

---

# Computer Science 141

## Computing Hardware

Fall 2009

Harvard University

Instructor: Prof. David Brooks

[dbrooks@eecs.harvard.edu](mailto:dbrooks@eecs.harvard.edu)

# Announcements

---

- This Wednesday – Guest lecture from ARM
- Today's Lecture:
  - Intro to Computer Architecture
  - Busses and interfaces between CPUs, memories, I/O

# Computer Architecture

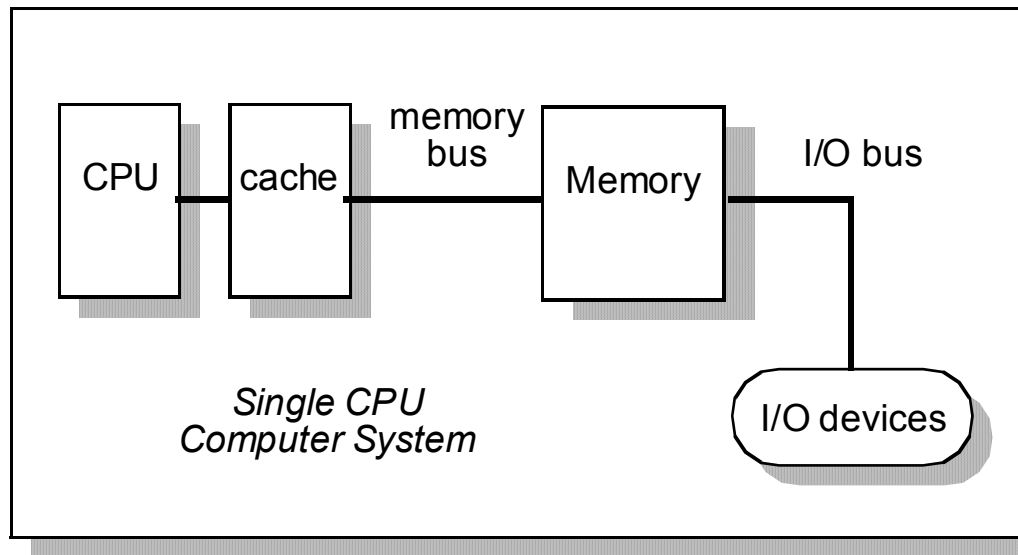
---

- Change focus
  - from logic design (gate-level design)
  - to computer architecture (HW/SW interface of computing systems)
- Definitions
  - architecture = interface between hardware and software in a computer system
  - Implementation (or organization) = specific hardware design that supports interface described by architecture

# Computer Architecture

---

- Topics
  - measures of performance
  - instruction sets
  - CPU, memory, and I/O design



*Emphasis on hardware/software interactions*

# Computer Architecture

---

- Computers built from logic gates
  - from 0's and 1's
  - to code and data for sophisticated applications
- Key principles of today's computer
  1. multiple interpretations for a bit string:

0000 0001 0000 1001 0011 0000 0010 0000

two's complement  $\Rightarrow$  17 379 36010

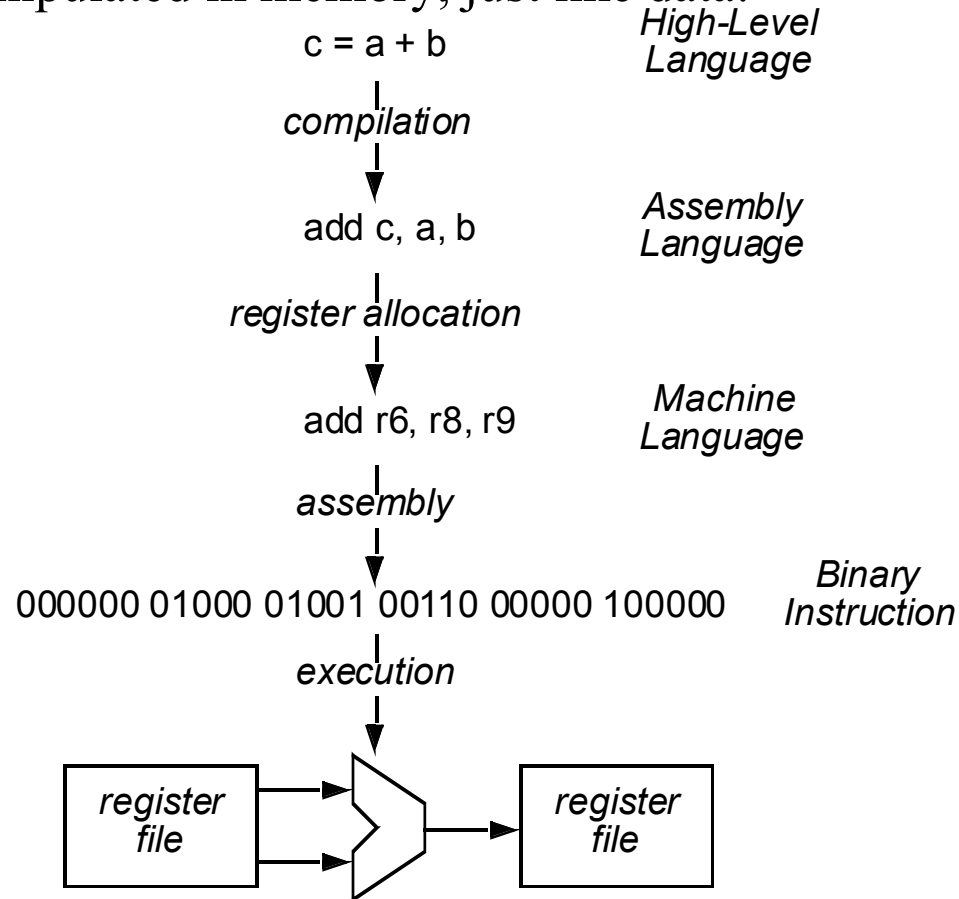
single precision FP  $\Rightarrow$   $1.071781159 \times 10^{-125}$

instruction  $\Rightarrow$  add r6, r8, r9

# Key principles of today's computer (cont.)

---

2. programs manipulated in memory, just like data:

