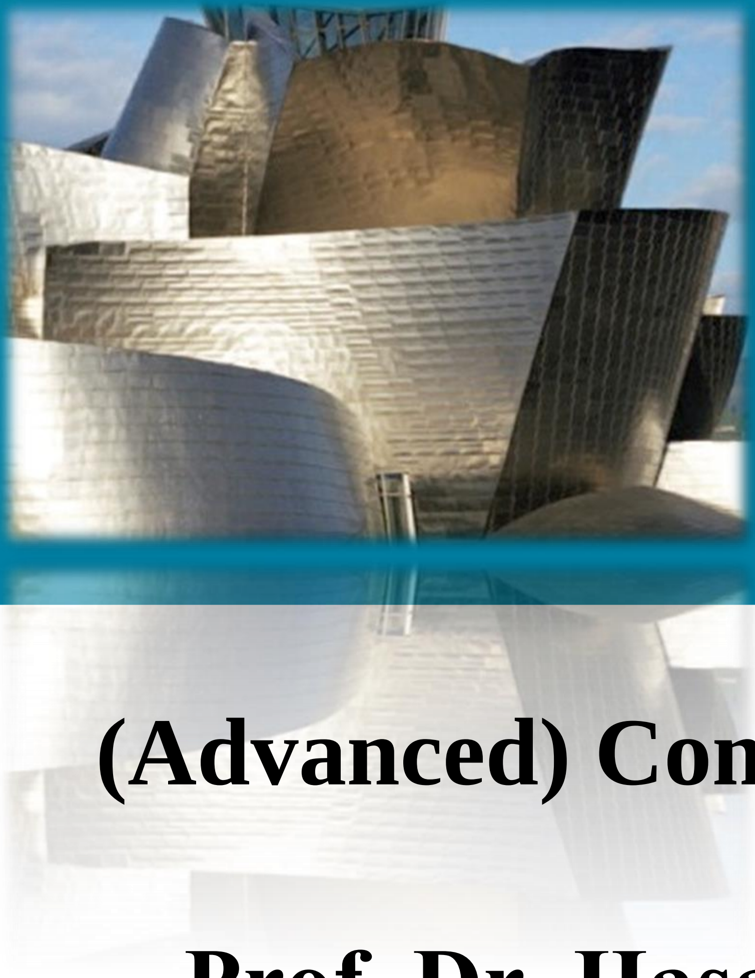


(Advanced) Computer Architechture

Prof. Dr. Hasan Hüseyin BALIK

(3rd Week)

Outline

2. The computer system
    - A Top-Level View of Computer Function and Interconnection
    - Operating System Support

+

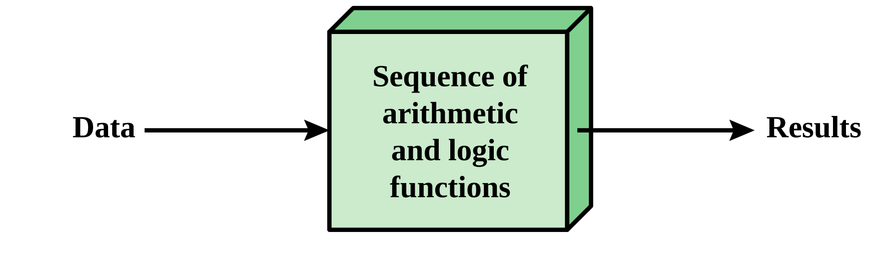
2.1 A Top-Level View of Computer Function and Interconnection

2.1 Outline

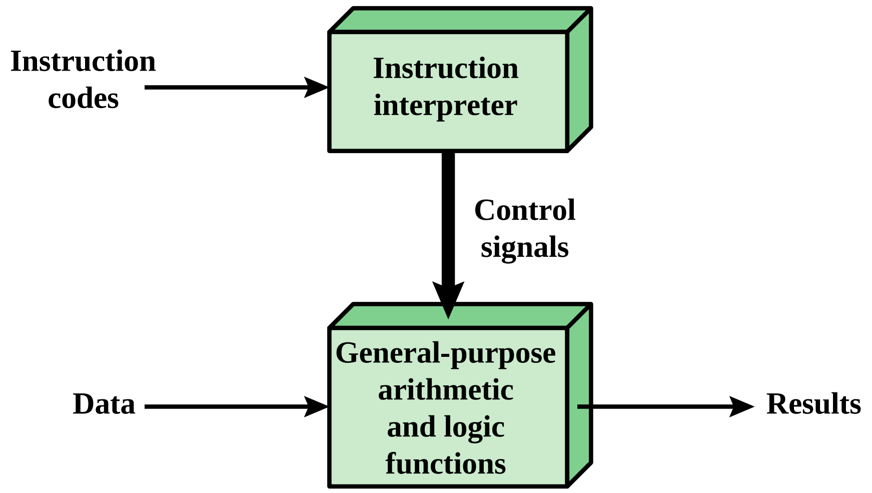
- Computer Components
- Computer Function
- Interconnection Structures
- Bus Interconnection
- Point-to-Point Interconnect
- PCI Express

Computer Components

- Contemporary computer designs are based on concepts developed by John von Neumann at the Institute for Advanced Studies, Princeton
- Referred to as the *von Neumann architecture* and is based on three key concepts:
  - Data and instructions are stored in a single read-write memory
  - The contents of this memory are addressable by location, without regard to the type of data contained there
  - Execution occurs in a sequential fashion (unless explicitly modified) from one instruction to the next
- *Hardwired program*
  - The resulting “program” is in the form of hardware and is termed a hardwired program.
  - The result of the process of connecting the various components in the desired configuration



(a) Programming in hardware



(b) Programming in software

Figure 3.1 Hardware and Software Approaches

Software and I/O Components

Software

- A sequence of codes or instructions
- Part of the hardware interprets each instruction and generates control signals
- Provide a new sequence of codes for each new program instead of rewiring the hardware

Major components:

- CPU
  - Instruction interpreter
  - Module of general-purpose arithmetic and logic functions
- I/O Components
  - Input module
    - Contains basic components for accepting data and instructions and converting them into an internal form of signals usable by the system
  - Output module
    - Means of reporting results

Memory, MAR, and MBR

Memory address register (MAR)

- Specifies the address in memory for the next read or write

Memory buffer register (MBR)

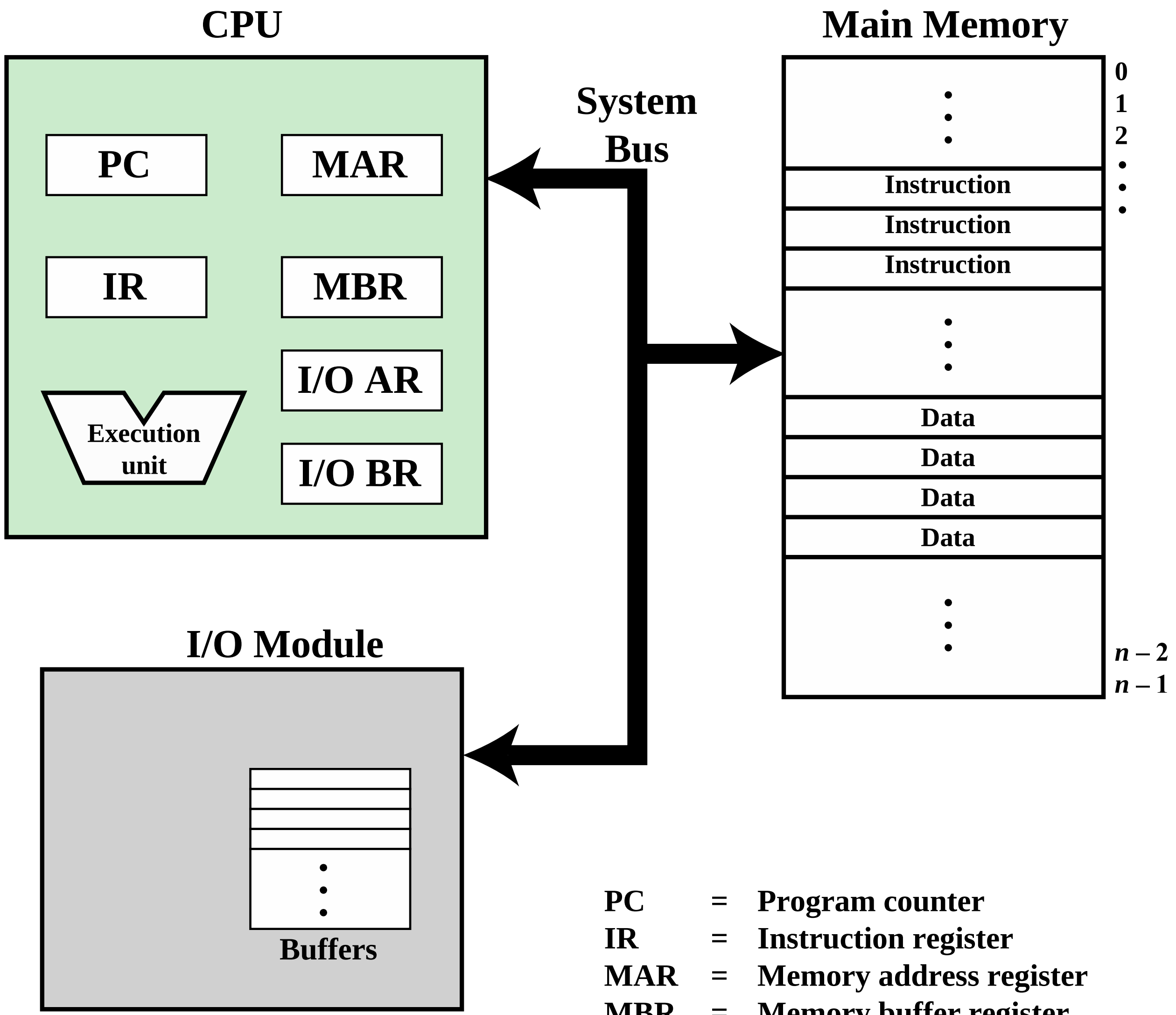
- Contains the data to be written into memory or receives the data read from memory

I/O address register (I/OAR)

- Specifies a particular I/O device

I/O buffer register (I/OBR)

