cache Lookup

- Size of Lookup table (cache directory)
- One entry for each byte in cache?
- Idea: Reduce the size by doing lookup for a chunk of contiguous memory locations
- For example, say chunk size is 32 Bytes

Bytes 0-31, 32-63 … 4294967264-4294967295

- Each such chunk is called a “block”
- For a cache of size 32KBytes, the Cache Directory size is only 1K

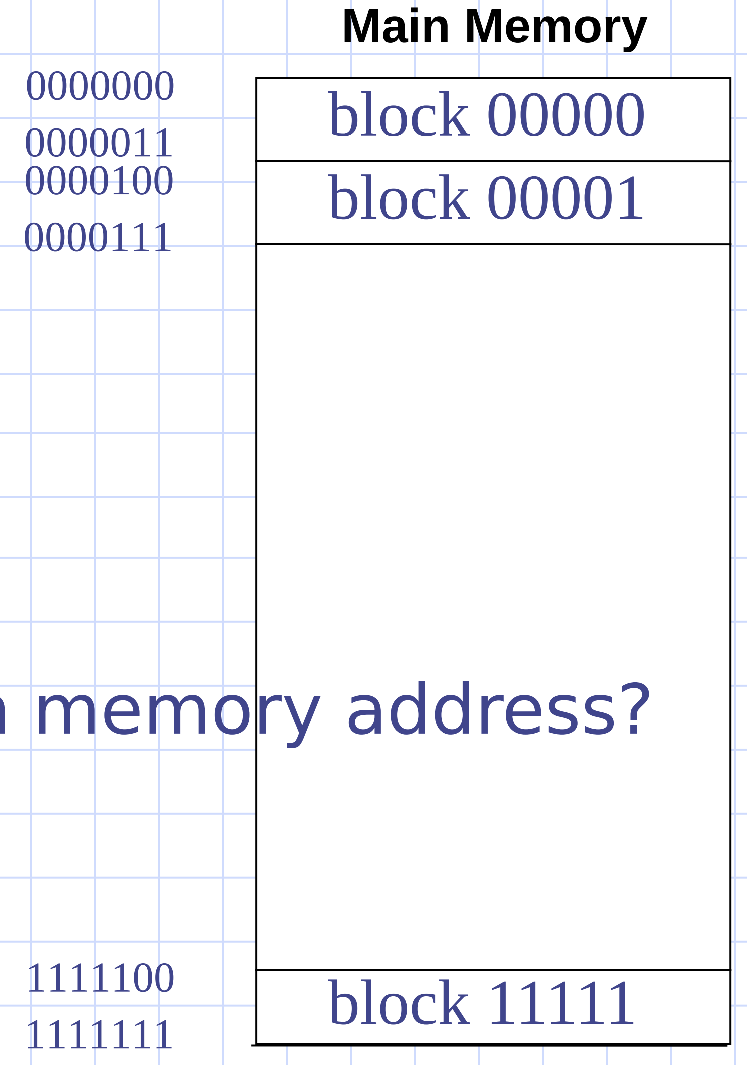
Cache Lookup

- Let’s use a toy example
  - 128 Byte main memory
  - 4 Byte block size
  - 16 Byte cache memory
- Size of a memory address?
  - 7 bits
  - 0000000 - 1111111

Cache Lookup

- Let’s use a toy example
  - 128 Byte main memory
  - 4 Byte block size
  - 16 Byte cache memory
- Size of a memory address?
  - 7 bits
  - 0000000 - 1111111
- Finding block number given memory address?