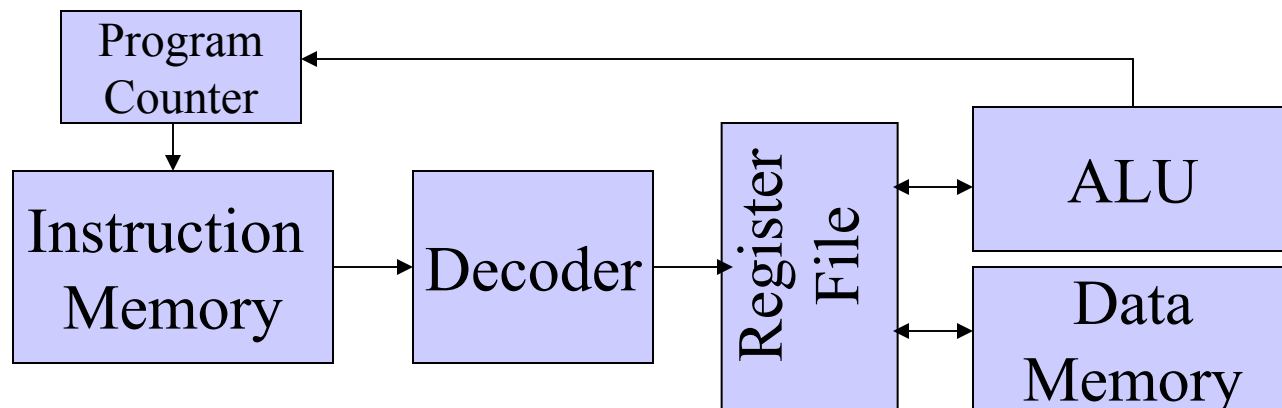


- Foundation of stored-program concept and all mainstream computing!
-

# CPU Design in 1 Slide

---

- Fetch program instructions from memory in sequential order
- Decode instructions into several categories
  - Arithmetic/Logic Instructions
    - Read Two Register Values, Perform Operation, Store One Result
  - Memory Instructions
    - Load and Store Data into Memory (or on-chip caches)
  - Control Flow Instructions
    - Change the flow of instructions based on a comparison

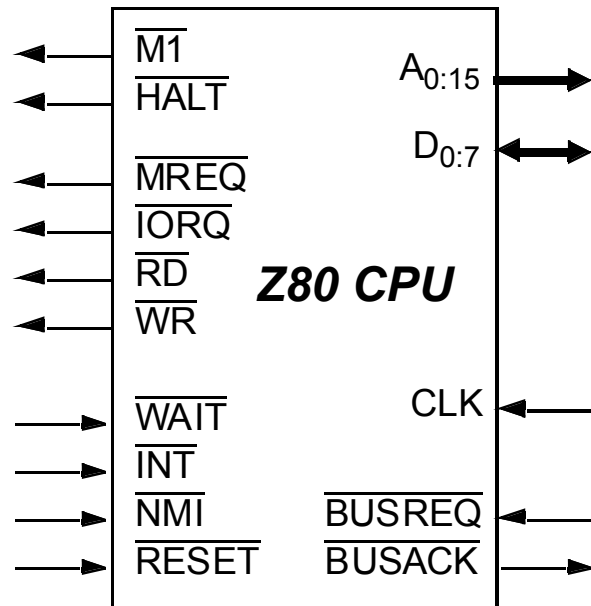


# Top-down decomposition

---

- Understand how CPU interfaces to memory and I/O subsystems
  - I/O subsystem as black box too
  - focus on interface protocols and implementation

# Case Study: Zilog Z80



- 16-bit address bus and 8-bit data bus
  - bus  $\equiv$  collection of 2 or more related signals
- memory ( $\overline{MREQ}$ ) vs. I/O ( $\overline{IORQ}$ ) control line, read vs. write
- CPU status lines —  $\overline{M1}$  and  $\overline{HALT}$ 
  - $\overline{M1}$  together with  $\overline{MREQ}$  indicates opcode fetch cycle
  - $\overline{M1}$  together with  $\overline{IORQ}$  indicates interrupt ack cycle

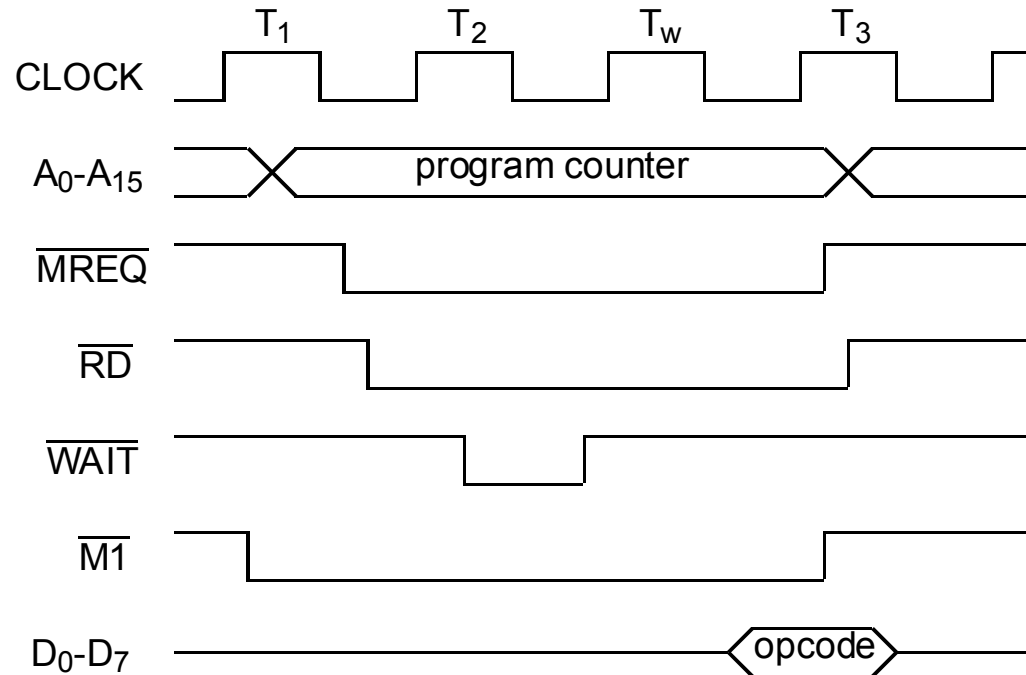
# Z80 CPU timing

---

- instructions execution follows a specific sequence of operations
- instruction fetch — operand fetch — execution — write result
- possible external operations:
  - memory read or write
  - I/O device read or write
  - interrupt acknowledge
- T cycle  $\equiv$  basic clock period for Z80
- 3 or more T cycles = machine cycle (e.g. M1 or M2 or etc.)
- M cycles can be extended either by
  - CPU automatically inserting 1 or more Wait states
  - user inserting 1 or more Wait states