- Used for the exchange of data between an I/O module and the CPU



PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

Figure 3.2 Computer Components: Top-Level View

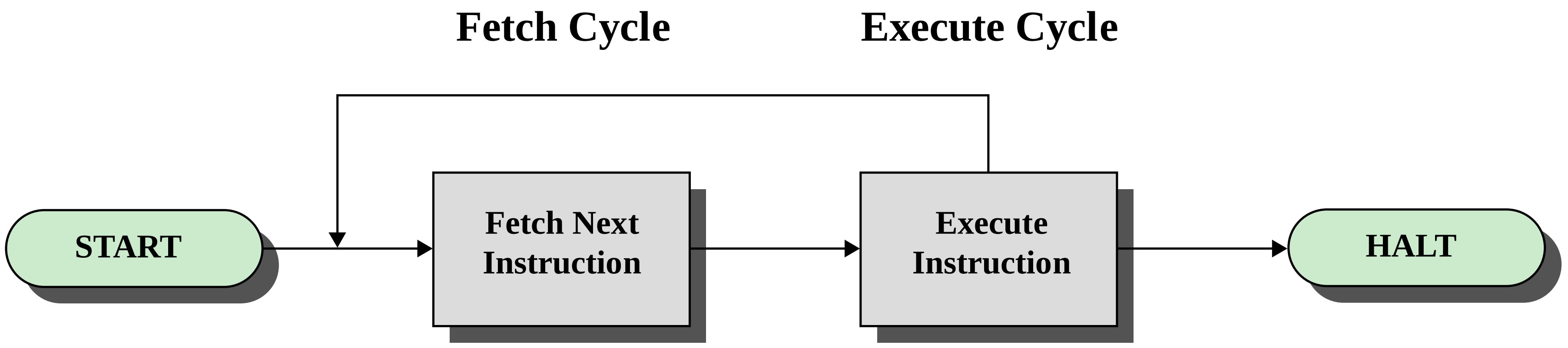
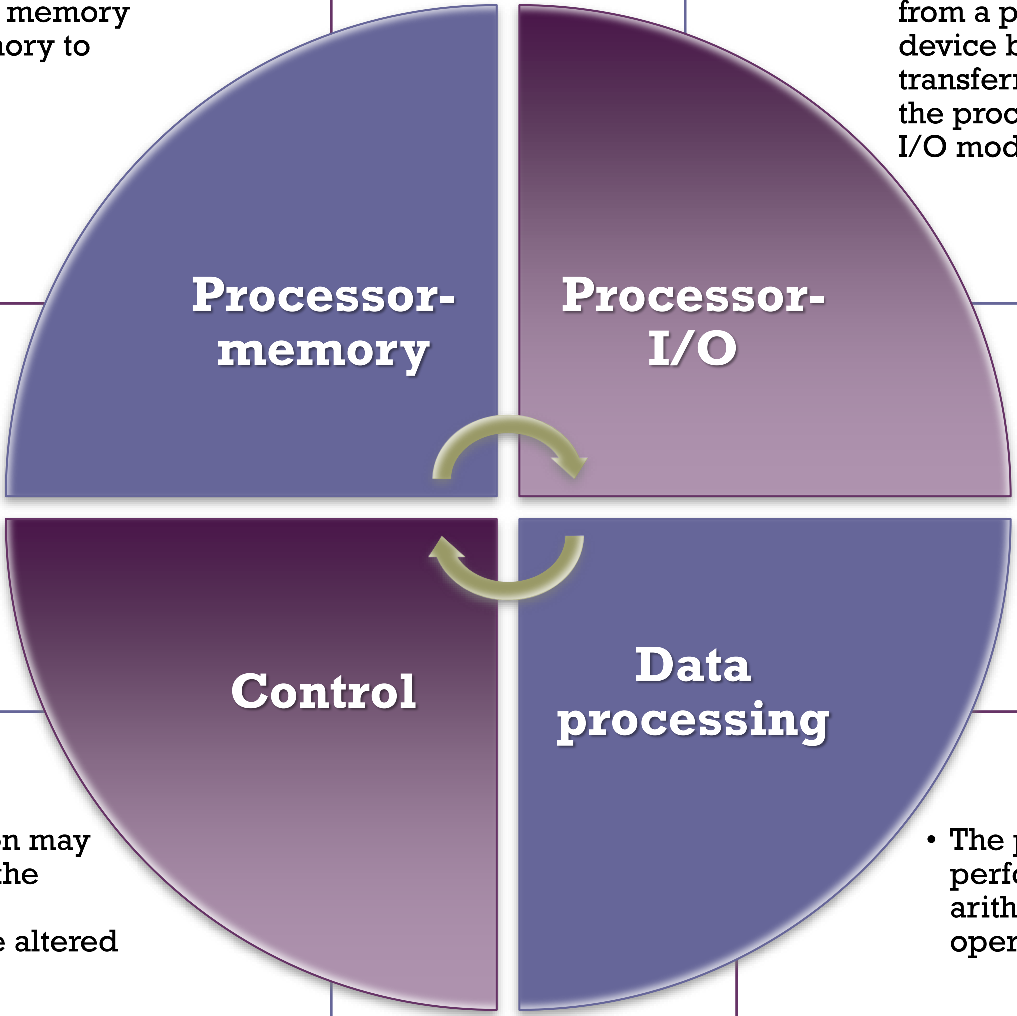


Figure 3.3 Basic Instruction Cycle

Fetch Cycle

- At the beginning of each instruction cycle the processor fetches an instruction from memory
- The program counter (PC) holds the address of the instruction to be fetched next
- The processor increments the PC after each instruction fetch so that it will fetch the next instruction in sequence
- The fetched instruction is loaded into the instruction register (IR)
- The processor interprets the instruction and performs the required action

Action Categories

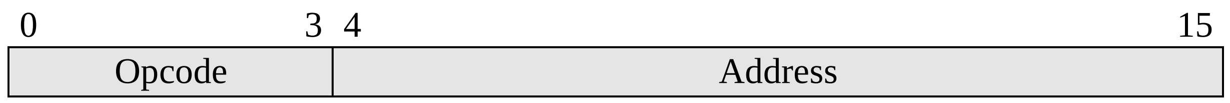


- Data transferred from processor to memory or from memory to processor

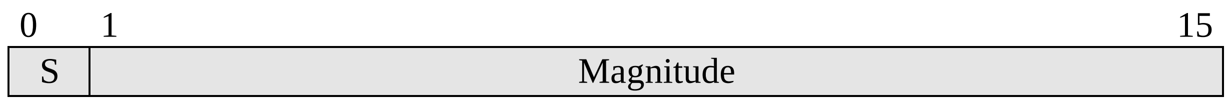
- Data transferred to or from a peripheral device by transferring between the processor and an I/O module

- An instruction may specify that the sequence of execution be altered

- The processor may perform some arithmetic or logic operation on data



(a) Instruction format



(b) Integer format

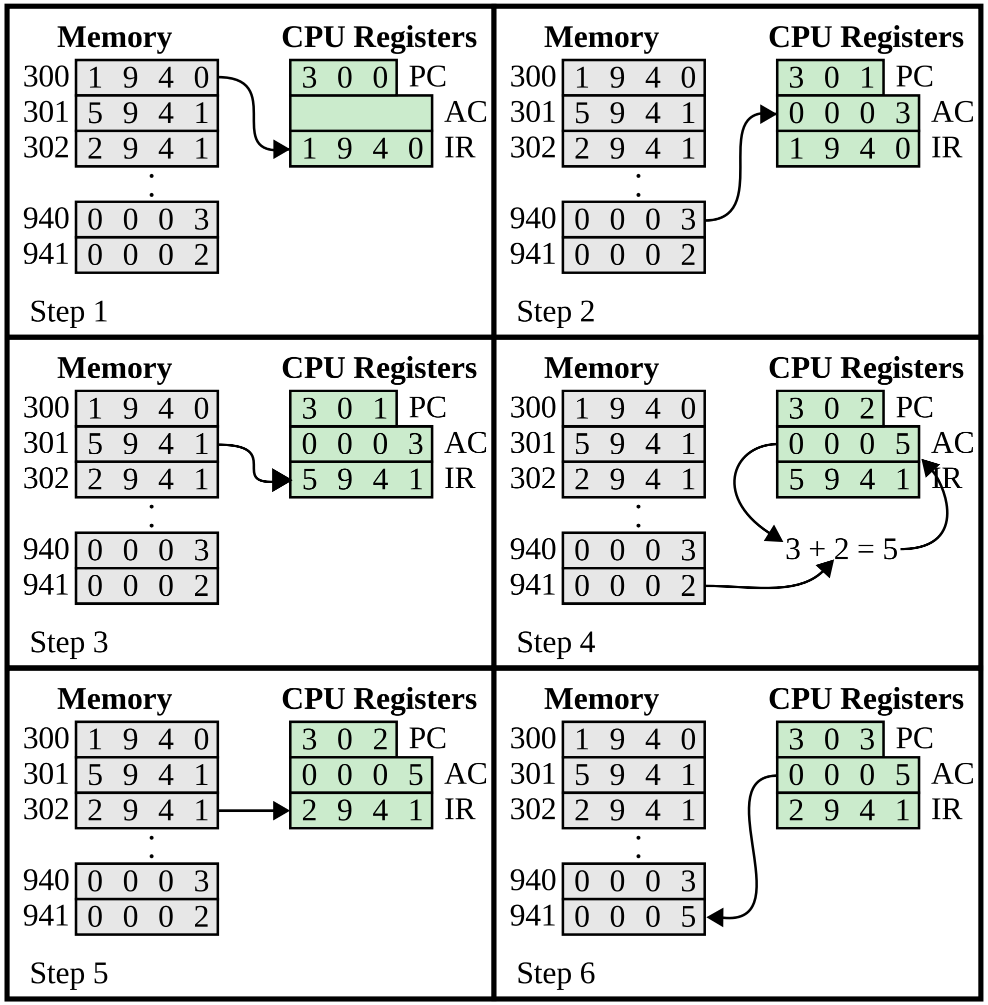
Program Counter (PC) = Address of instruction
Instruction Register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

(d) Partial list of opcodes

Figure 3.4   Characteristics of a Hypothetical Machine



0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)

