

198:211

Computer Architecture

- Topics:
 - System I/O
 - Buses

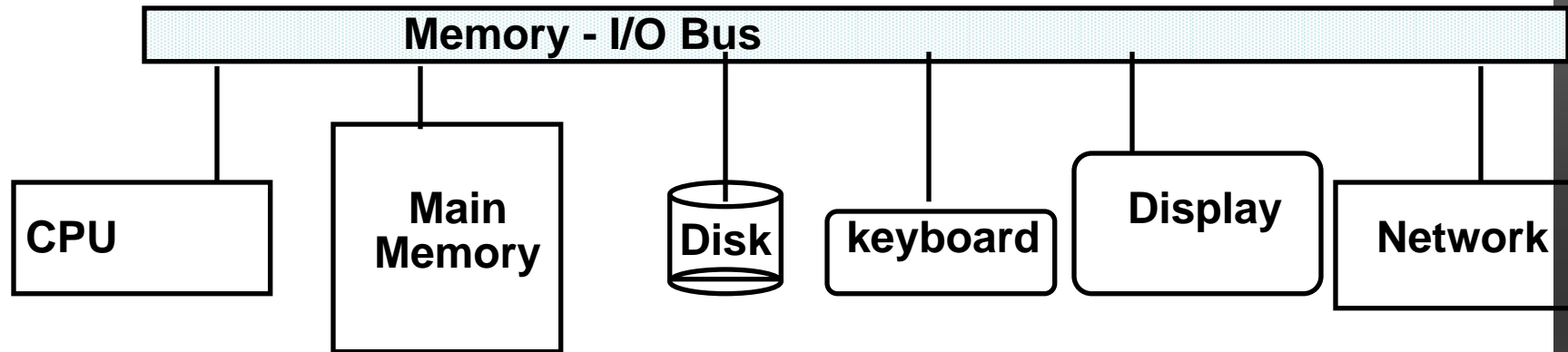
I/O or input and output

- In addition to memory, data transfer needs to occur between CPU and Input output devices
- When reading from memory, a byte or several bytes can be transferred from memory to register using
 - `mov address, %eax` or `mov %eax, address`
- I/O devices also are sources or destinations for bytes of data
- I/O devices can be viewed just as memory
- I/O devices can be viewed as separate from memory

I/O programming

- There are two ways of addressing I/O devices
- **Memory mapped I/O**
 - The address space is divided between memory and I/O devices
 - Higher order addresses can refer to device
 - Lower order addresses can refer to memory
 - `mov %eax, address` will fetch data from I/O or memory based on the address
 - E.g., memory range to from 0000 to BFFF
 - I/O range from C000 to CFFF
 - Device or memory selection based on address range
 - Different devices can have different addresses in the I/O range

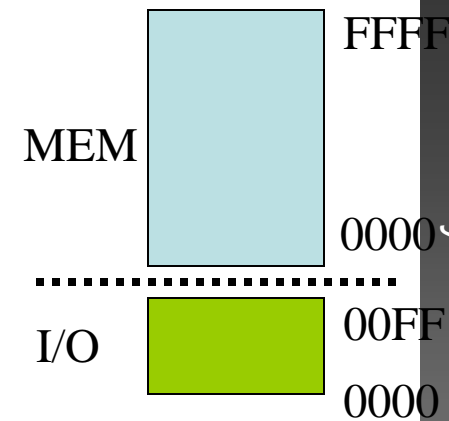
Memory mapped I/O



- Send or receive data to /from I/O device is a memory transfer instruction (mov) with the right address
- Main memory not selected when address is in I/O range
- Adv
 - Uniformity of programming, same mov works for I/O and memory
- Dis adv
 - Memory address space is reduced

I/O mapped I/O

- Memory and I/O devices use distinct address spaces
 - Isolated I/O
- Two separate instructions to address I/O devices
- A separate code or control signal based on the op code will select memory or I/O
- IN for input
- OUT for output
- mov for memory access
- Less flexible for programming