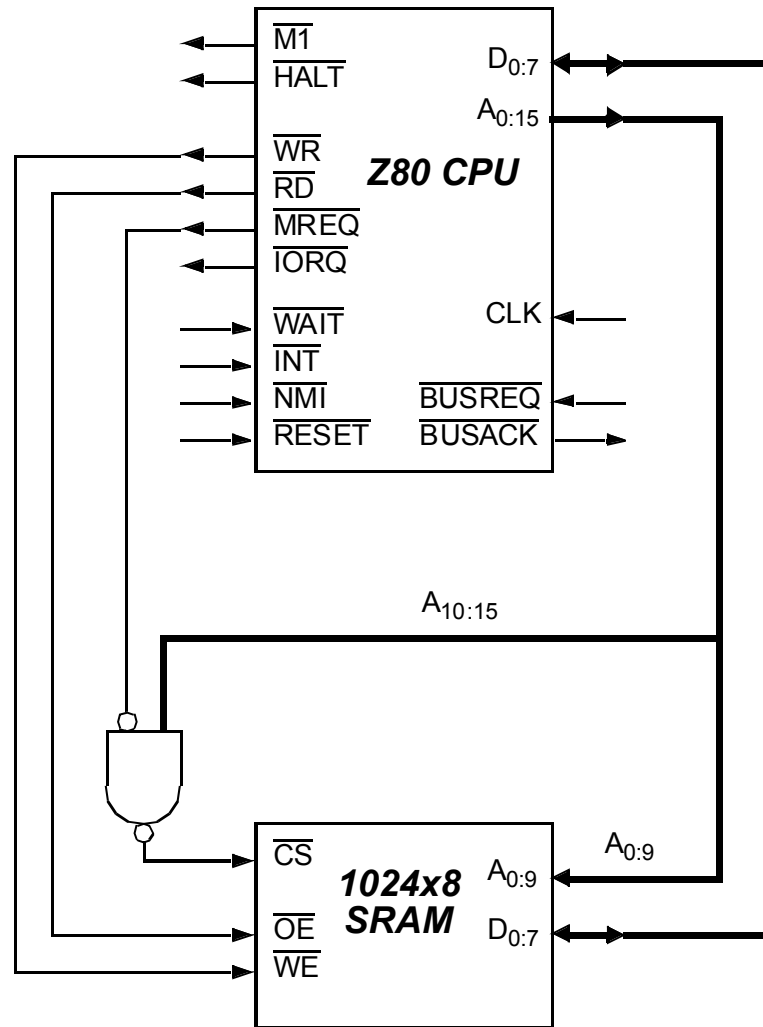
# Z80 CPU timing

- external actions on instruction opcode fetch



- data latched on rising edge of  $T_3$
- $\overline{\text{WAIT}}$  sampled on falling edge of  $T_2$

# Connecting the Z80 to a 1024x8 SRAM



- next step in design process — does my design meet all of the timing requirements?

# Buses

---

- Bus  $\equiv$  one set of wires used as shared communication link between multiple subsystems of computer
  - + versatility - easy to add/swap peripherals
  - + low cost - standard interface to amortize interface design cost
  - bottleneck - possible limitation on maximum I/O throughput

# 3 types of buses

- Processor-memory bus

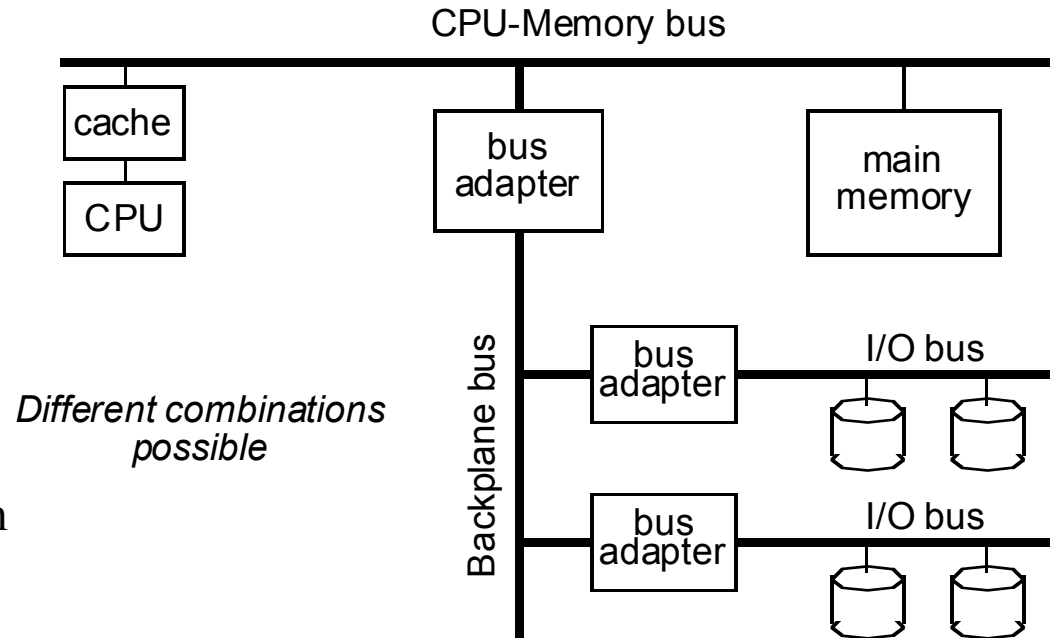
- short, few devices  $\Rightarrow$  high-speed
- specialized to meet needs of processor and memory

- Backplane bus

- middle-ground between other two types
- usually standardized, may be open or not

- I/O bus

- long, many devices  $\Rightarrow$  low-speed
- generalized to fit any device needs



# Performance issues in bus design

---

- bus speed limited by physical factors
  - length of bus — longer means slower
  - no. of device connections supported — more means slower
- bus speed limited by range of devices
  - some devices require low latency
  - some devices require high bandwidth
- General structure of a (communication) bus
  - set of data lines
    - usually a binary power of 2 in size (e.g. 1 or 8 or 32 bits wide)
    - extra lines for parity, etc.
  - set of control lines
    - overhead

