An introduction to heterogenous computing

Accelerating numerical codes

Krishna Kumar *[1]

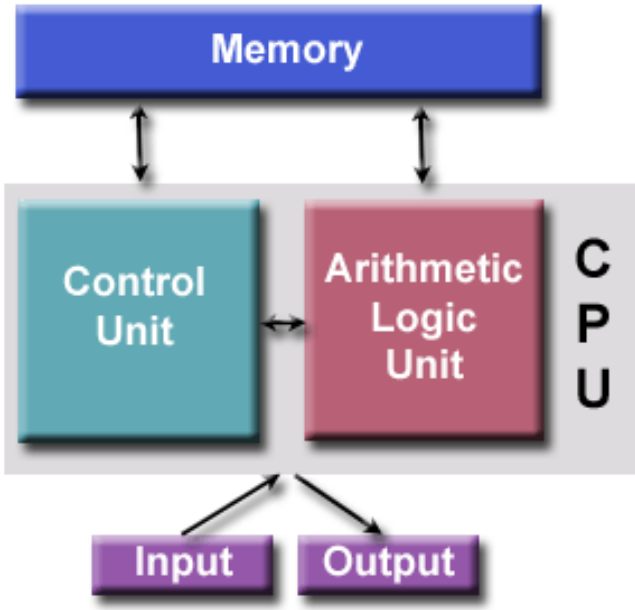
August 10, 2015

[1] github.com/kks32

Outline

von Neumann Architecture

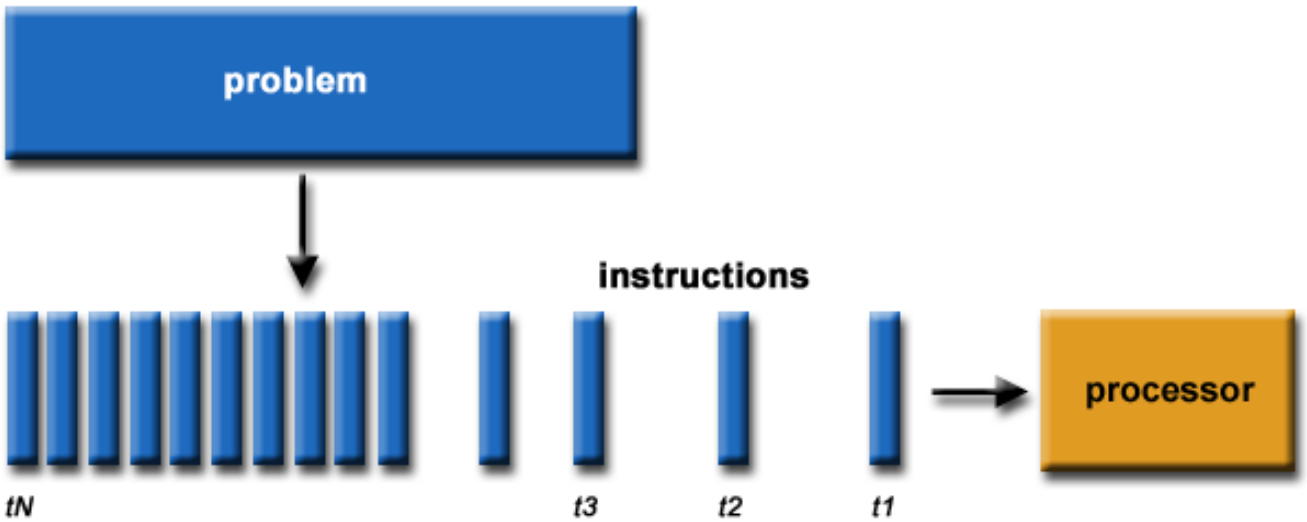
- Hungarian mathematician John von Neumann circa 1940 - the general requirements for an electronic computer.
- "Stored-program computer" - both program instructions and data are kept in electronic memory.
  - *Read/write*, random access memory is used to store both program instructions and data.
  - *Control unit* fetches instructions/data from memory, decodes the instructions and then sequentially coordinates operations to accomplish the programmed task.
  - *Aritmetic Unit* performs basic arithmetic operations
  - *Input/Output* is the interface to the human operator