IN—Input from Port

Opcode	Instruction	Description
E4 <i>ib</i>	IN AL, <i>imm8</i>	Input byte from <i>imm8</i> I/O port address into AL
E5 <i>ib</i>	IN AX, <i>imm8</i>	Input byte from <i>imm8</i> I/O port address into AX
E5 <i>ib</i>	IN EAX, <i>imm8</i>	Input byte from <i>imm8</i> I/O port address into EAX
EC	IN AL,DX	Input byte from I/O port in DX into AL
ED	IN AX,DX	Input word from I/O port in DX into AX
ED	IN EAX,DX	Input doubleword from I/O port in DX into EAX

Description

This instruction copies the value from the I/O port specified with the second operand (source operand) to the destination operand (first operand). The source operand can be a byte-immediate or the DX register; the destination operand can be register AL, AX, or EAX, depending on the size of the port being accessed (8, 16, or 32 bits, respectively). Using the DX register as a source operand allows I/O port addresses from 0 to 65,535 to be accessed; using a byte immediate allows I/O port addresses 0 to 255 to be accessed.

When accessing an 8-bit I/O port, the opcode determines the port size; when accessing a 16- and 32-bit I/O port, the operand-size attribute determines the port size.

At the machine code level, I/O instructions are shorter when accessing 8-bit I/O ports. Here, the upper eight bits of the port address will be 0.

OUT—Output to Port

Opcode	Instruction	Description
E6 <i>ib</i>	OUT <i>imm8</i> , AL	Output byte in AL to I/O port address <i>imm8</i>
E7 <i>ib</i>	OUT <i>imm8</i> , AX	Output word in AX to I/O port address <i>imm8</i>
E7 <i>ib</i>	OUT <i>imm8</i> , EAX	Output doubleword in EAX to I/O port address <i>imm8</i>
EE	OUT DX, AL	Output byte in AL to I/O port address in DX
EF	OUT DX, AX	Output word in AX to I/O port address in DX
EF	OUT DX, EAX	Output doubleword in EAX to I/O port address in DX

Description

This instruction copies the value from the second operand (source operand) to the I/O port specified with the destination operand (first operand). The source operand can be register AL, AX, or EAX, depending on the size of the port being accessed (8, 16, or 32 bits, respectively); the destination operand can be a byte-immediate or the DX register. Using a byte immediate allows I/O port addresses 0 to 255 to be accessed; using the DX register as a source operand allows I/O ports from 0 to 65,535 to be accessed.

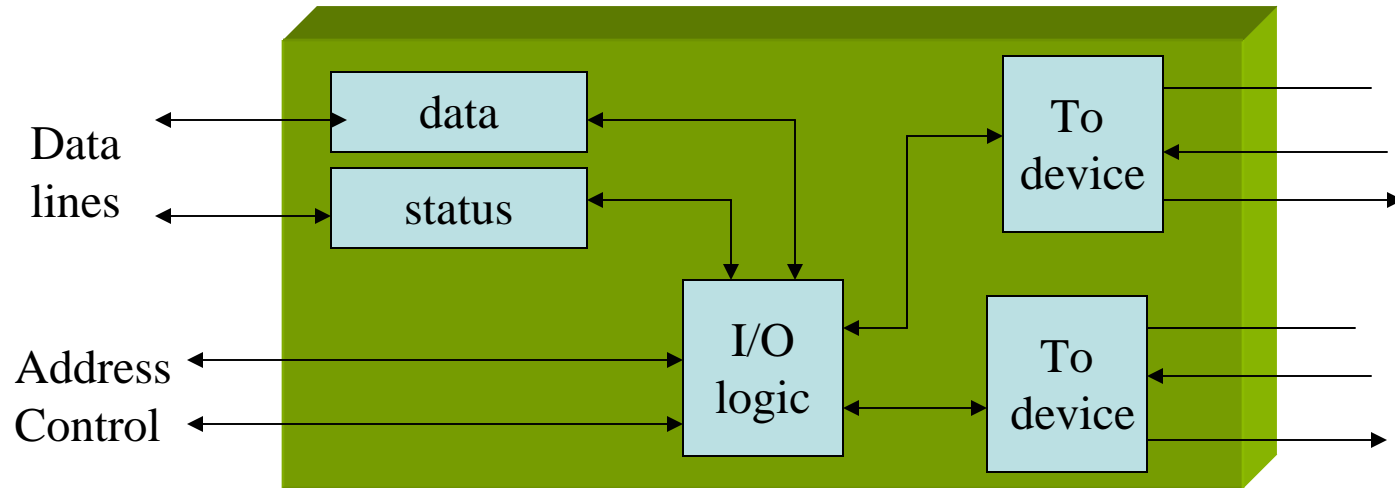
The size of the I/O port being accessed is determined by the opcode for an 8-bit I/O port or by the operand-size attribute of the instruction for a 16- or 32-bit I/O port.