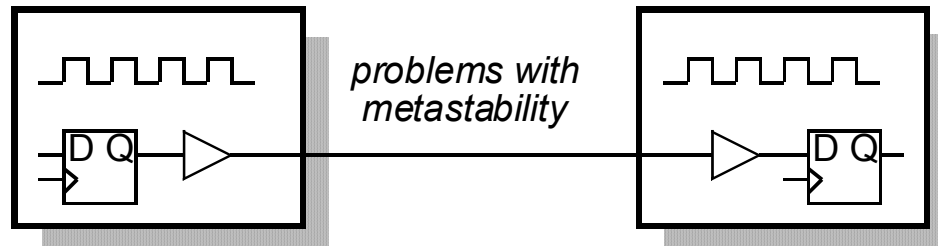
# Bus transaction

---

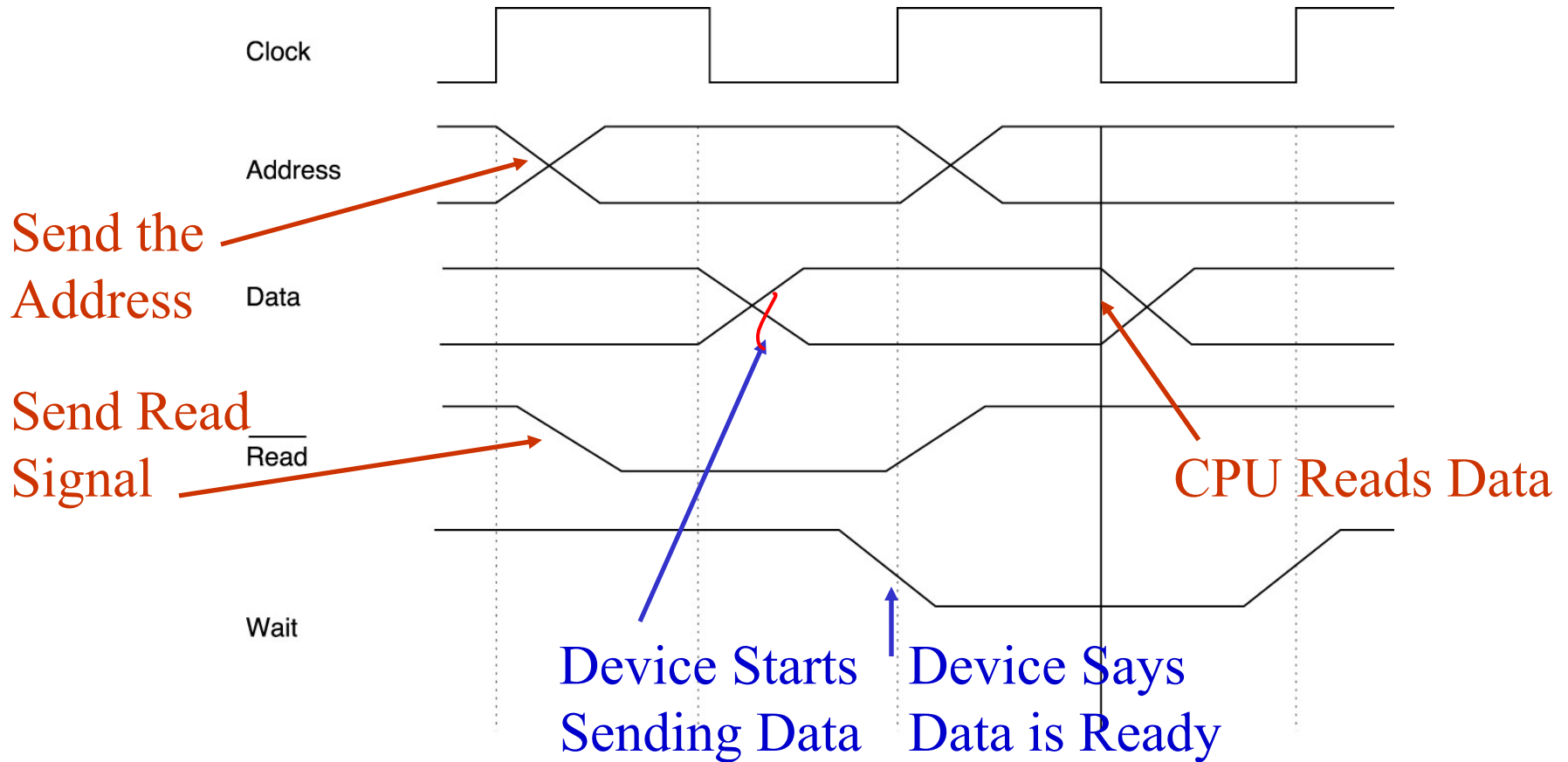
- 2 parts
  - send address
  - send/receive data
- 2 types
  - read transaction - from “mem” to CPU
  - write transaction - to “mem” from CPU

# Timing schemes

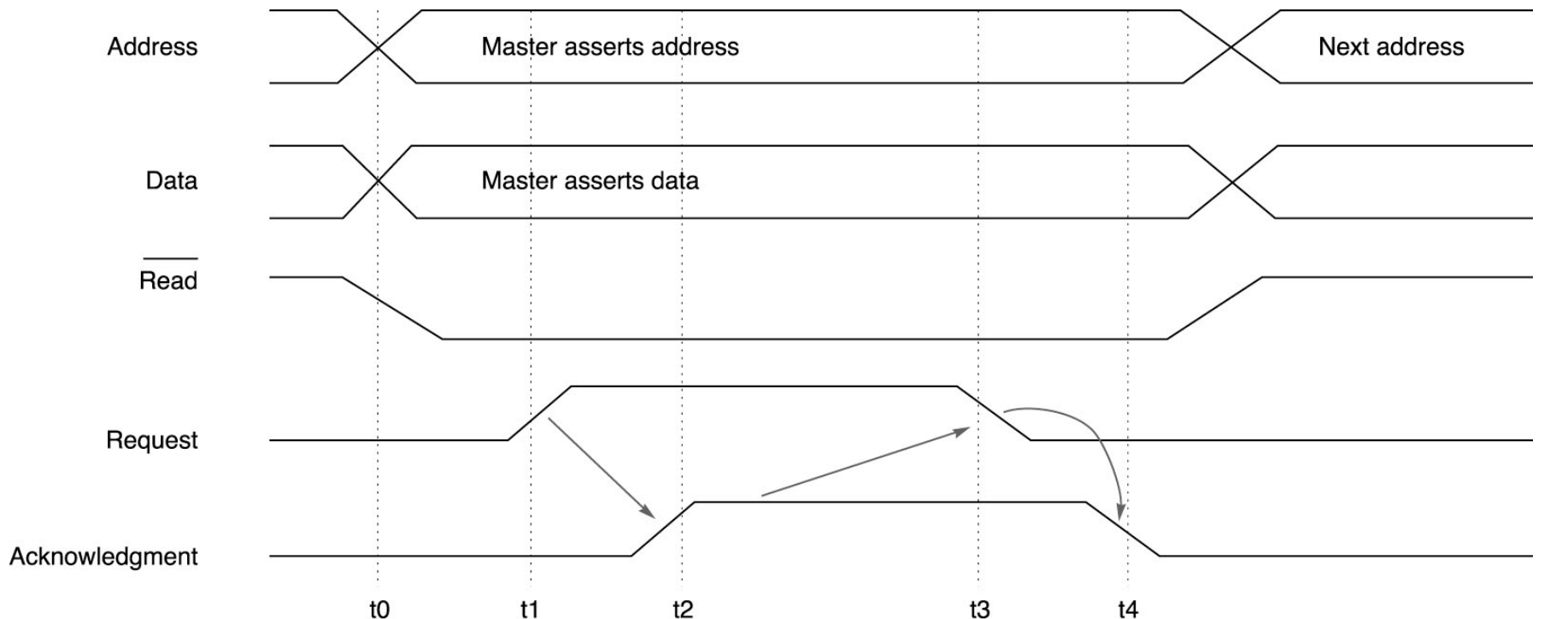
- Synchronous bus (e.g. Intel PCI bus)
  - clock included in control lines
  - implement fixed protocol relative to clock signal
  - + straightforward implementation (FSM)
  - + bus runs fast (implicit synchronization)
  - every device must run at same clock rate
  - problems with clock skew (bus must be short (or slow))
- Asynchronous bus (e.g. SCSI bus)
  - no clock, use handshake protocol (request – reply – ack)
  - advantages over synchronous bus
    - + easier to accommodate large variety of I/O devices
    - + easier to scale bus (long bus ok)
  - disadvantages over synchronous bus
    - need to synchronize exchange of info (need synchronizers)