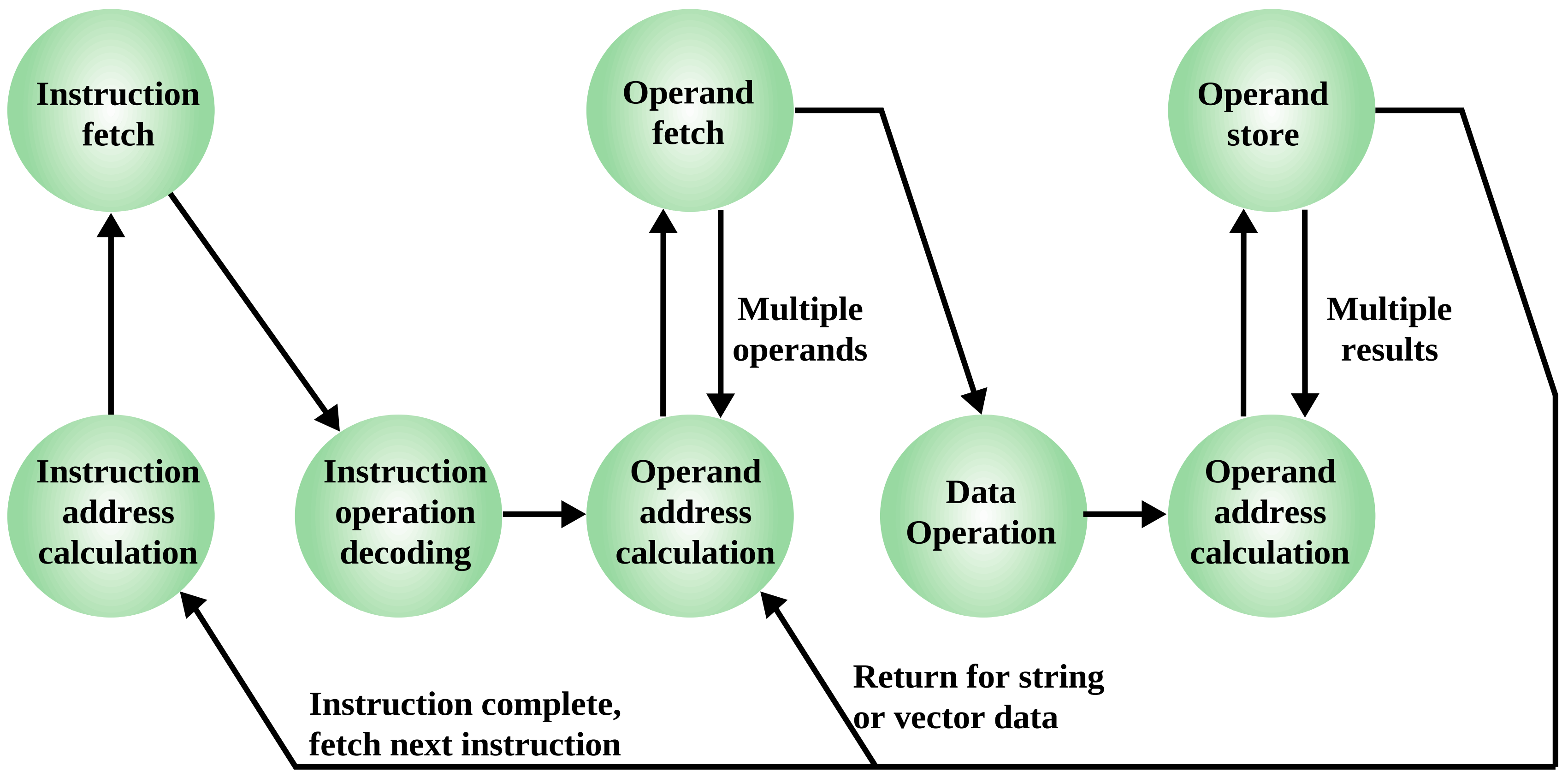
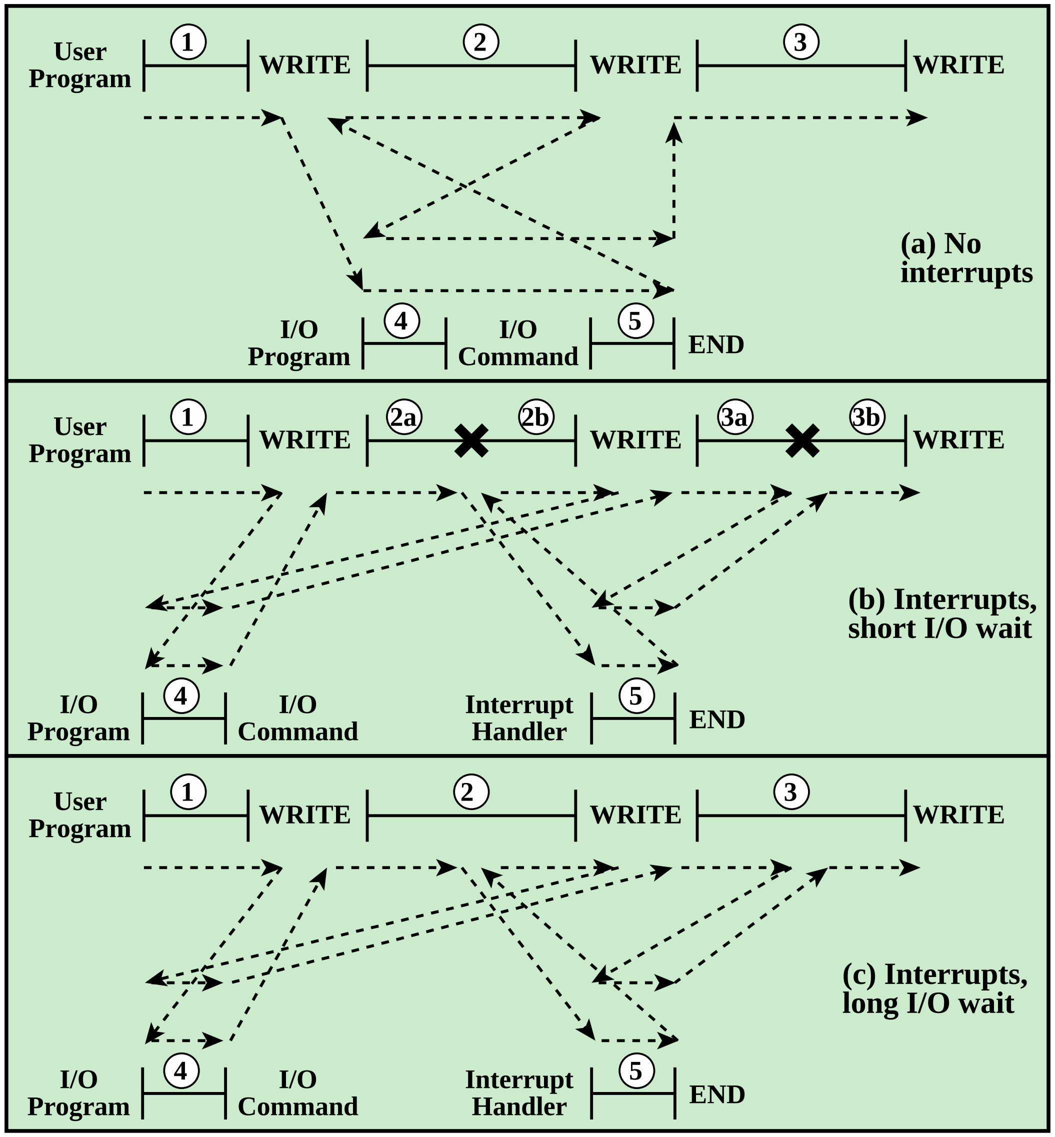

Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Return for string or vector data
Instruction complete, fetch next instruction

Figure 3.6 Instruction Cycle State Diagram

Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation, request service from the processor, or to signal a variety of error conditions.
Hardware Failure	Generated by a failure such as power failure or memory parity error.