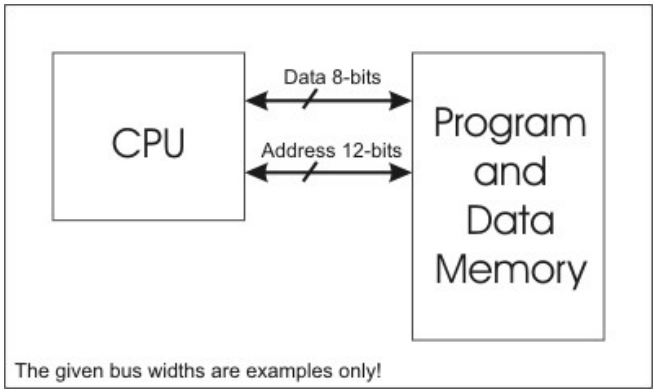
Basic architecture

Serial computing

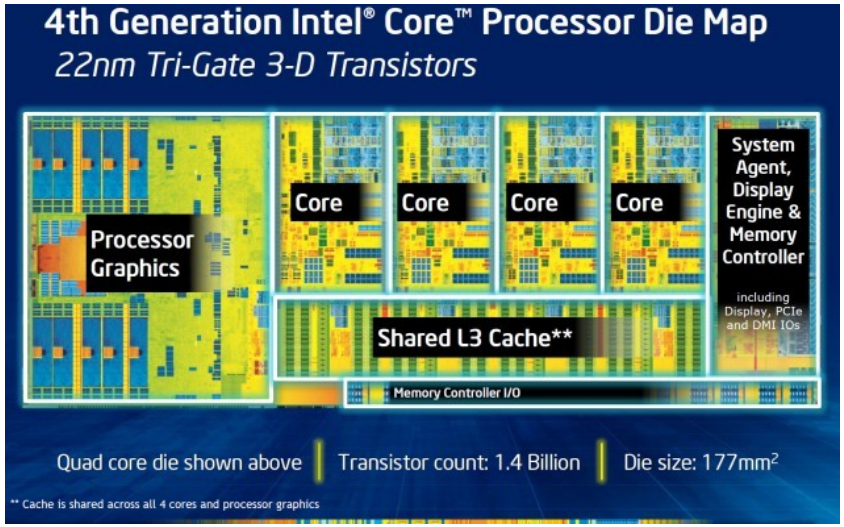
- Traditionally, software has been written for serial computation:
  - A problem is broken into a discrete series of instructions
  - Instructions are executed sequentially one after another
  - Executed on a single processor
  - Only one instruction may execute at any moment in time



Memory model



Ideal memory model:
We write for this architecture



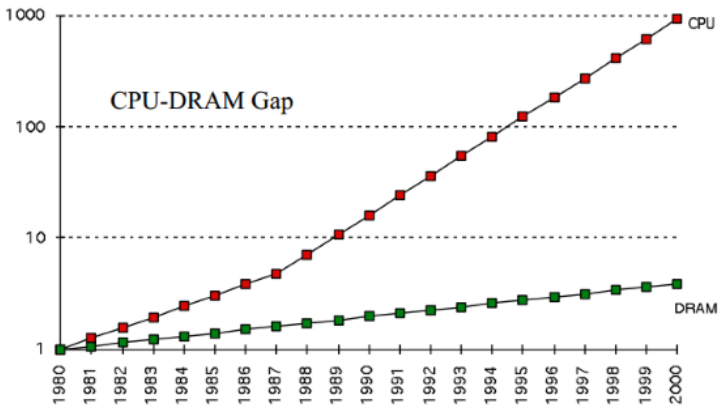
Real memory model: How it actually looks

The underlying assumption is cache coherency!

In a shared memory multiprocessor with a separate cache memory for each processor, it is possible to have many copies of any one instruction operand: one copy in the main memory and one in each cache memory. When one copy of an operand is changed, the other copies of the operand must be changed also. Cache coherency ensures that changes in the values of shared operands are propagated throughout the system in a timely fashion.

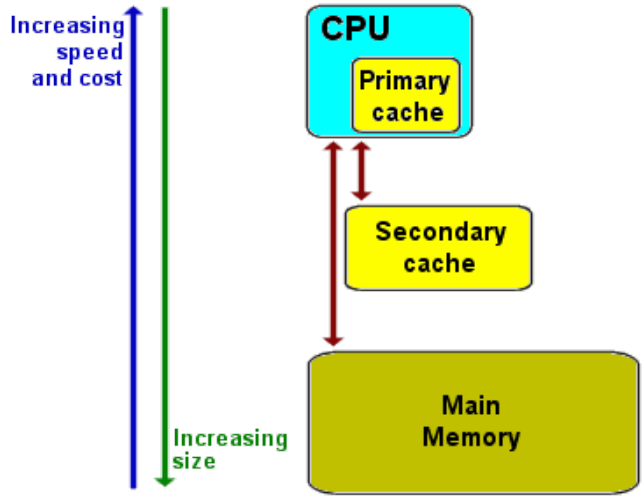
What are caches

- Processor vs Memory Performance



1980: no cache in microprocessor;
1995 2-level cache

CPU vs DRAM



Cache

- CPU caches are small pools of memory that store information the CPU is most likely to need next.
- A cache miss means the CPU has to go scampering off to find the data elsewhere. This is where the L2 cache comes into play while it's slower, it's also much larger.
- If data can't be found in the L2 cache, the CPU continues down the chain to L3 (typically still on-die), then L4 (if it exists) and main memory (DRAM).