# Synchronous Data Transfer: Read Operation



# Asynchronous Data Transfer: Write Operation



t0: Master asserts lines  
t1: Master waits and asserts Req

t2: Device asserts Ack (data recvd)  
t3: Master releases Req (Handshake)  
t4: Device releases Ack

# Summary

---

- CPU-memory bus almost always synchronous
- Backplane bus sometimes synchronous, sometimes asynchronous
- I/O bus never synchronous, always asynchronous

# Bus Bandwidth

---

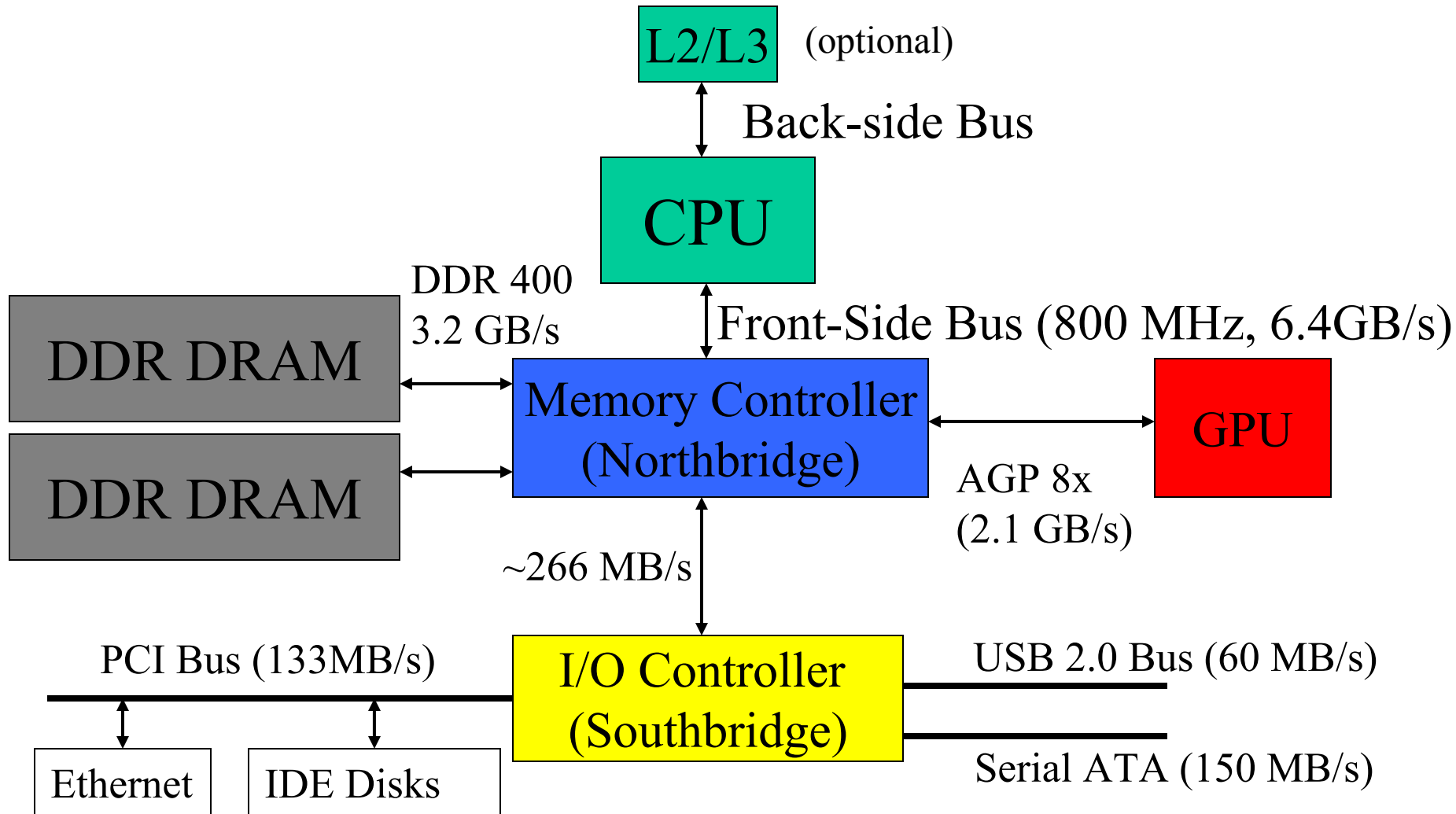
- Limited by
  - steps in transaction — e.g. send address, wait for data, return data
  - characteristics of bus structure, e.g.
    - serial or parallel transmission of data
    - pipelined transactions, etc.