✖ = interrupt occurs during course of execution of user program

Figure 3.7 Program Flow of Control Without and With Interrupts

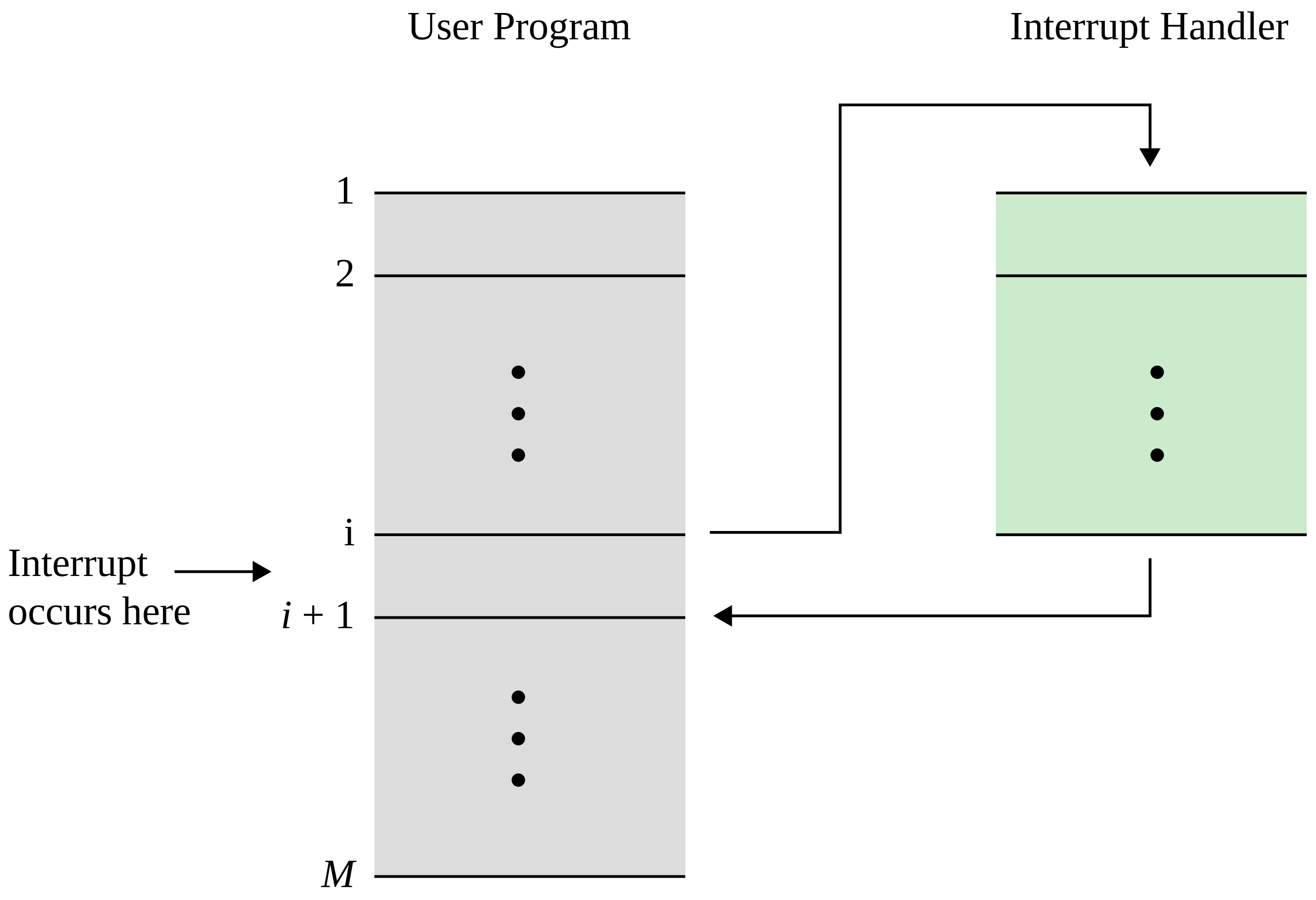


Figure 3.8 Transfer of Control via Interrupts

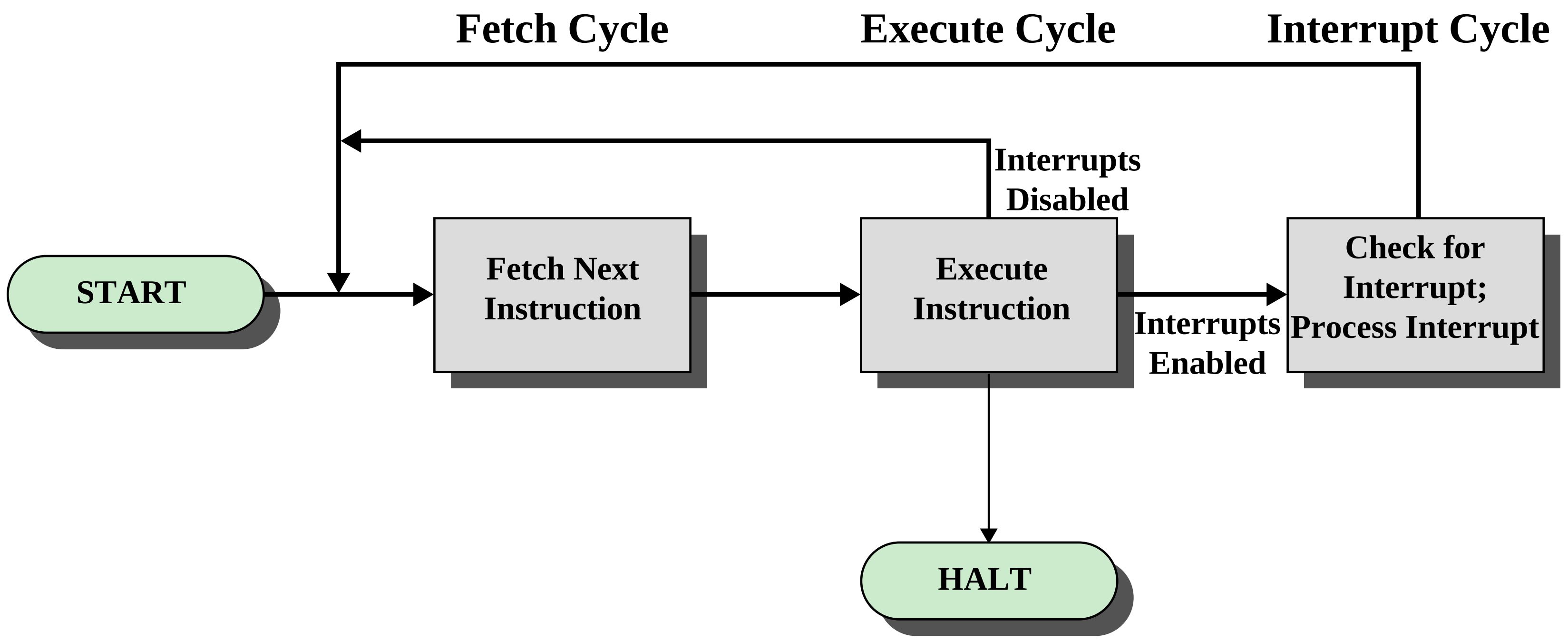


Figure 3.9 Instruction Cycle with Interrupts

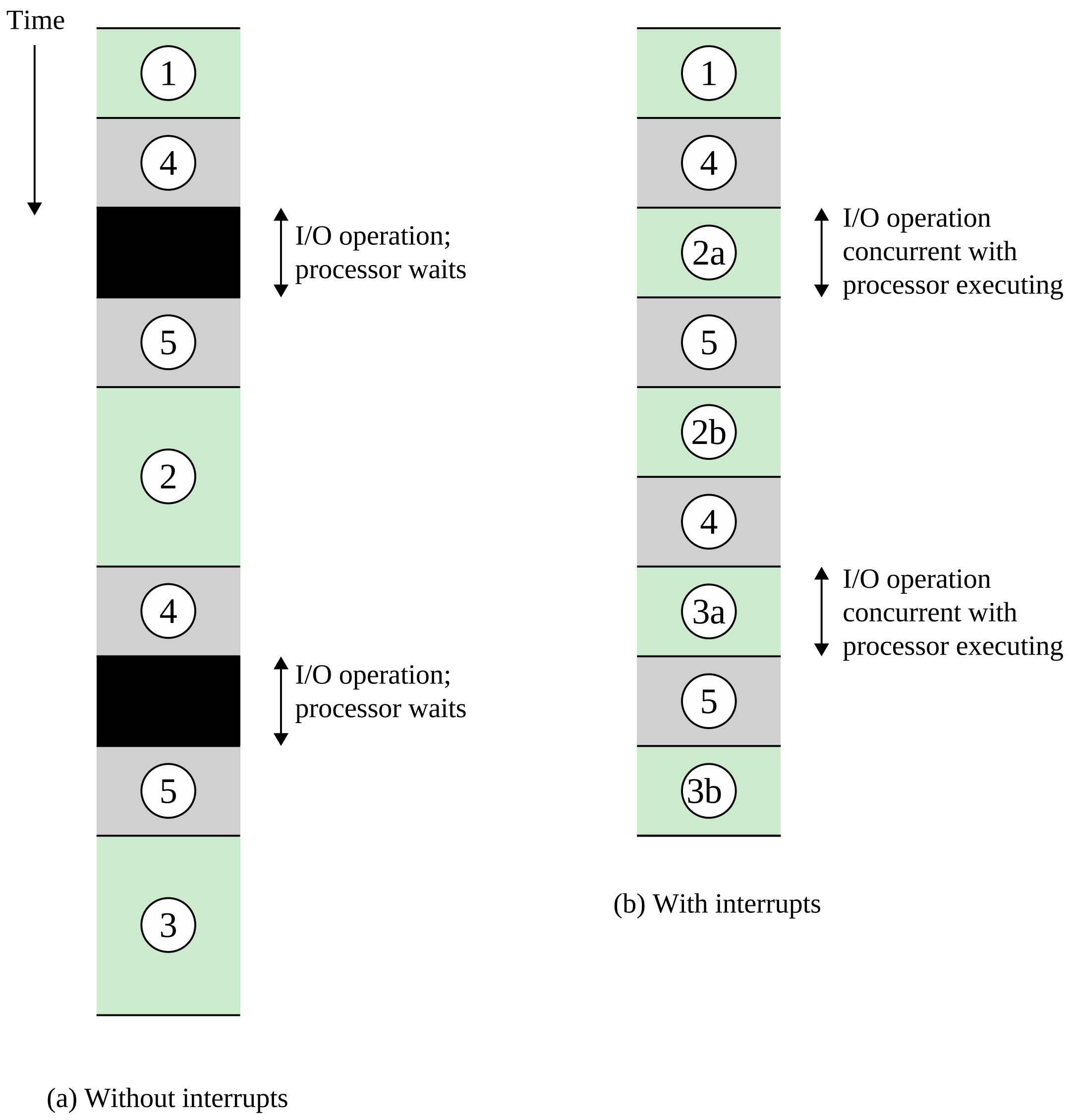

Time
1
4
I/O operation; processor waits
5
2
4
I/O operation; processor waits
5
3
(a) Without interrupts
1
4
2a
I/O operation concurrent with processor executing
5
2b
4
3a
I/O operation concurrent with processor executing
5
3b
(b) With interrupts

Figure 3.10 Program Timing: Short I/O Wait

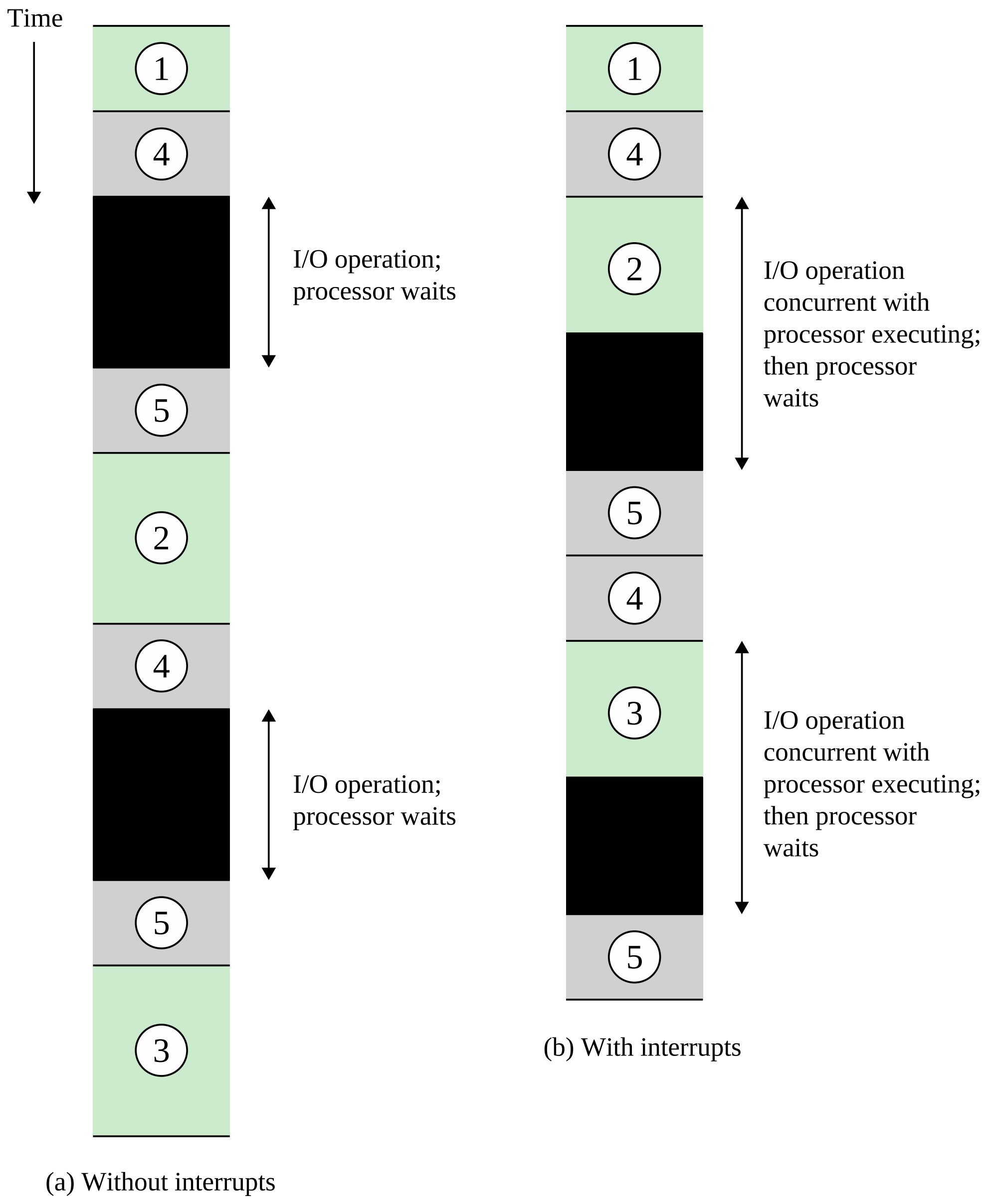

Time
1
4
I/O operation; processor waits
5
2
4
I/O operation; processor waits
5
3
(a) Without interrupts
1
4
2
I/O operation concurrent with processor executing; then processor waits
5
4
3
I/O operation concurrent with processor executing; then processor waits
5
(b) With interrupts

