Basic Memory Structure

- **m address lines**
- **n data bits**
- **Stores** $2^m$ n-bit data words
- **Memory Chip**
- **2^m x n**

- **Address Decoder**
- **2^m word lines**
- **n data bit lines**
Memory Types

**RAM -- Random Access Memory (Static RAM)**
- readable and writeable
- volatile -- loses data if the power is turned off
- static -- doesn’t lose data if the power is on
- fast read and write
- not as dense or cheap as DRAM

**DRAM -- Dynamic RAM**
- capacitor as memory storage element
- volatile
- dynamic -- can lose memory even while power is on so it needs refresh
- denser, cheaper per bit than SRAM
- destructive readout
- slower reads and writes

FPM -- Fast Page mode, one address multiple reads at subsequent addresses
EDO -- Extended Data Out, similar to FPM
SDRAM -- synchronous DRAM. Uses a clock to speed up the access
DDR DRAM -- dual data rate DRAM. Supplies new data on both clock edges
DDR2 and DDR3 -- deeper burst depth in prefetch buffer, DDR4 -- lower voltage
Memory Types

**ROM** -- Read Only Memory
- non-volatile
- relatively slow

ROM -- mask programmed
PROM -- Programmable ROM, programmed by burning fuses with a programmer
EPROM -- Eraseable PROM, programmed with a programmer, erased by intense UV light
EEPROM -- Electrically Eraseable PROM, erased one byte at a time, longer read than write cycle
FLASH -- Electrically Eraseable PROM, erased a block at a time, faster than EEPROM
NVRAM -- (future) speed of RAM, non-volatile
We would like to have a memory system that costs as much (per bit) as the cheapest memory and that has an access time approaching that of the fastest memory.

This is done by having a small amount of fast and expensive memory followed by larger amounts of progressively slower but cheaper memory.
We would like to have a memory system that costs as much (per bit) as the cheapest memory and that has an access time approaching that of the fastest memory. This is done by having a small amount of fast and expensive memory followed by larger amounts of progressively slower but cheaper memory.
The most frequently used data and instructions are moved closer to the processor

**Locality of Reference**

**Temporal Locality:** Once an item has been accessed, it is highly likely to be accessed again soon.

**Spatial Locality:** Once an item has been accessed, items near it are highly likely to be accessed soon.
Locality of Reference

Temporal Locality: Once an item has been accessed, it is highly likely to be accessed again soon.

Spatial Locality: Once an item has been accessed, items near it are highly likely to be accessed soon.

Sequential Locality: Instructions tend to be accessed sequentially. (Really just Spatial Locality)