At the machine code level, I/O instructions are shorter when accessing 8-bit I/O ports. Here, the upper eight bits of the port address will be 0.

Interfacing with I/O

- Many devices, with varying speeds, complexity
- CPU/bus shared among all peripherals and memory
- CPU should be able to select a device and transfer data to the device
- Interpretation of data left to each device
- Unlike memory, device need to be ready before initiating transfer
- All of this handed by I/O module

I/O module



- CPU selects device by means of address
- Data corresponds to instructions for device
- Each device has its own set of commands
- Status of device can be checked by reading status registers

Data transfer schemes

- There are two schemes
- Programmed data transfers
 - CPU transfers data from I/O devices onto registers
 - Useful for small data transfers
- Direct memory access or DMA
 - Device or I/O module directly transfers data to memory
 - Useful for large block transfers

Programmed I/O

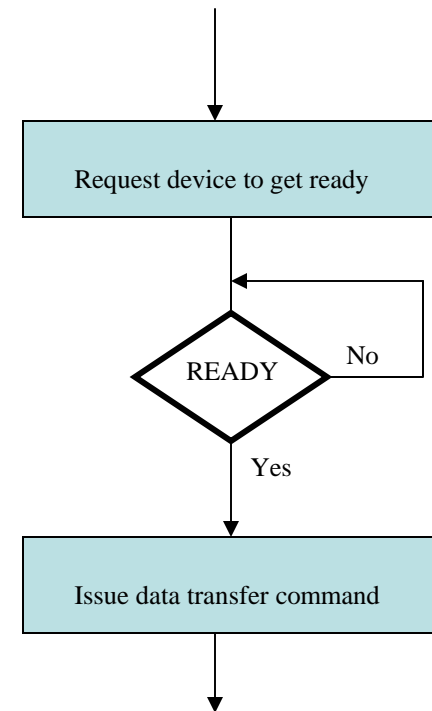
- Programmed I/O can be further classified as
 - Synchronous transfer
 - Asynchronous transfer
 - Interrupt driven transfer
- All of the above can be used to interface with different I/O devices
- Require special hardware features in the CPU

Synchronous transfer

- Simplest among three
- CPU and I/O speed match
- Transfer a byte, word, or double word
- Memory mapped
 - `mov %eax, 2`
 - Address of device port is 2
- I/O mapped
 - `mov $2, %edx`
 - `out %eax, %edx`
- Similarly for Input device,
 - Memory mapped: `mov 3, %eax` or
 - I/O mapped `mov $3, %edx`
 - `in %edx, %eax`

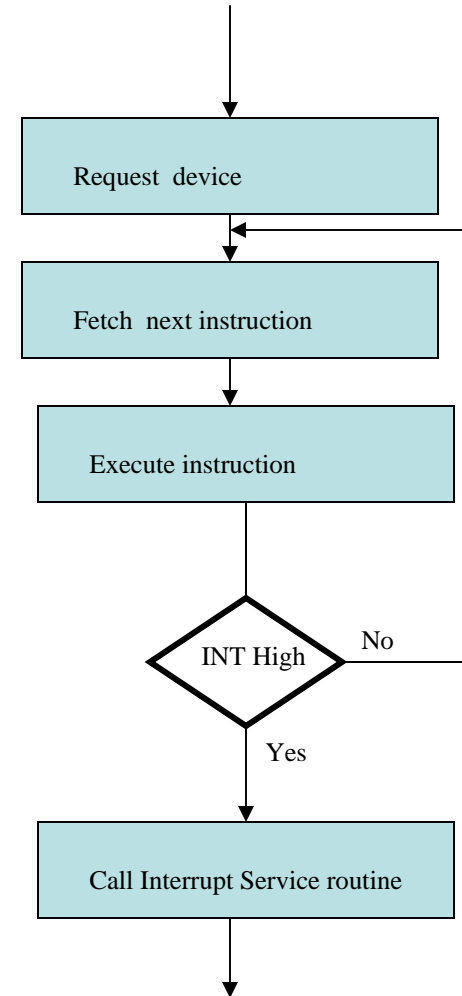
Asynchronous transfer

- I/O devices slower
- Instruct device to be ready
- Wait until device ready
- Device has status flag/register
- Busy waiting
- Waste of CPU resources