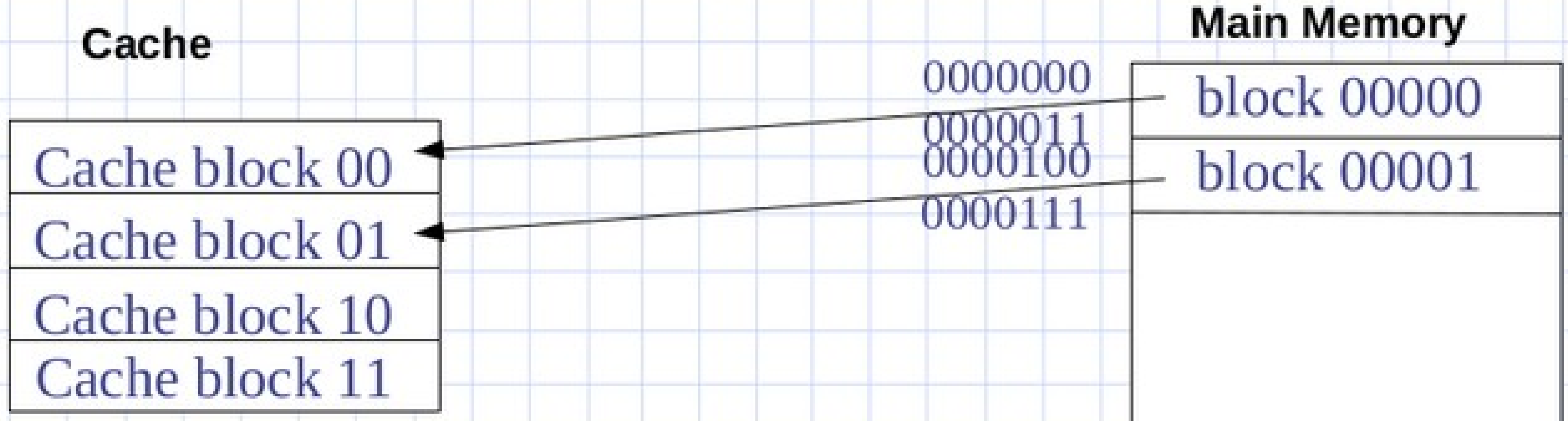
$b_6b_5b_4b_3b_2b_1b_0$

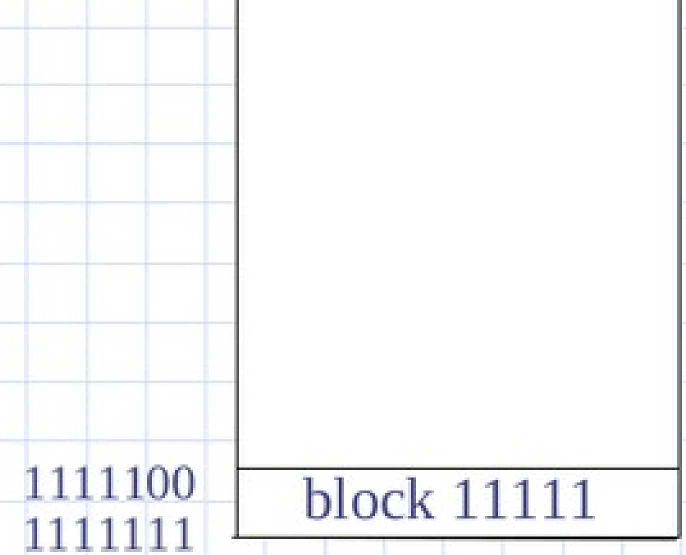


Cache Design

- Recall: Cache size is 16 Bytes
- i.e., it can hold 4 memory blocks
- How to do the lookup?
  - Option 1: Look for the required memory block in all the 4 cache directory entries (“Fully associative” lookup)
  - Option 2: Look in exactly one of the cache directory entries, based on the bits of the memory block number (“Direct mapped” lookup)

Direct Mapped Cache Lookup





- Only one memory block can be in any of these cache blocks at a time
- Main memory blocks that could be in Cache block 00: 000**00**, 001**00**, 010**00**, 011**00**, 100**00**, 101**00**, 110**00**, 111**00**
- Cache Directory is used to keep track of the current occupant of each cache block