Figure 3.11 Program Timing: Long I/O Wait

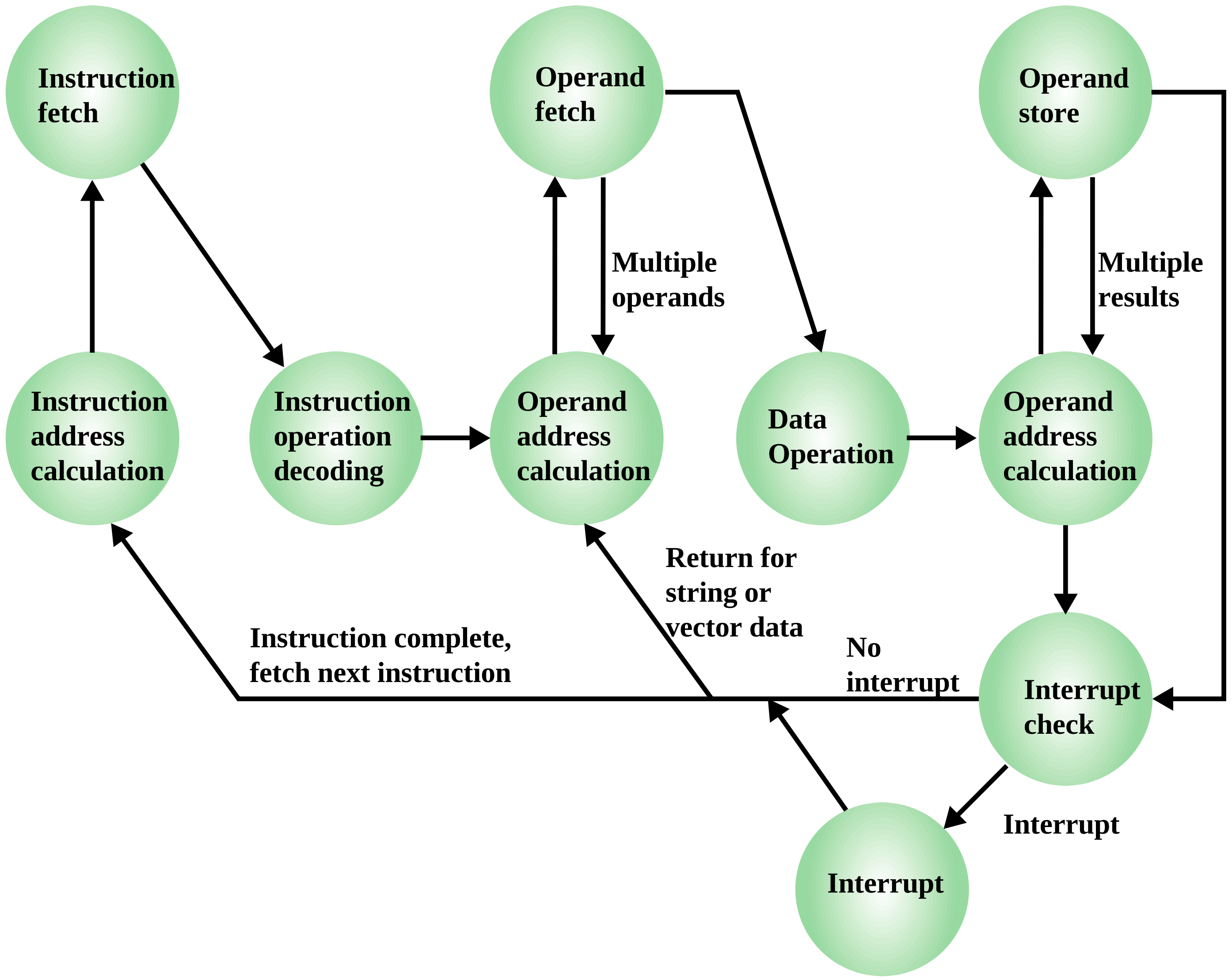
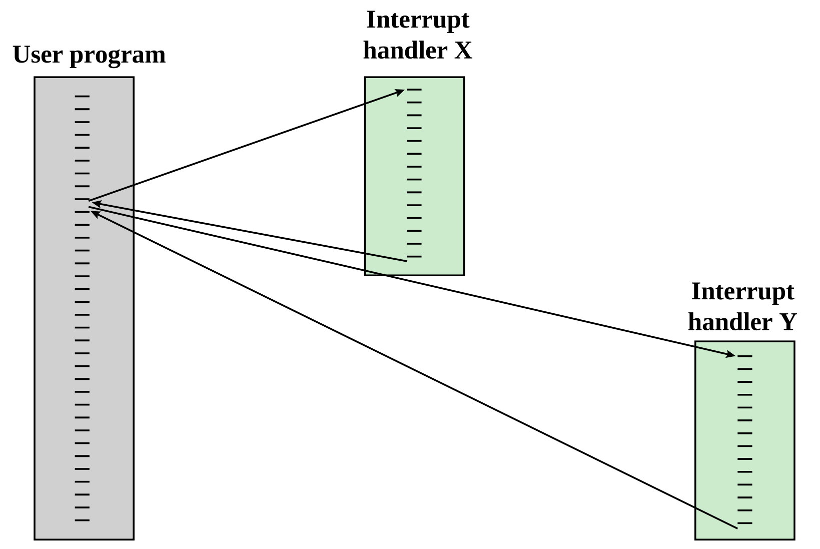
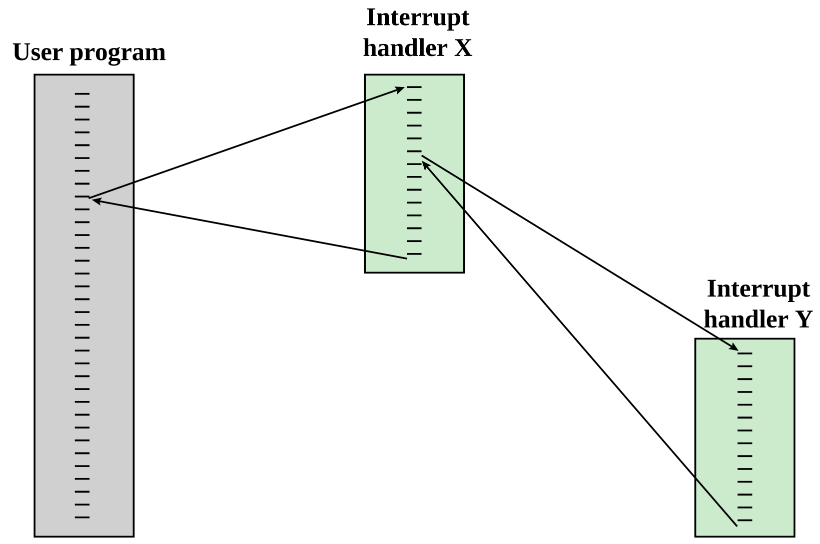

Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Return for string or vector data
Instruction complete, fetch next instruction
No interrupt
Interrupt check
Interrupt
Interrupt

Figure 3.12 Instruction Cycle State Diagram, With Interrupts



(a) Sequential interrupt processing



(b) Nested interrupt processing

Figure 3.13 Transfer of Control with Multiple Interrupts

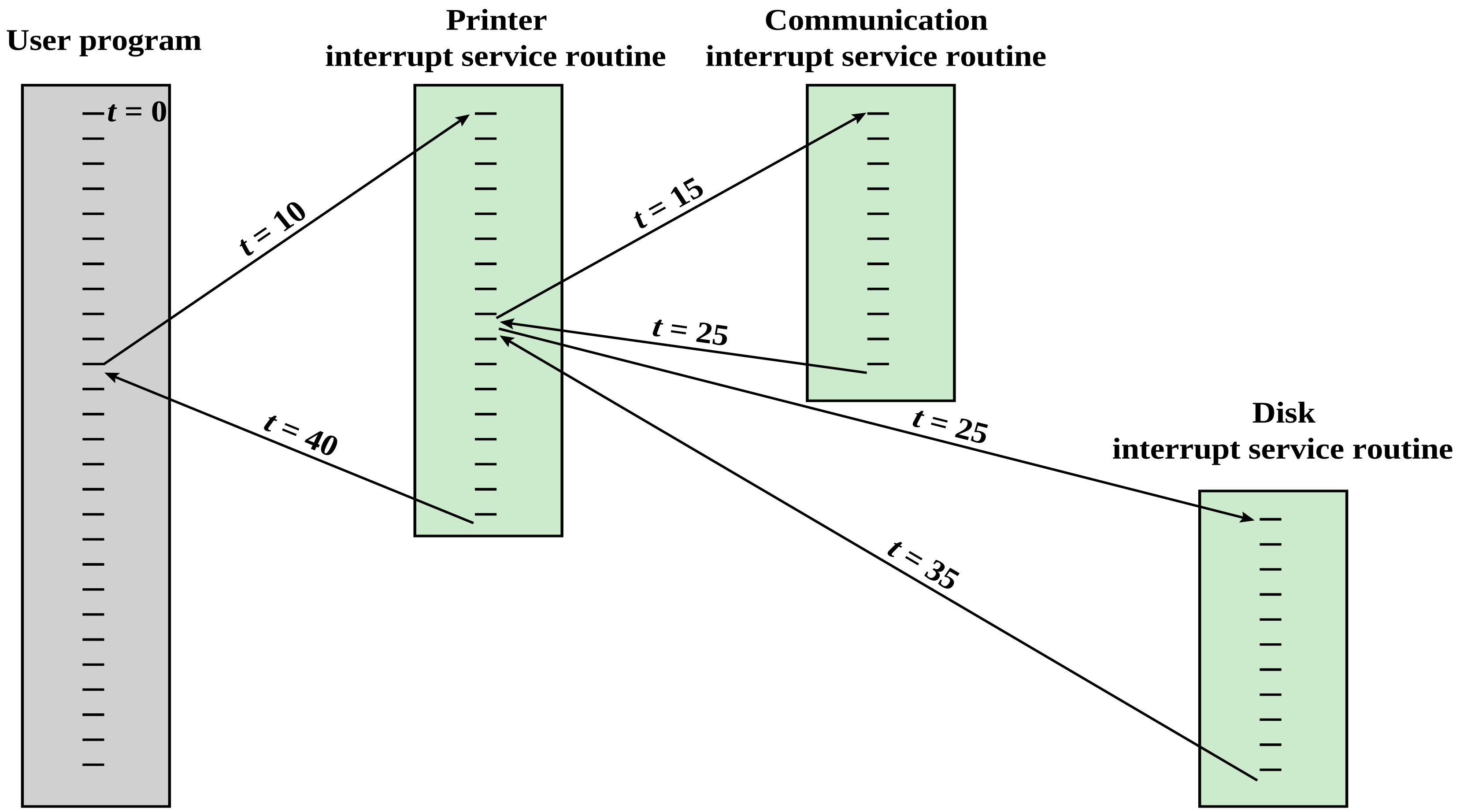

User program
Printer interrupt service routine
Communication interrupt service routine
Disk interrupt service routine
t = 0
t = 10
t = 15
t = 25
t = 25
t = 35
t = 40

Figure 3.14 Example Time Sequence of Multiple Interrupts

I/O Function

- I/O module can exchange data directly with the processor
- Processor can read data from or write data to an I/O module
  - Processor identifies a specific device that is controlled by a particular I/O module
  - I/O instructions rather than memory referencing instructions
- In some cases it is desirable to allow I/O exchanges to occur directly with memory
  - The processor grants to an I/O module the authority to read from or write to memory so that the I/O memory transfer can occur without tying up the processor
  - The I/O module issues read or write commands to memory relieving the processor of responsibility for the exchange
  - This operation is known as direct memory access (DMA)