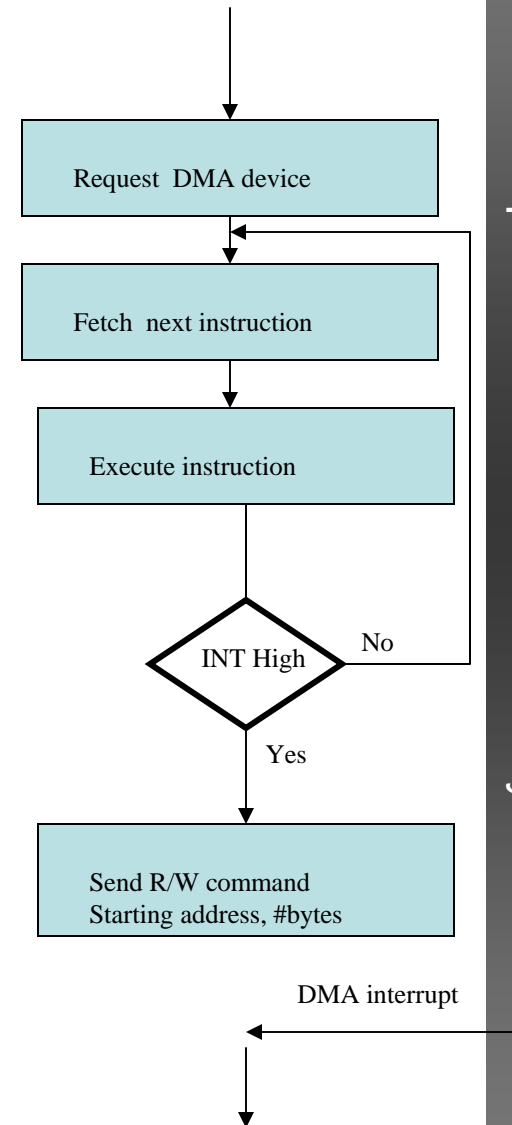
Interrupt driven I/O

- Processor need not wait for slow device
- Processor continues with other instructions
- Device interrupts processor when ready
- Interrupt Service Routine
 - CPU transfers word from device to register
 - CPU writes word from register to memory

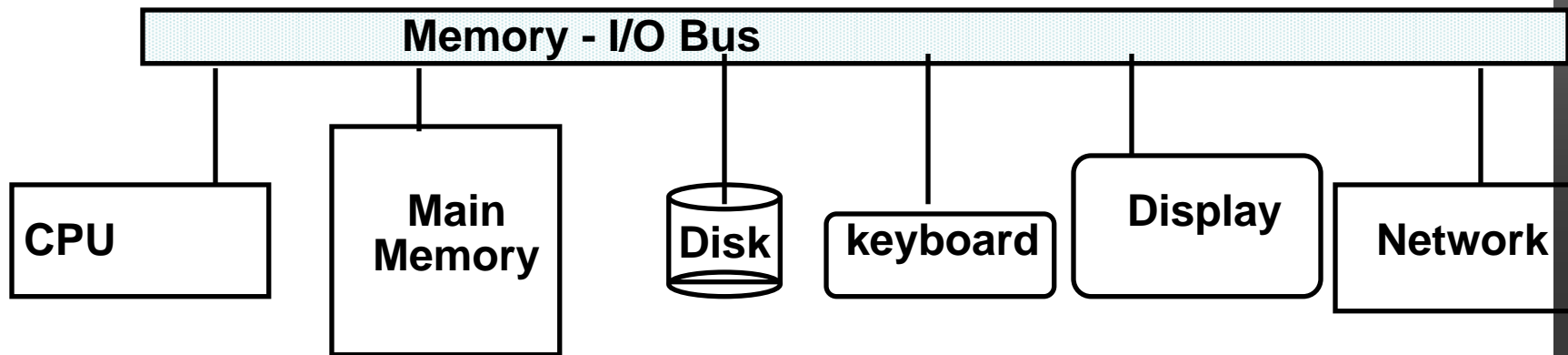


DMA or direct memory

- Bulk data transfers
- Direct device to memory transfer
- Memory bus is contention between CPU and DMA unit
- During DMA
- Either CPU is in hold state
- Or
- Cycle stealing
- CPU and DMA access in interleaved