Cache

Cache RAM

Cache block 00
Cache block 01
Cache block 10
Cache block 11

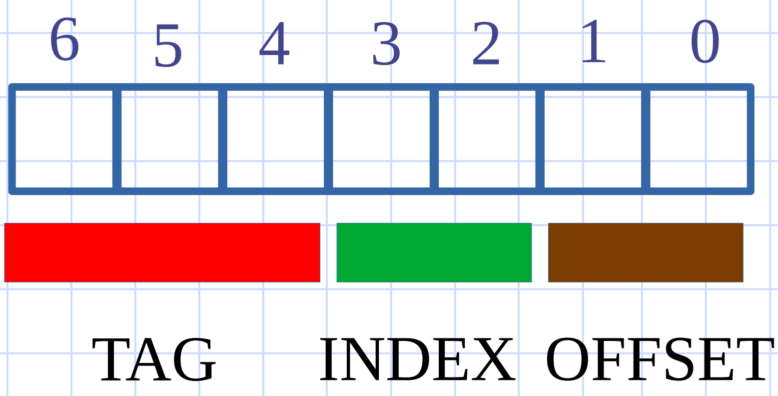
Cache Directory

Tag	V?	...
	0	
000	1	
	0	
111	1	

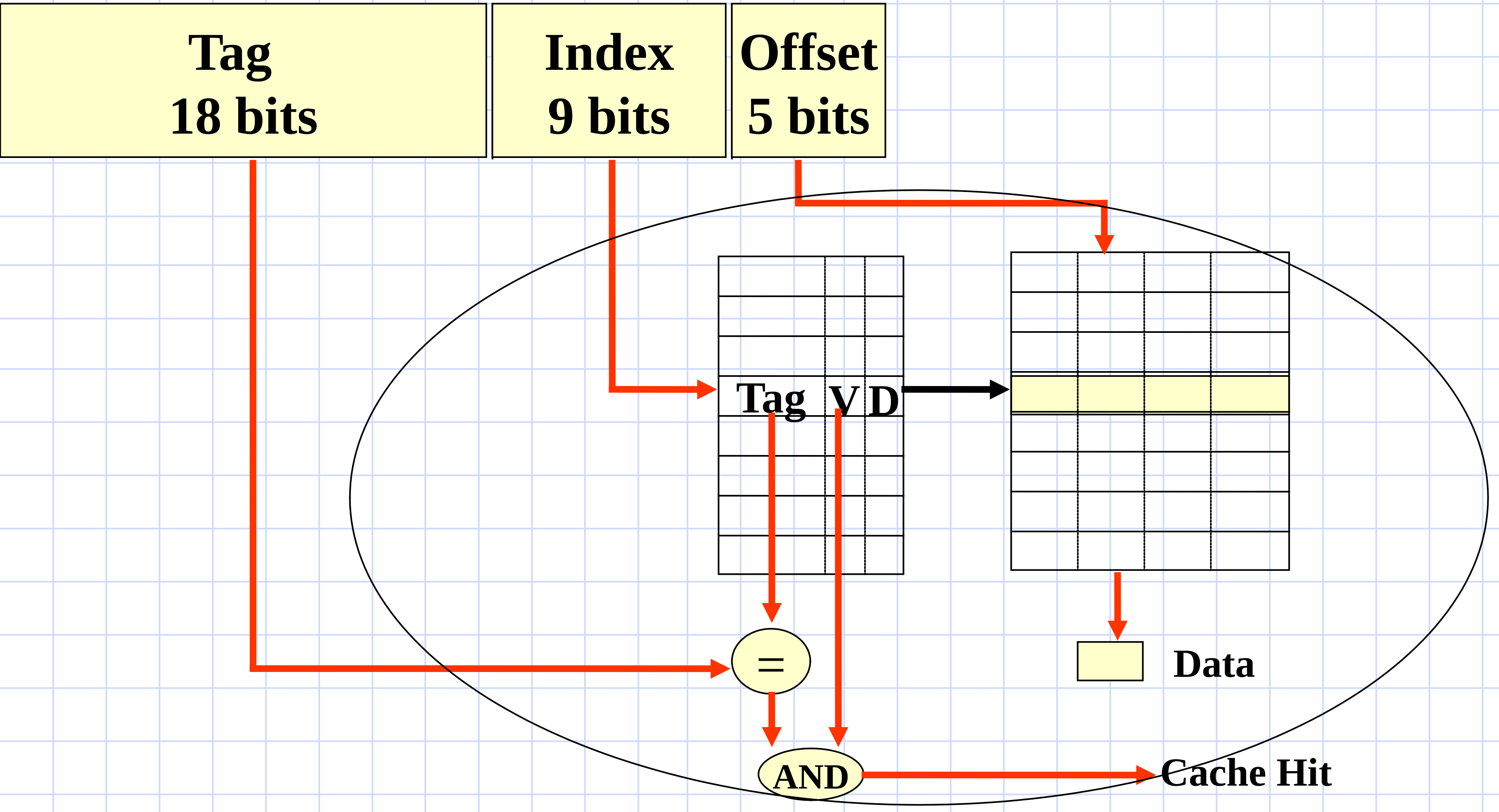
Main Memory

0000000 – 0000011	block 00000
0000100 – 0000111	block 00001
1111100 – 1111111	block 11111

Bits of 7 bit Memory Address (from cache perspective)



Cache Lookup and Access



Direct Mapped Cache Lookup