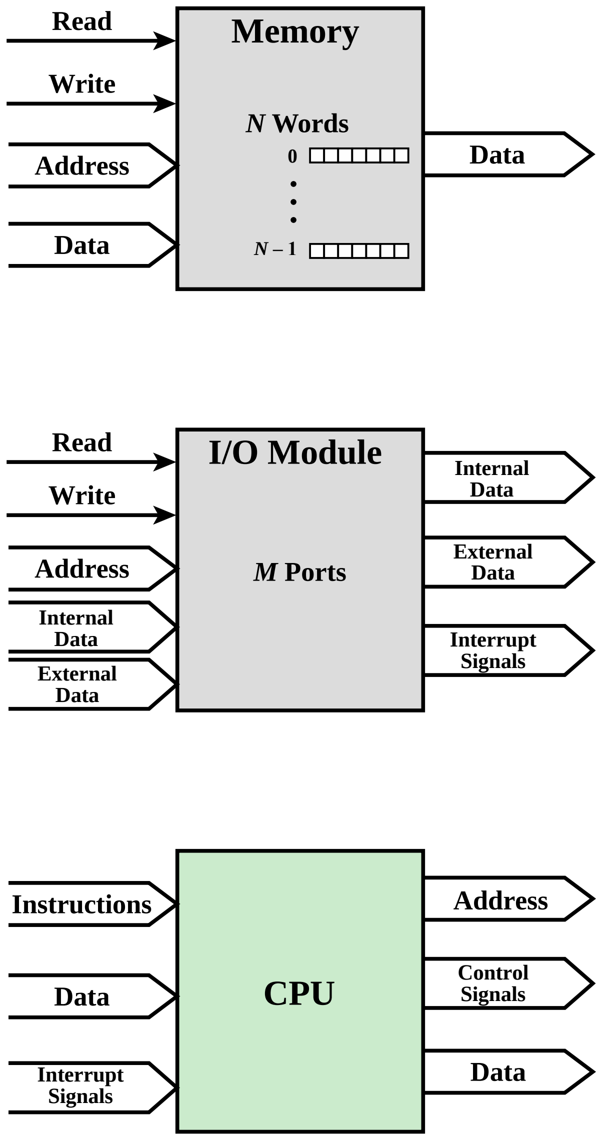


Figure 3.15 Computer Modules

The interconnection structure must support the following types of transfers:

Memory to processor
Processor reads an instruction or a unit of data from memory

Processor to memory
Processor writes a unit of data to memory

I/O to processor
Processor reads data from an I/O device via an I/O module

Processor to I/O
Processor sends data to the I/O device

I/O to or from memory
An I/O module is allowed to exchange data directly with memory without going through the processor using direct memory access

Bus Interconnection

A communication pathway connecting two or more devices

- Key characteristic is that it is a shared transmission medium

Signals transmitted by any one device are available for reception by all other devices attached to the bus

- If two devices transmit during the same time period their signals will overlap and become garbled

Typically consists of multiple communication lines

- Each line is capable of transmitting signals representing binary 1 and binary 0

Computer systems contain a number of different buses that provide pathways between components at various levels of the computer system hierarchy

System bus

- A bus that connects major computer components (processor, memory, I/O)

The most common computer interconnection structures are based on the use of one or more system buses

Data Bus

- Data lines that provide a path for moving data among system modules
- May consist of 32, 64, 128, or more separate lines
- The number of lines is referred to as the *width* of the data bus
- The number of lines determines how many bits can be transferred at a time
- The width of the data bus is a key factor in determining overall system performance

Address Bus

- Used to designate the source or destination of the data on the data bus
  - If the processor wishes to read a word of data from memory it puts the address of the desired word on the address lines
- Width determines the maximum possible memory capacity of the system
- Also used to address I/O ports
  - The higher order bits are used to select a particular module on the bus and the lower order bits select a memory location or I/O port within the module

Control Bus

- Used to control the access and the use of the data and address lines
- Because the data and address lines are shared by all components there must be a means of controlling their use
- Control signals transmit both command and timing information among system modules
- Timing signals indicate the validity of data and address information
- Command signals specify operations to be performed

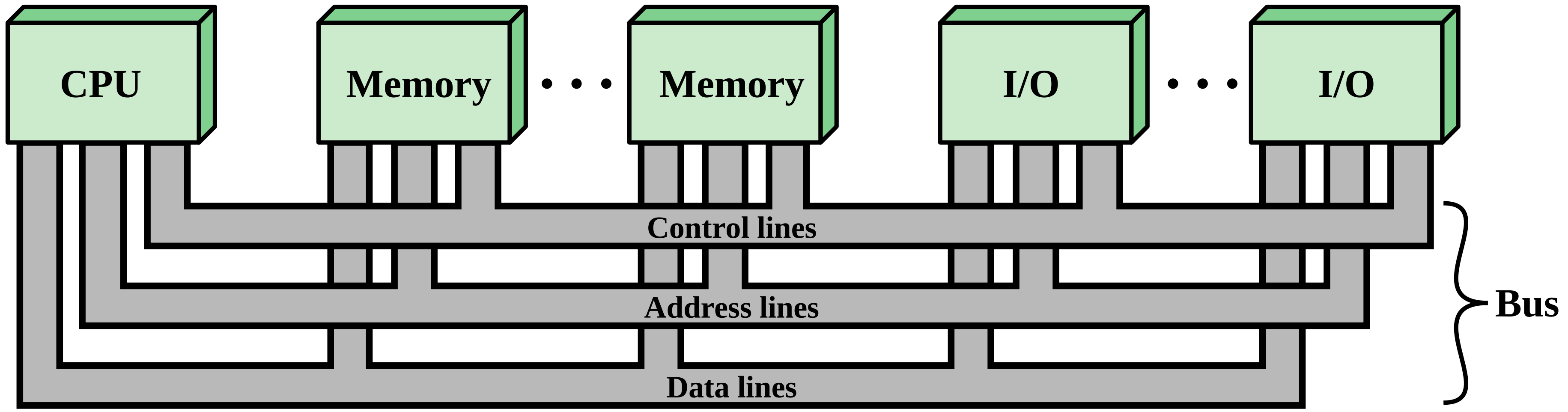


Figure 3.16 Bus Interconnection Scheme

Point-to-Point Interconnect

Principal reason for change was the electrical constraints encountered with increasing the frequency of wide synchronous buses

At higher and higher data rates it becomes increasingly difficult to perform the synchronization and arbitration functions in a timely fashion

A conventional shared bus on the same chip magnified the difficulties of increasing bus data rate and reducing bus latency to keep up with the processors

Has lower latency, higher data rate, and better scalability

Quick Path Interconnect

QPI

- Introduced in 2008
- Multiple direct connections
  - Direct pairwise connections to other components eliminating the need for arbitration found in shared transmission systems
- Layered protocol architecture
  - These processor level interconnects use a layered protocol architecture rather than the simple use of control signals found in shared bus arrangements
- Packetized data transfer
  - Data are sent as a sequence of packets each of which includes control headers and error control codes