System bus

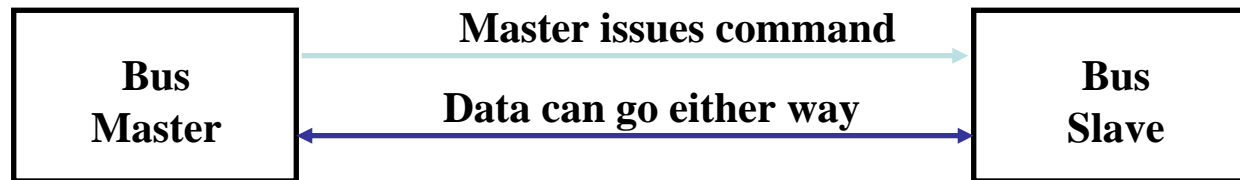


- A bus is a shared communication link
- Contains address bus, data bus
- Each bus is a set of wires
- Bus can transfer several bits between devices connected by bus
- Bus width determines the number of bits transferred in a cycle

Characteristics of bus

- Several devices can be connected
- Single bus for all devices – cost sharing
- Added/removed without affecting others
- I/O devices can be connected to other devices following the same bus standard
- Disadvantages:
- Bus contention
- Speed of I/O devices determined by bus speed
- Bus speed determined by number of devices
- Slower devices impact others

Bus architecture



- Any interaction consists of two steps
 - 1. Issue command 2. transfer data
- Master Initiates
 - Issues command, starting address, #bytes
- Slave Responds
 - Sends or receives data as per command from master

Computer buses

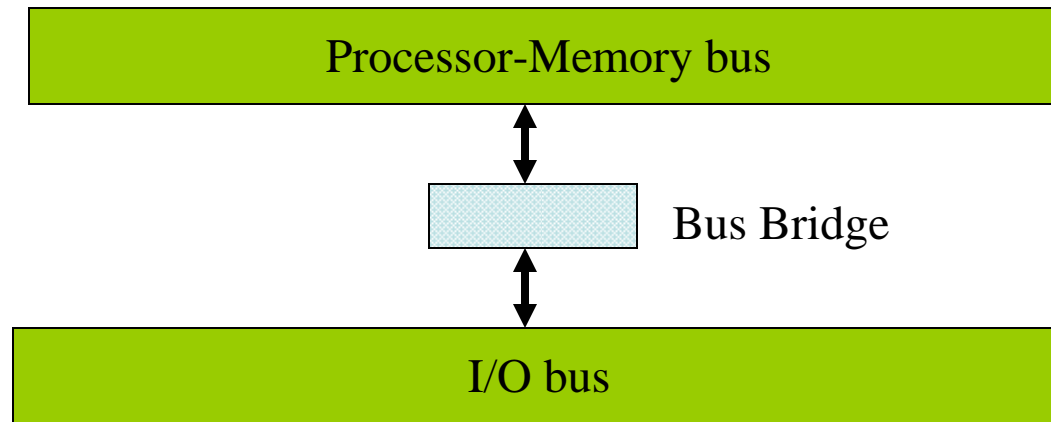
- Modern computers have several I/O devices
- Varying speeds
- A simple linear bus will not suffice
- Modern computers have hierarchical buses
- Bus is split into different segments
- CPU-Memory one bus
- CPU-I/O devices another bus
- CPU-cache – another bus