# 4 bus characteristics that affect bus BW

---

1. width of data bus
  - + doubling width of bus means double the BW
  - more expensive bus lines
2. Separate vs. multiplexed address and data lines
  - + write transaction takes 1 less cycle
  - + read transactions can be pipelined
  - more expensive bus lines
3. block transfers
  - + one address reads multiple words
  - more complexity in bus control
  - possibly more latency due to longer wait for bus access
4. split transactions (release bus between request and reply)
  - + minimize idle time on bus
  - longer latency due to need to acquire bus twice per operation
  - more complexity in bus control

# Integrated System Diagram



# FSB Speeds/Bandwidth

Processor Class	FSB Frequency	Theoretical Bandwidth
Pentium II	66/100 MHz	533/800 MB/s
Pentium III	100/133 MHz	800/1066 MB/s
Pentium 4	100/133/200/266 MHz (4x)	3200/4266/6400/8533 MB/s
Core2Duo	133/200/266/333 MHz (4x)	4266/6400/8533/10667 MB/s
Pentium M	100/133/166 MHz (4x)	3200/4266/5333 MB/s
Athlon	100/133 MHz (2x)	1600/2133 MB/s
Athlon XP	133/166/200 MHz (2x)	2133/2666/3200 MB/s
Athlon 64/Opteron	600/800/1000 (HT) MHz	7500/12800/14400 MB/s
PowerPC970	900/1000/1250 MHz	7200/8000/10000 MB/s