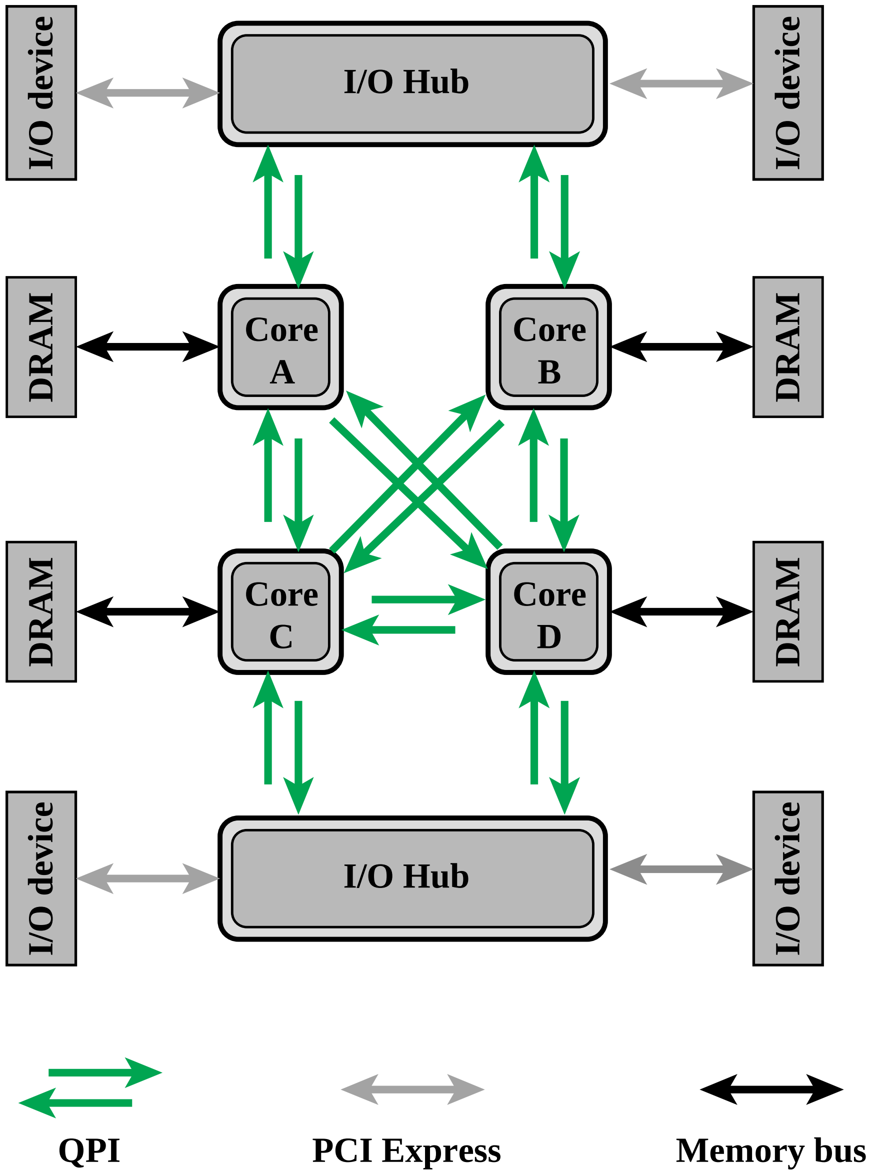


Figure 3.17 Multicore Configuration Using QPI

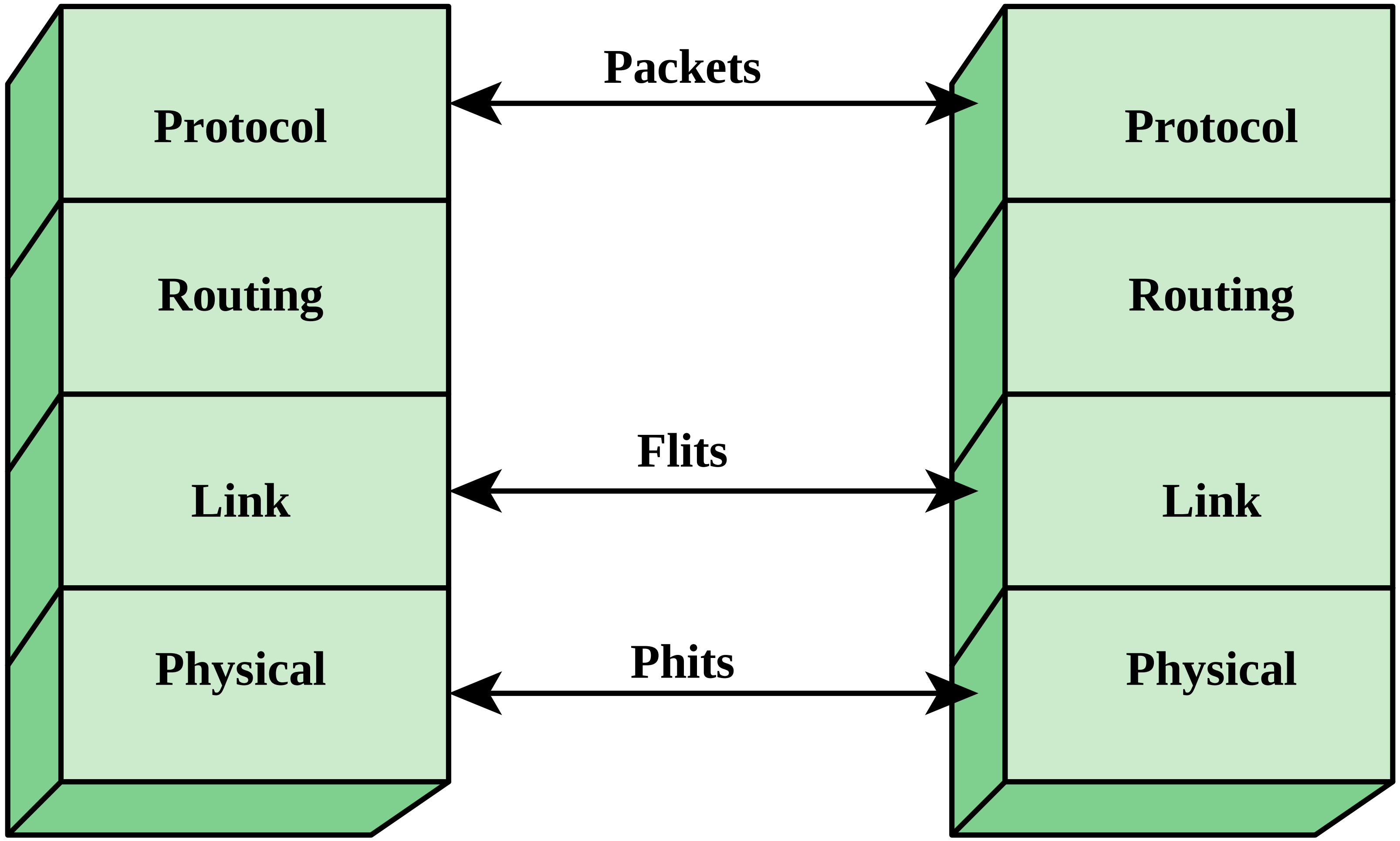


Figure 3.18 QPI Layers

QPI Layers

- **Physical:** Consists of the actual wires carrying the signals, as well as circuitry and logic to support ancillary features required in the transmission and receipt of the 1s and 0s. The unit of transfer at the Physical layer is 20 bits, which is called a Phit (physical unit).
- **Link**: Responsible for reliable transmission and flow control. The Link layer’s unit of transfer is an 80-bit Flit (flow control unit).
- **Routing**: Provides the framework for directing packets through the fabric.
- **Protocol**: The high-level set of rules for exchanging packets of data between devices. A packet is comprised of an integral number of Flits.

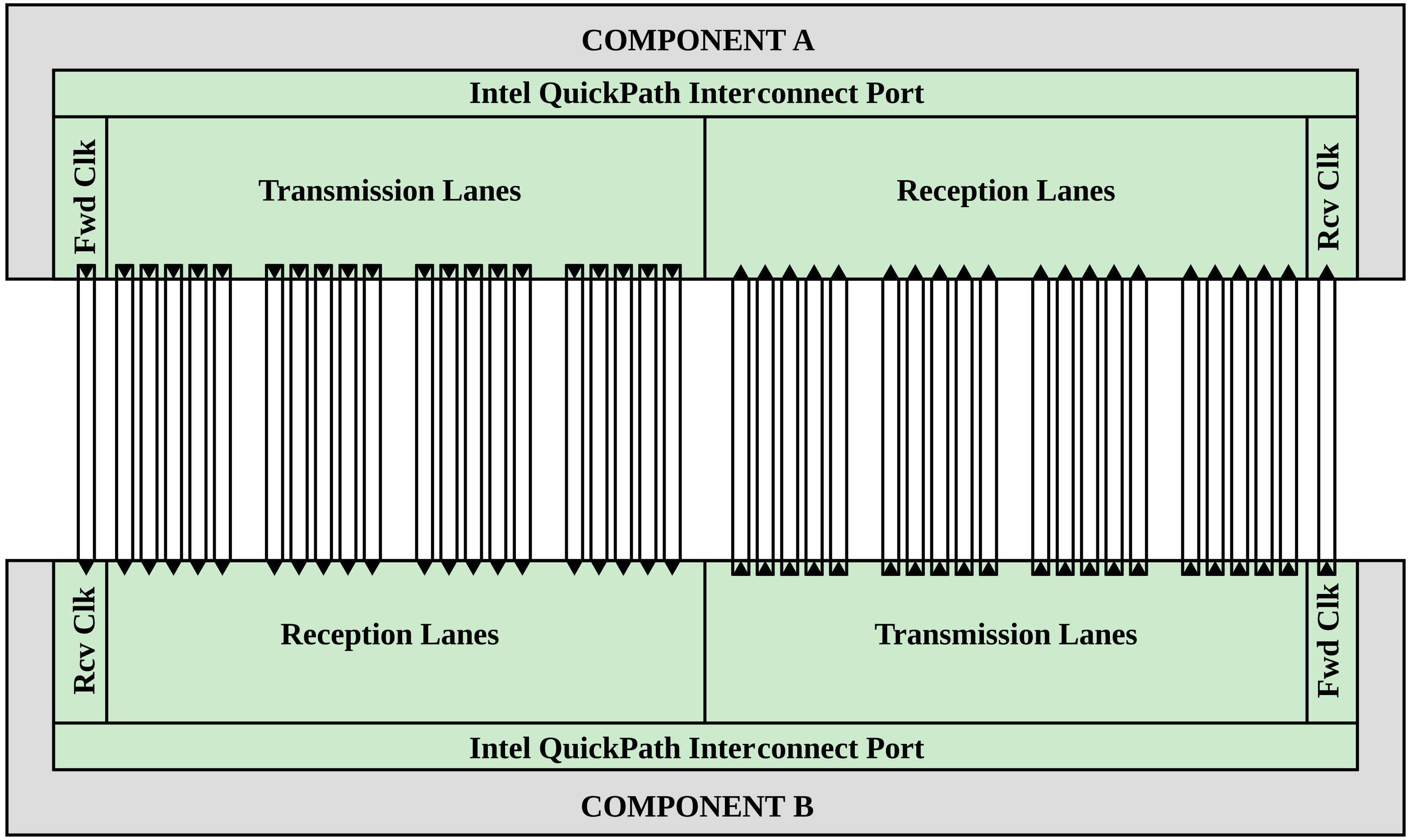


Figure 3.19 Physical Interface of the Intel QPI Interconnect

QPI Link Layer

- Performs two key functions: *flow control* and *error control*
  - Operate on the level of the flit (flow control unit)
  - Each flit consists of a 72-bit message payload and an 8-bit error control code called a *cyclic redundancy check* (CRC)

- Flow control function
  - Needed to ensure that a sending QPI entity does not overwhelm a receiving QPI entity by sending data faster than the receiver can process the data and clear buffers for more incoming data

Error control function
  - Detects and recovers from bit errors, and so isolates higher layers from experiencing bit errors

QPI Routing and Protocol Layers

Routing Layer

- Used to determine the course that a packet will traverse across the available system interconnects
- Defined by firmware and describe the possible paths that a packet can follow

Protocol Layer

- Packet is defined as the unit of transfer
- One key function performed at this level is a cache coherency protocol which deals with making sure that main memory values held in multiple caches are consistent
- A typical data packet payload is a block of data being sent to or from a cache

Peripheral Component Interconnect (PCI)

- A popular high bandwidth, processor independent bus that can function as a mezzanine or peripheral bus
- Delivers better system performance for high speed I/O subsystems
- PCI Special Interest Group (SIG)
  - Created to develop further and maintain the compatibility of the PCI specifications
- PCI Express (PCIe)
  - Point-to-point interconnect scheme intended to replace bus-based schemes such as PCI
  - Key requirement is high capacity to support the needs of higher data rate I/O devices, such as Gigabit Ethernet
  - Another requirement deals with the need to support time dependent data streams