Backplane bus



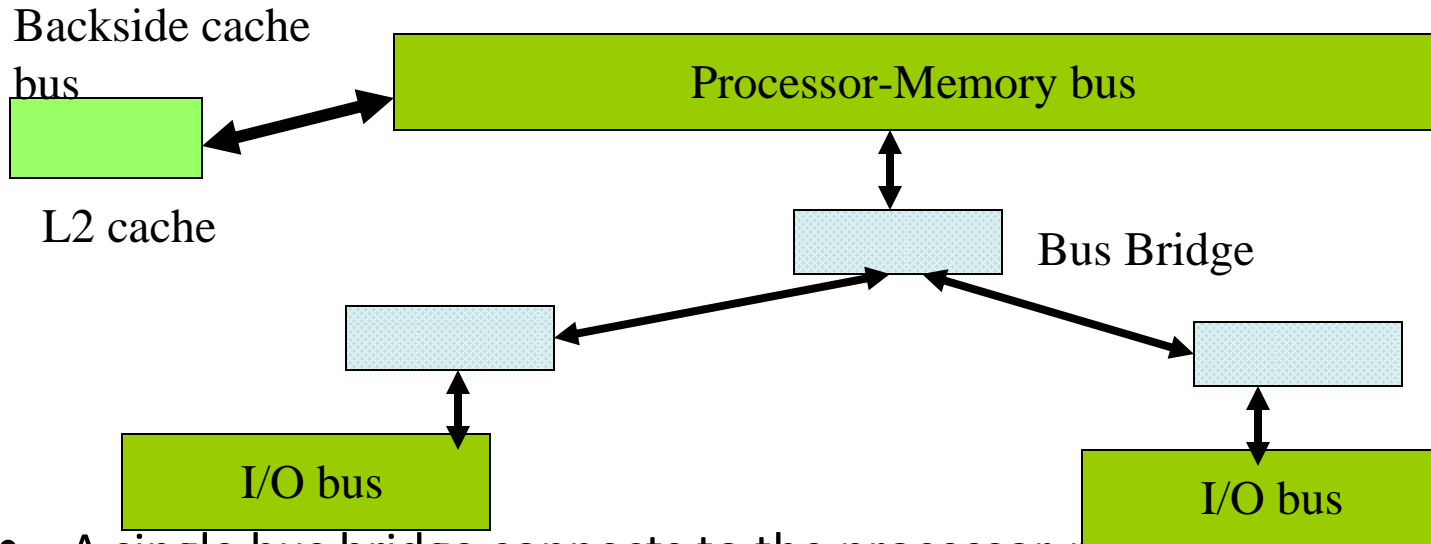
- Single bus for memory and I/O
- Cheap
- Slow and bus becomes bottleneck

Two-bus systems



- Processor-memory traffic on one bus
- I/O devices connected by a bridge
- Bridge can connect to different kinds of buses
- Traffic is isolated
- I/O buses can provided expansion slots for devices

hierarchical-bus systems



- A single bus bridge connects to the processor-memory bus
- Other I/O buses connected to this bus bridge (tree)
- CPU-memory sees little contention
- Costly

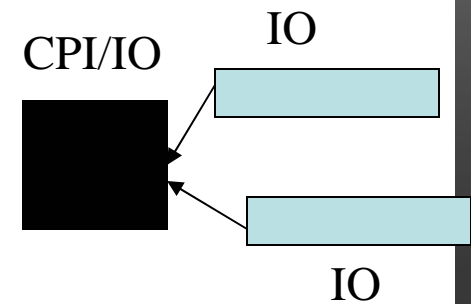
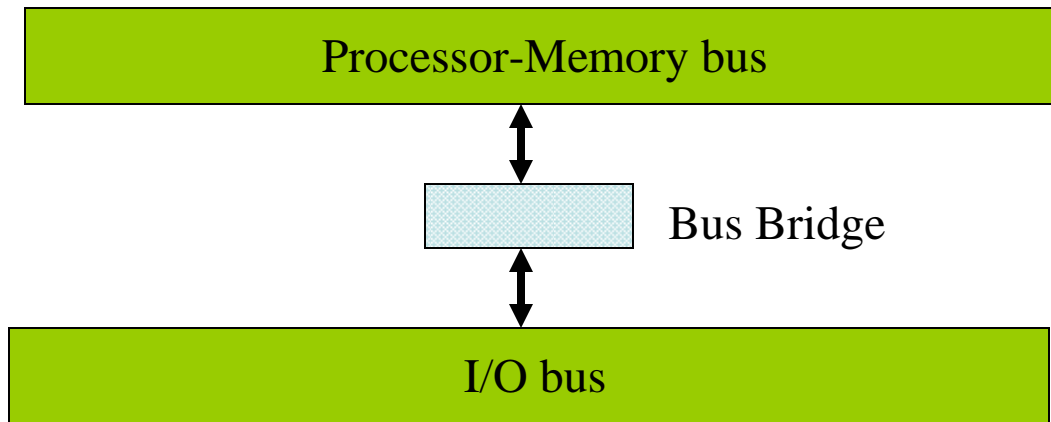
Examples of buses