Typical PC-class FSBs operate at 64-bits/cycle

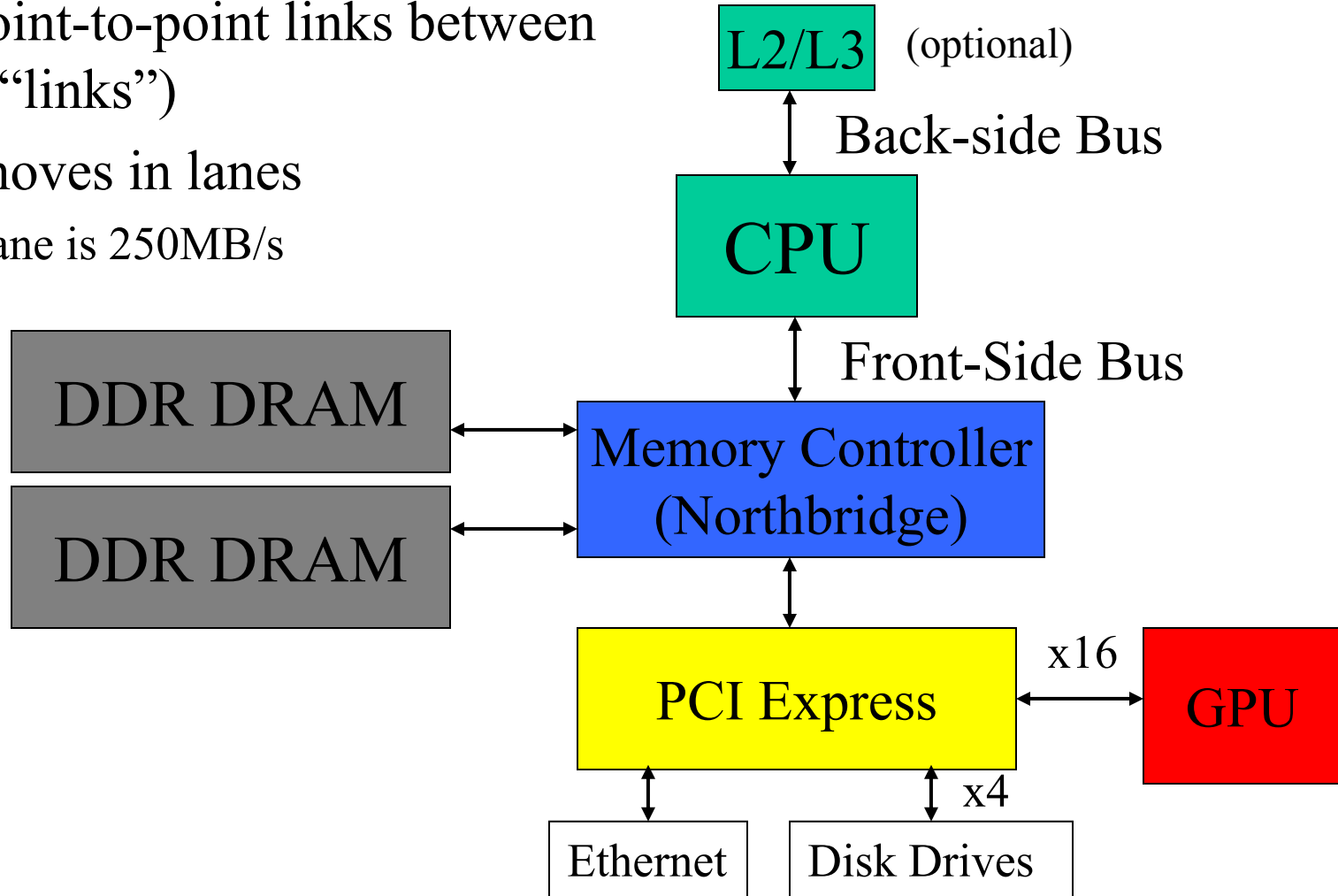
# PCI Bus

---

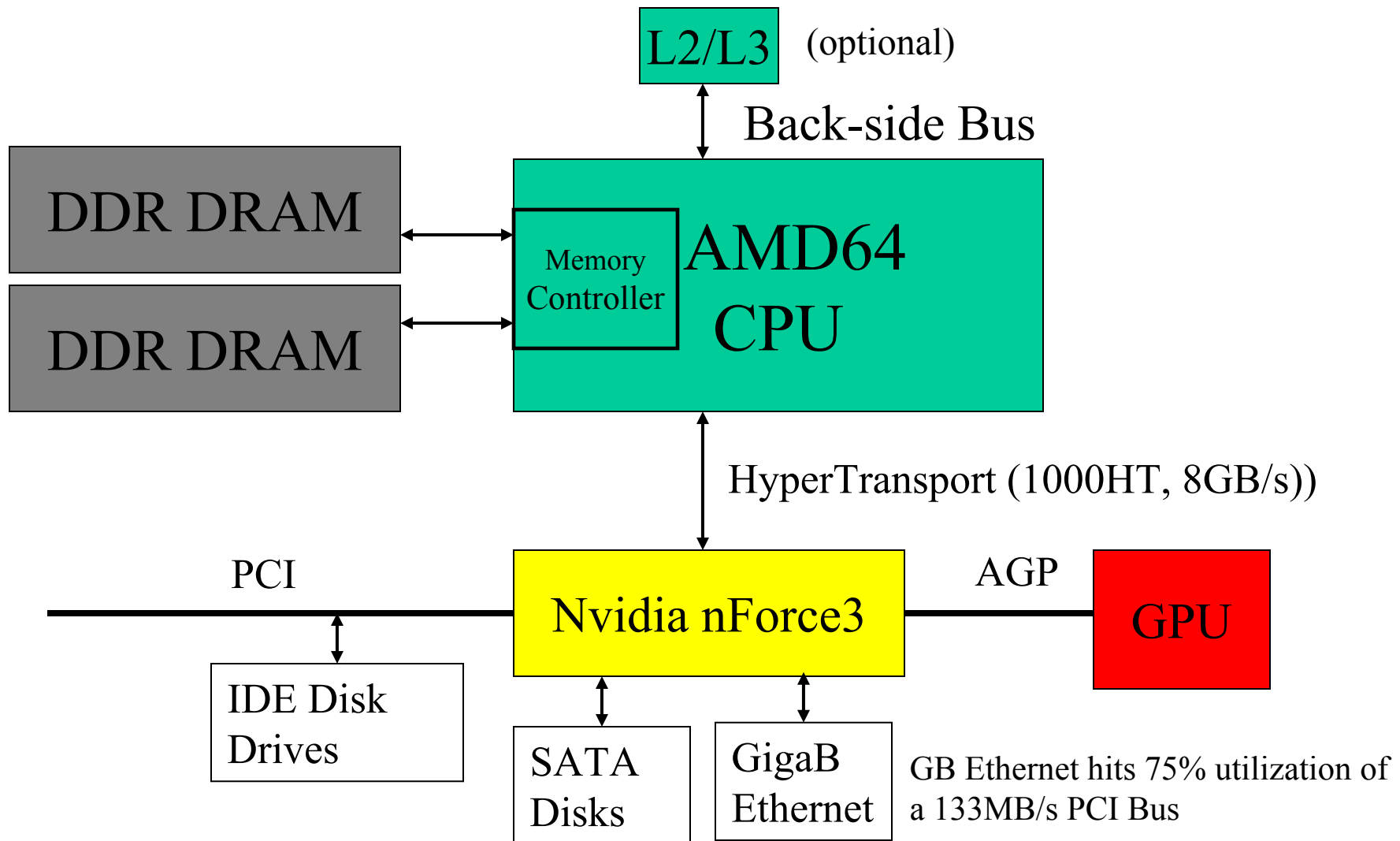
- PCI 2.0 first major standard (replaced ISA, etc)
  - 33MHz clock
  - 32-bit (133MB/s) or 64-bit (266MB/s) bus widths
- PCI 2.2 allows 66MHz signals, up to 533 MB/s
- PCI-Express provides good scalability:
  - X1: 250MB/s (doubled in PCI-E 2.0)
  - X16: 4000MB/s
- Coming soon – PCI Express 3.0, will double bandwidth

# PCI Express

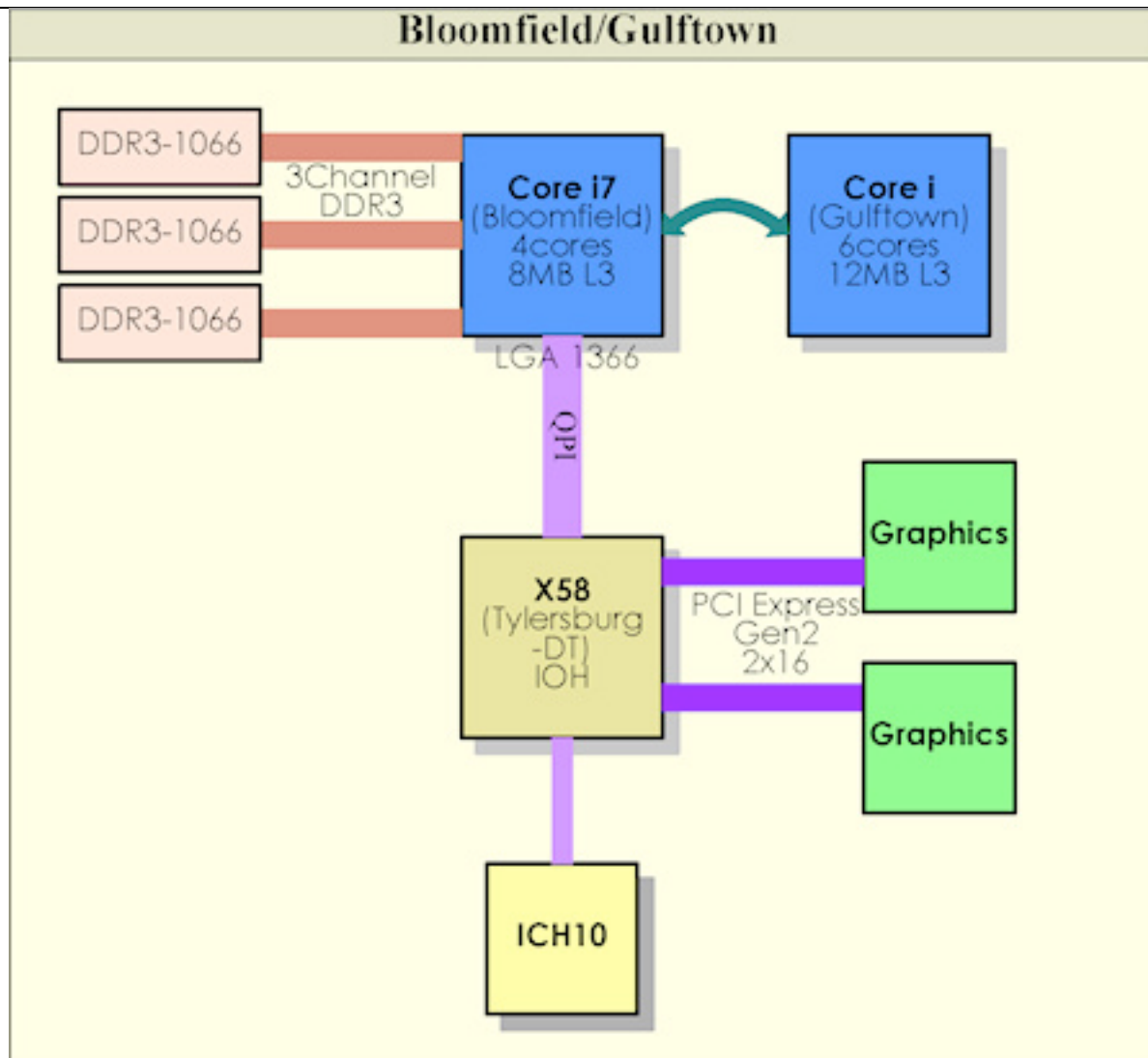
- Serial, point-to-point links between devices (“links”)
- Traffic moves in lanes
  - Each lane is 250MB/s