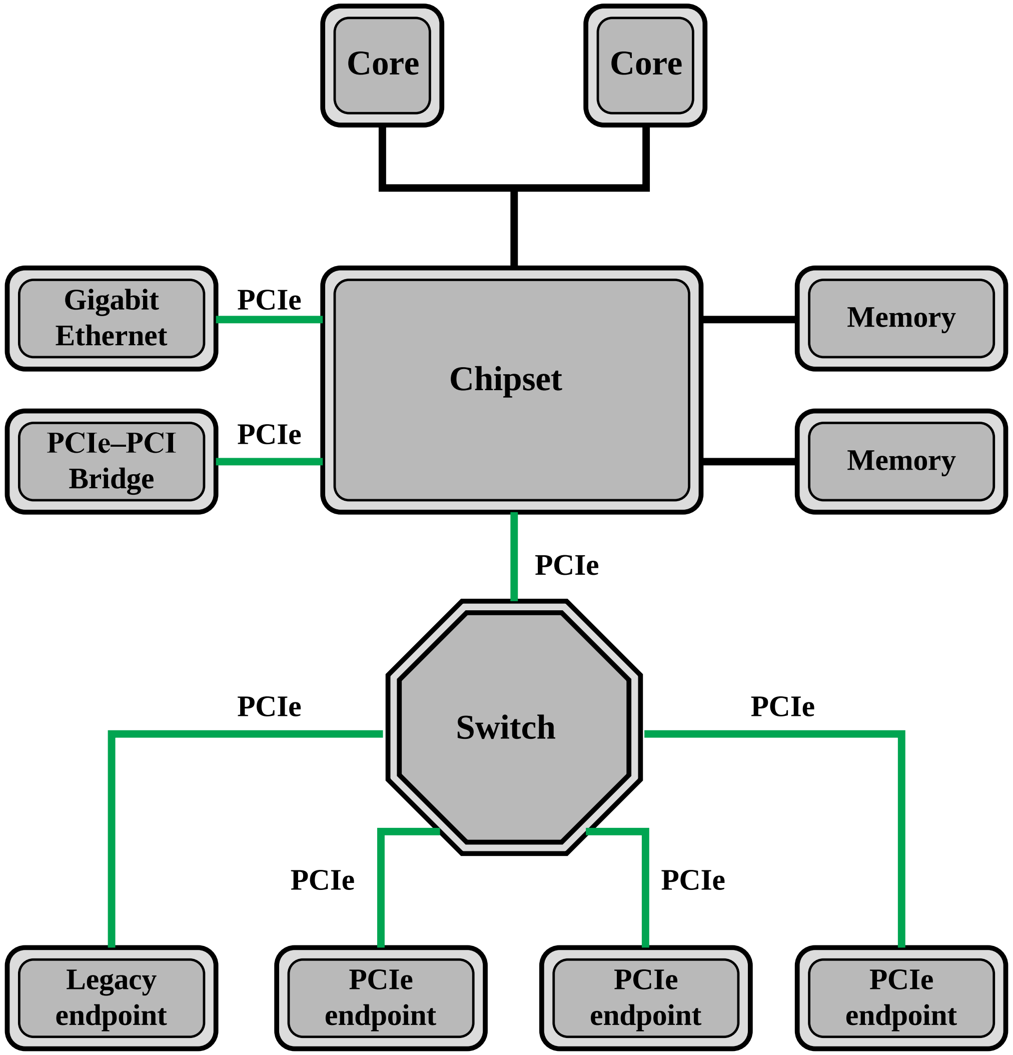


Figure 3.21 Typical Configuration Using PCIe

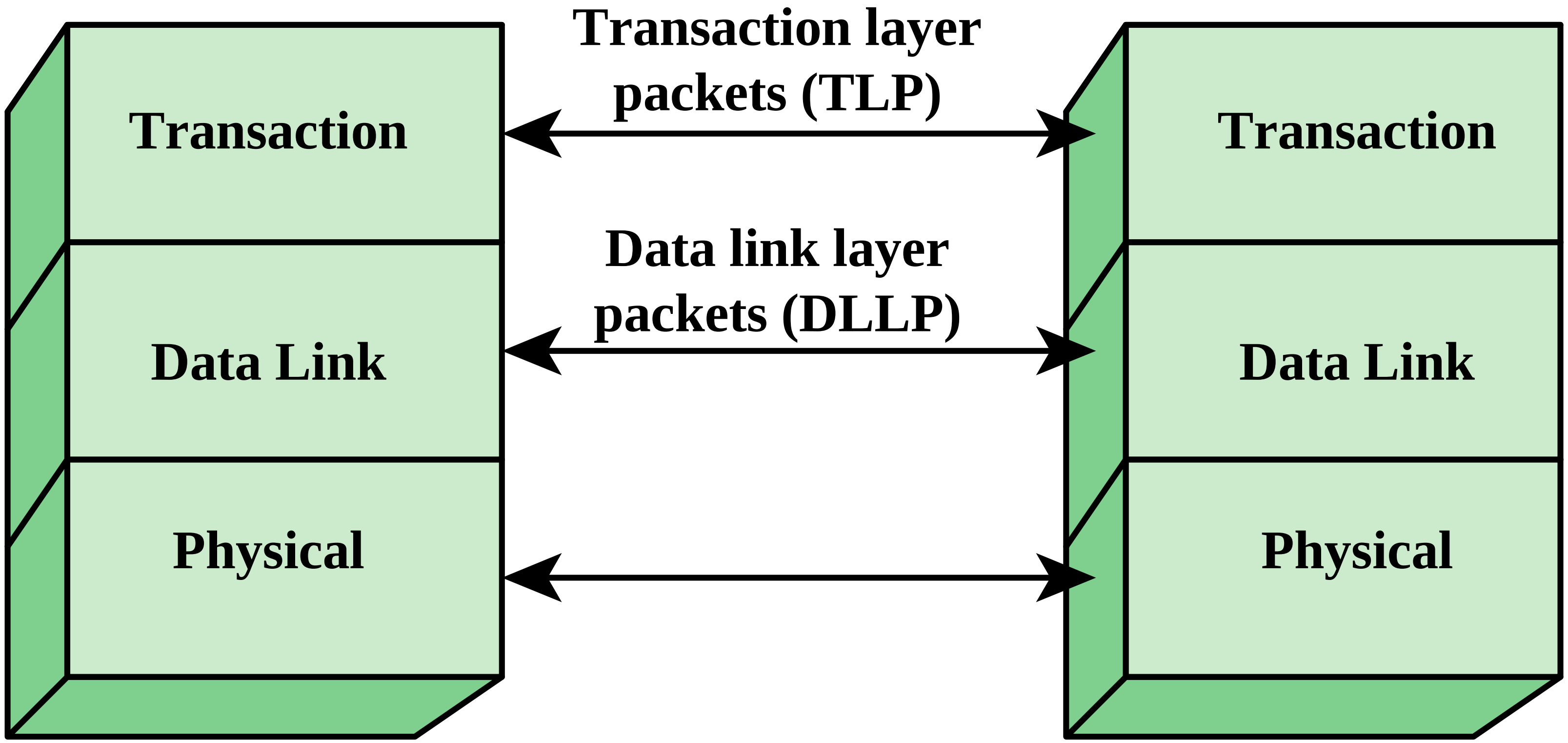

Transaction layer packets (TLP)
Data link layer packets (DLLP)
Transaction
Data Link
Physical
Transaction
Data Link
Physical

Figure 3.22 PCIe Protocol Layers

PCIe Layers

- **Physical:** Consists of the actual wires carrying the signals, as well as circuitry and logic to support ancillary features required in the transmission and receipt of the 1s and 0s.

- **Data Link**: Responsible for reliable transmission and flow control. Data packets generated and consumed by the DLL are called Data Link Layer Packets (DLLPs).

- **Transaction:** Generates and consumes data packets used to implement load/store data transfer mechanisms and also manages the flow control of those packets between the two components on a link. Data packets generated and consumed by the TL are called Transaction Layer Packets (TLPs).

PCIe Transaction Layer (TL)

- Receives read and write requests from the software above the TL and creates request packets for transmission to a destination via the link layer
- Most transactions use a *split transaction* technique
  - A request packet is sent out by a source PCIe device which then waits for a response called a *completion* packet
- TL messages and some write transactions are posted transactions (meaning that no response is expected)
- TL packet format supports 32-bit memory addressing and extended 64-bit memory addressing

The TL supports four address spaces:

- Memory
  - The memory space includes system main memory and PCIe I/O devices
  - Certain ranges of memory addresses map into I/O devices
- I/O
  - This address space is used for legacy PCI devices, with reserved address ranges used to address legacy I/O devices
- Configuration
  - This address space enables the TL to read/write configuration registers associated with I/O devices
- Message
  - This address space is for control signals related to interrupts, error handling, and power management

PCIe TLP Transaction Types

Address Space	TLP Type	Purpose
Memory	Memory Read Request	Transfer data to or from a location in the system memory map.
	Memory Read Lock Request	
	Memory Write Request	
I/O	I/O Read Request	Transfer data to or from a location in the system I/O Write Request memory map for legacy devices.
	I/O Write Request	
Configuration	Config Type 0 Read Request	Transfer data to or from a location in the configuration space of a PCIe device.
	Config Type 0 Write Request	
	Config Type 1 Read Request	
	Config Type 1 Write Request	
Message	Message Request	Provides in-band messaging and event reporting.
	Message Request with Data	
Memory, I/O, Configuration	Completion	Returned for certain requests.
	Completion with Data	
	Completion Locked	
	Completion Locked with Data	

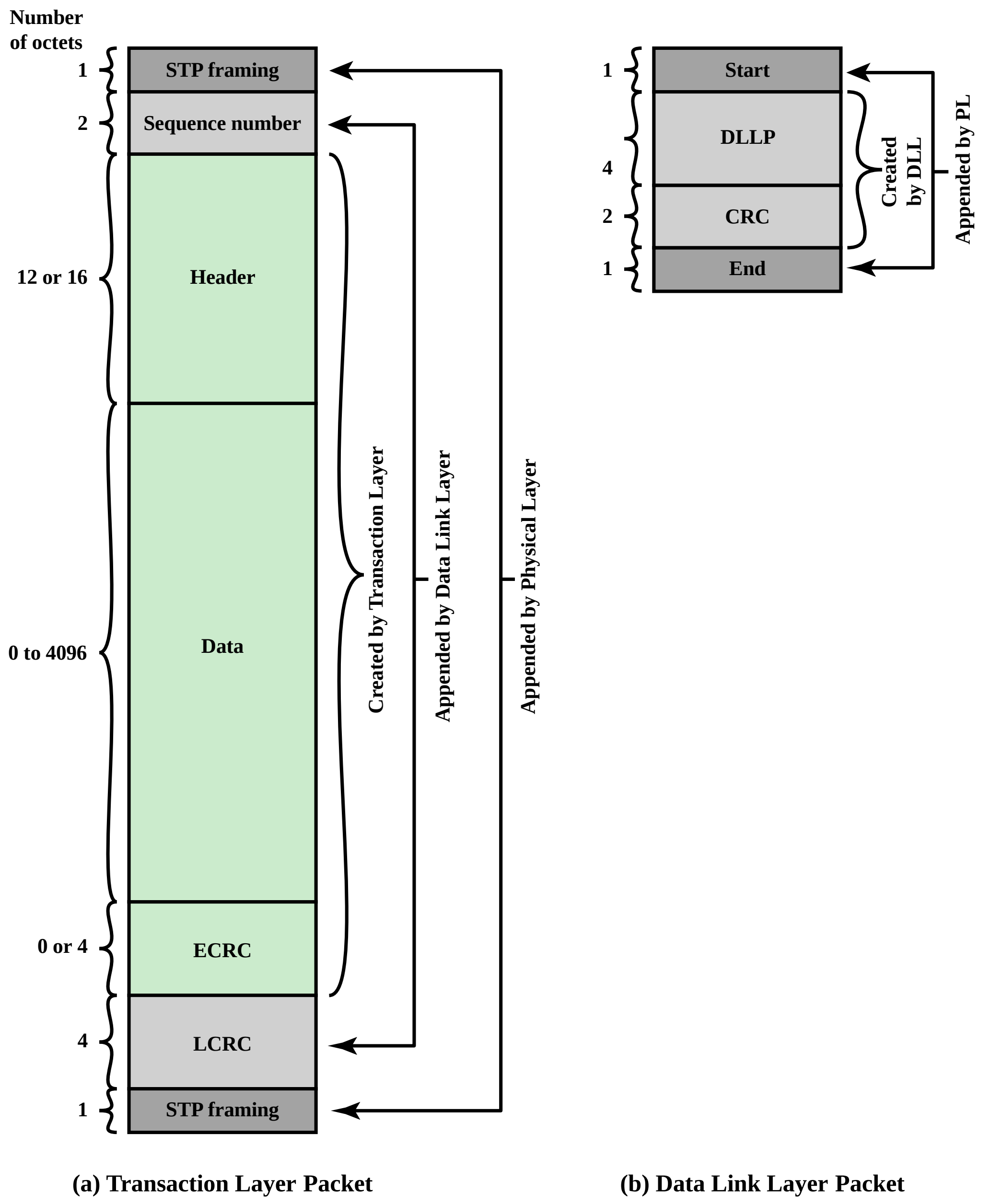


Figure 3.25 PCIe Protocol Data Unit Format