- ISA bus – Industry Standard bus
 - Old technology
 - 8 Mhz, < 1 byte transfer/cycle, bus B/W 5.3 MB/sec
- EISA bus – Extended ISA
 - Old technology
 - 8 Mhz, 4 byte transfer, bus B/W 32 Mb/sec
- PCI bus- Peripheral Component Interconnect
 - Speeds up to 132 MB/s
 - Bus speed of 33mhz, 4 Bytes/transfer
 - PCI popularized Plug and Play

Examples of buses

- PCI-X extended PCI
 - 133 MHz, 8 bytes/transfer, 1064 MB/sec or 1 GB/sec
 - Used to connect gigabit ethernet, high speed disks
- SCSI (Small Computer System Interface)
 - Capable of handling internal/external peripherals
 - Speed anywhere from 80 – 640 Mb/s
 - Many types of SCSI
 - Fast SCSI
 - Ultra SCSI
 - Ultra wide SCSI

Parallel vs serial (point-to-point) bus



- Parallel bus
 - Bus shared among devices
 - Bus arbitration is slow
 - Example: PCI, SCSI
- Serial I/O
 - Point to pint links connected directly to CPU
 - Requires lots of additional high speed hardware
 - Examples: SATA, USB, firewire

USB

- 1.0
 - plug-and-play
 - Full speed USB devices signal at 12Mb/s
 - Low speed devices use a 1.5Mb/s subchannel.
 - Up to 127 devices chained together
- 2.0
 - data rate of 480 mega bits per second



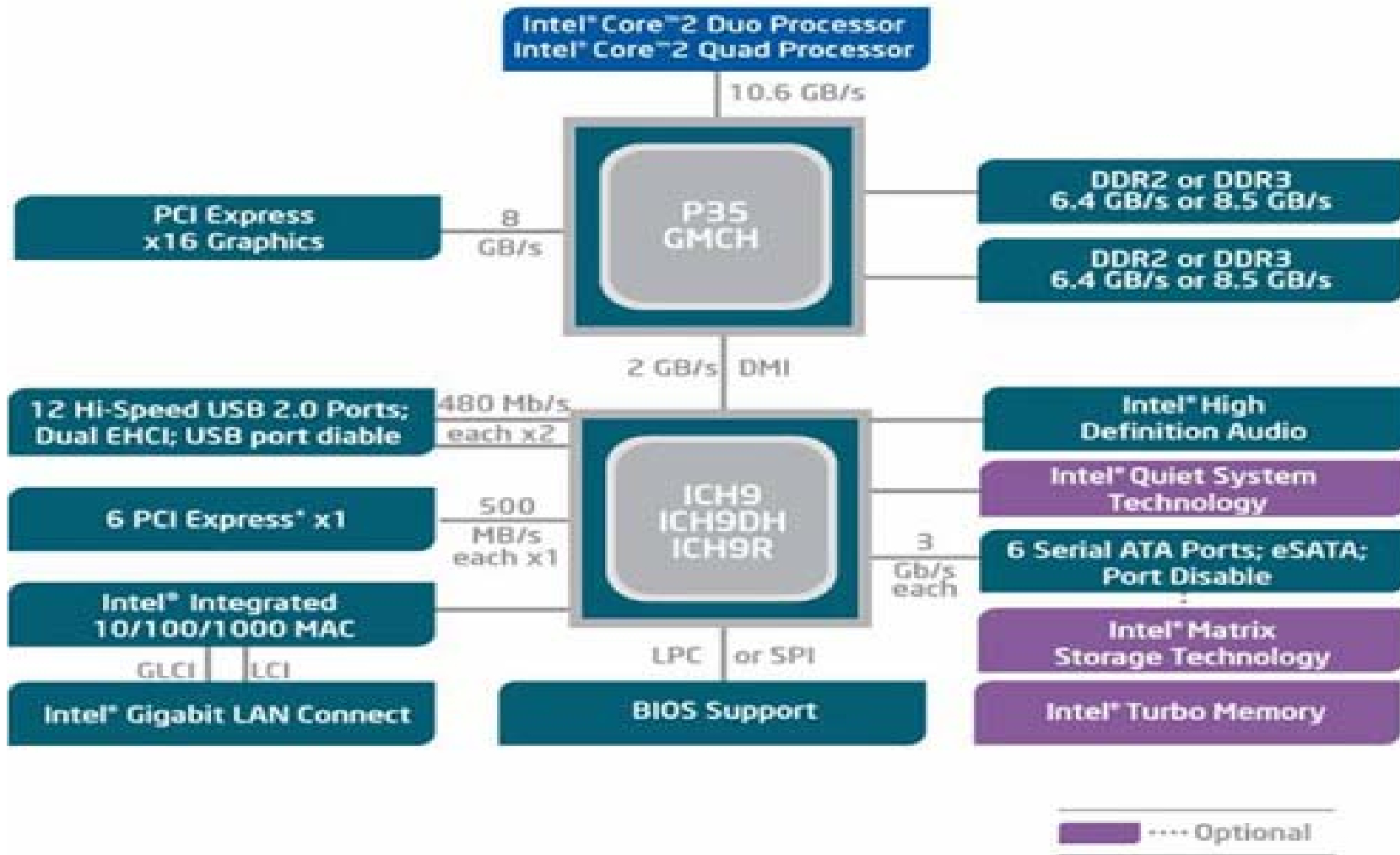
Firewire (apple)