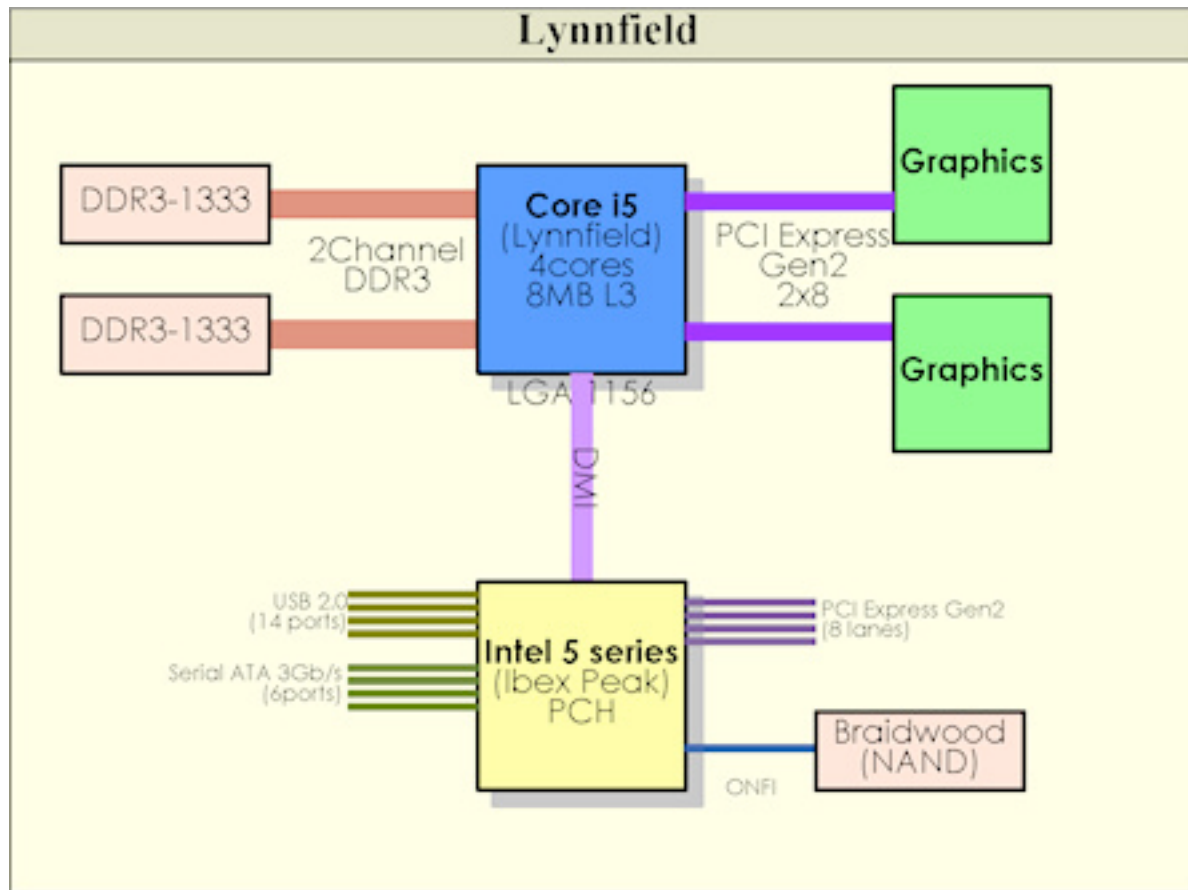
# Recent Trends: Integration



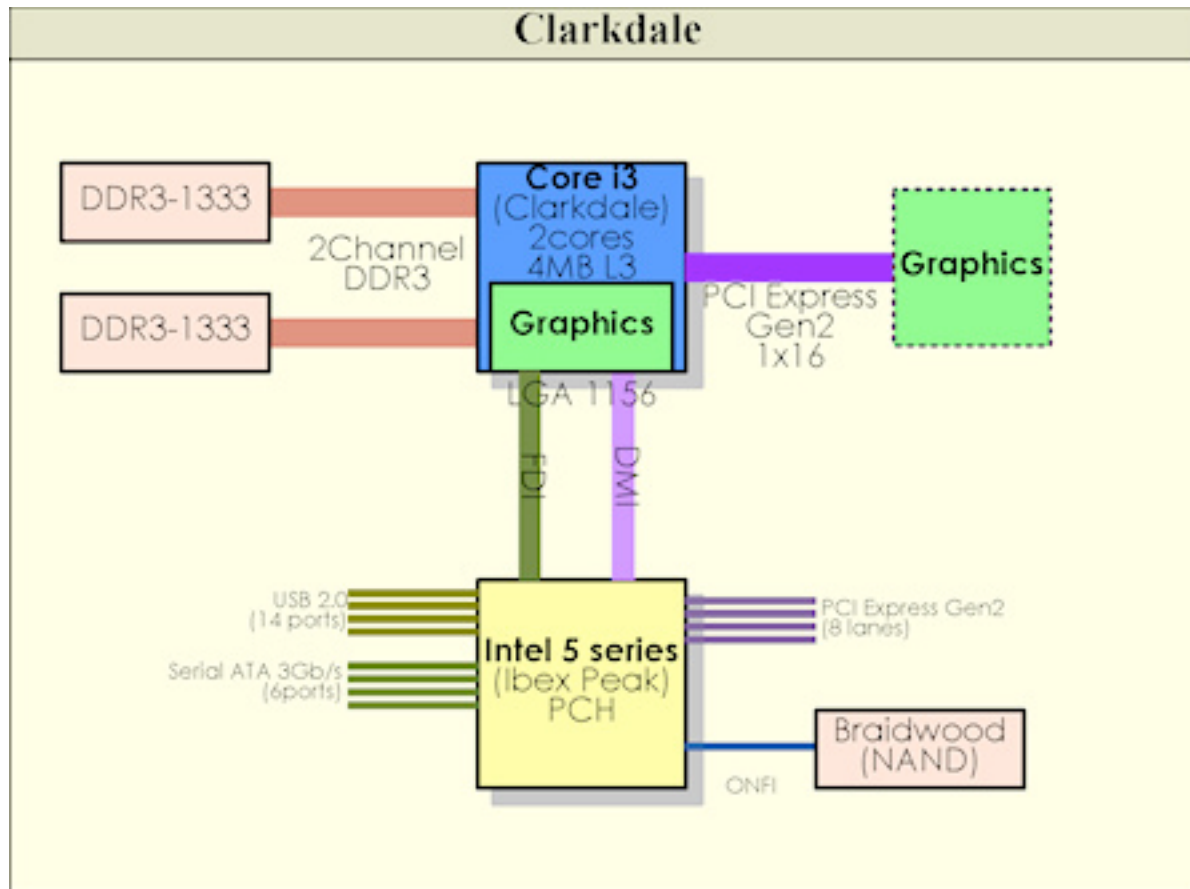
# Nehalem Platform Architectures



# “High-end”



# “Mainstream”



# Looking Ahead...

---

- ARM lecture on Wednesday
- Next week – MIPS ISA and Datapath
  - Will form the basis of Lab 8