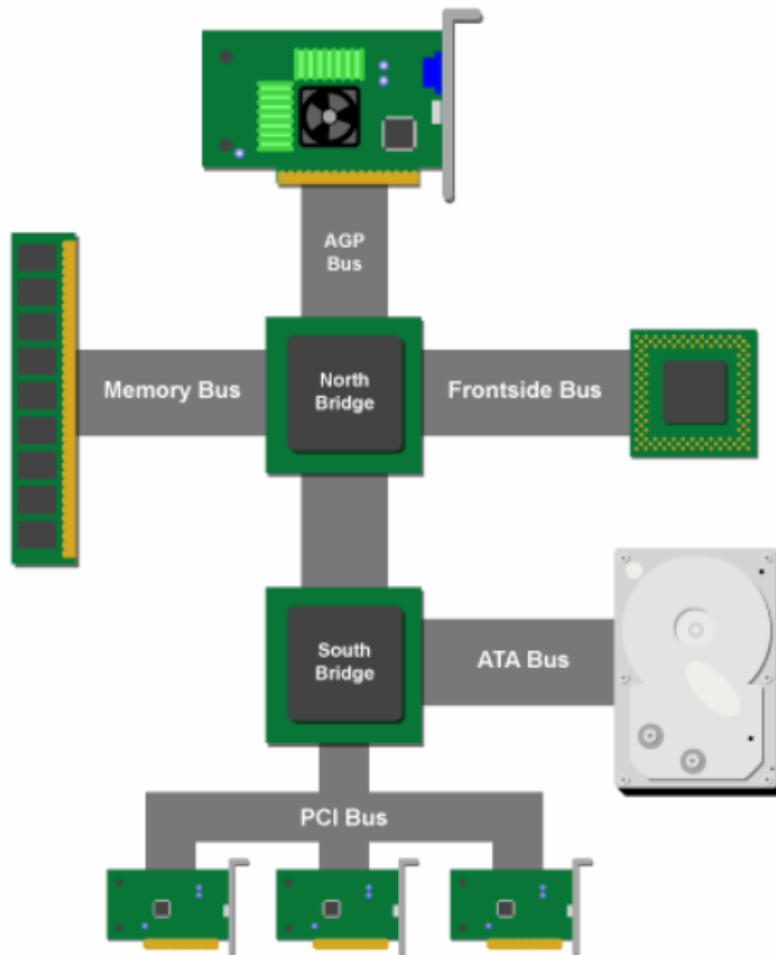
- High speed serial port
- 400 mbps transfer rate
- 30 times faster than USB 1.0
- plug-and-play



Intel Bus



North bridge and South bridge bus



http://www.testbench.in/pcie_sys_2.PNG

