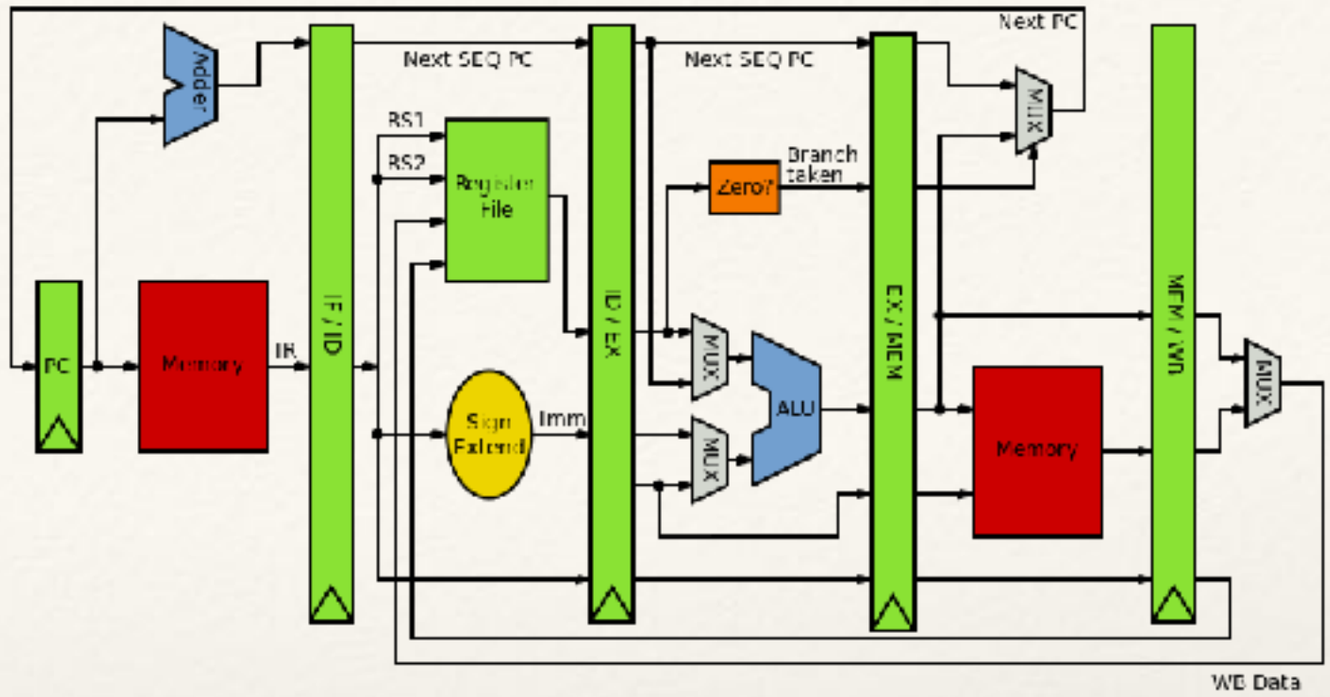


COMP2120 Computer Organization

A Top-Level View of Computer Function And Interconnection

Chapter 3

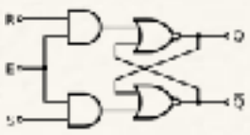
Dr. Ronald H.Y. Chung

THE UNIVERSITY OF HONG KONG

DEPARTMENT OF COMPUTER SCIENCE

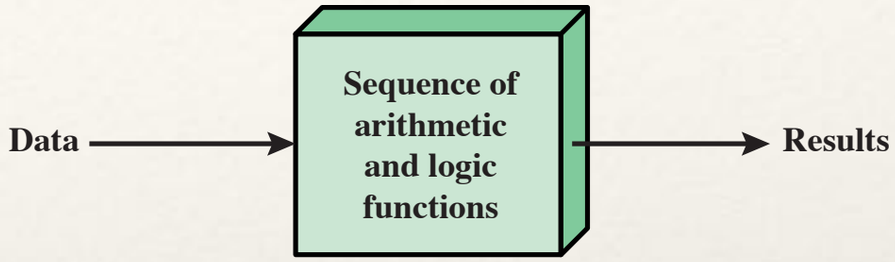
Computer Components

- 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 result of the process of connecting the various components in the desired configuration

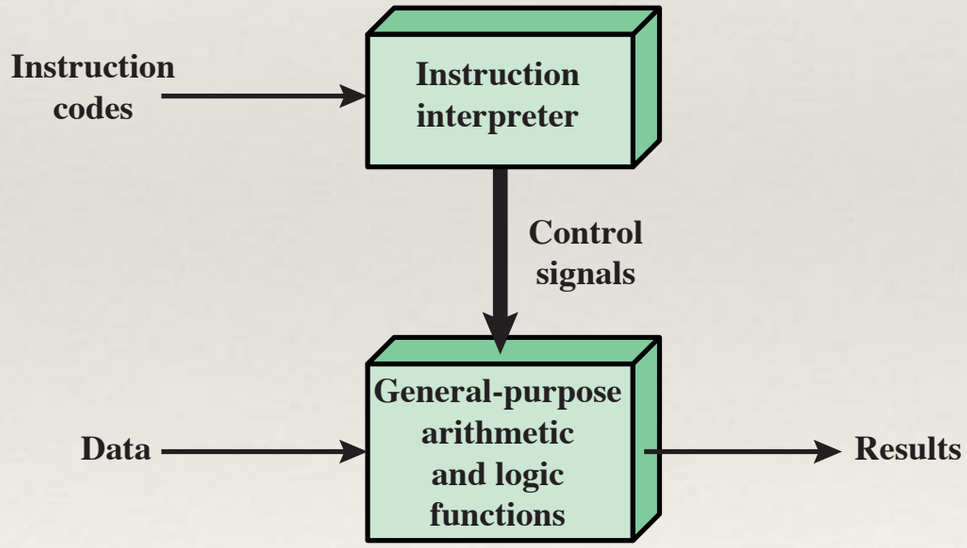


Software Approach

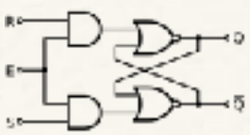
- Software
  - A sequence of codes or instructions
  - Part of the hardware interprets each instruction and generates control signals
  - Provide a new sequence of codes for each new program instead of rewiring the hardware
- Major Components
  - CPU
    - Instruction interpreter
    - Module of general-purpose 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
    - A place to temporarily store both instructions and data



(a) Programming in hardware

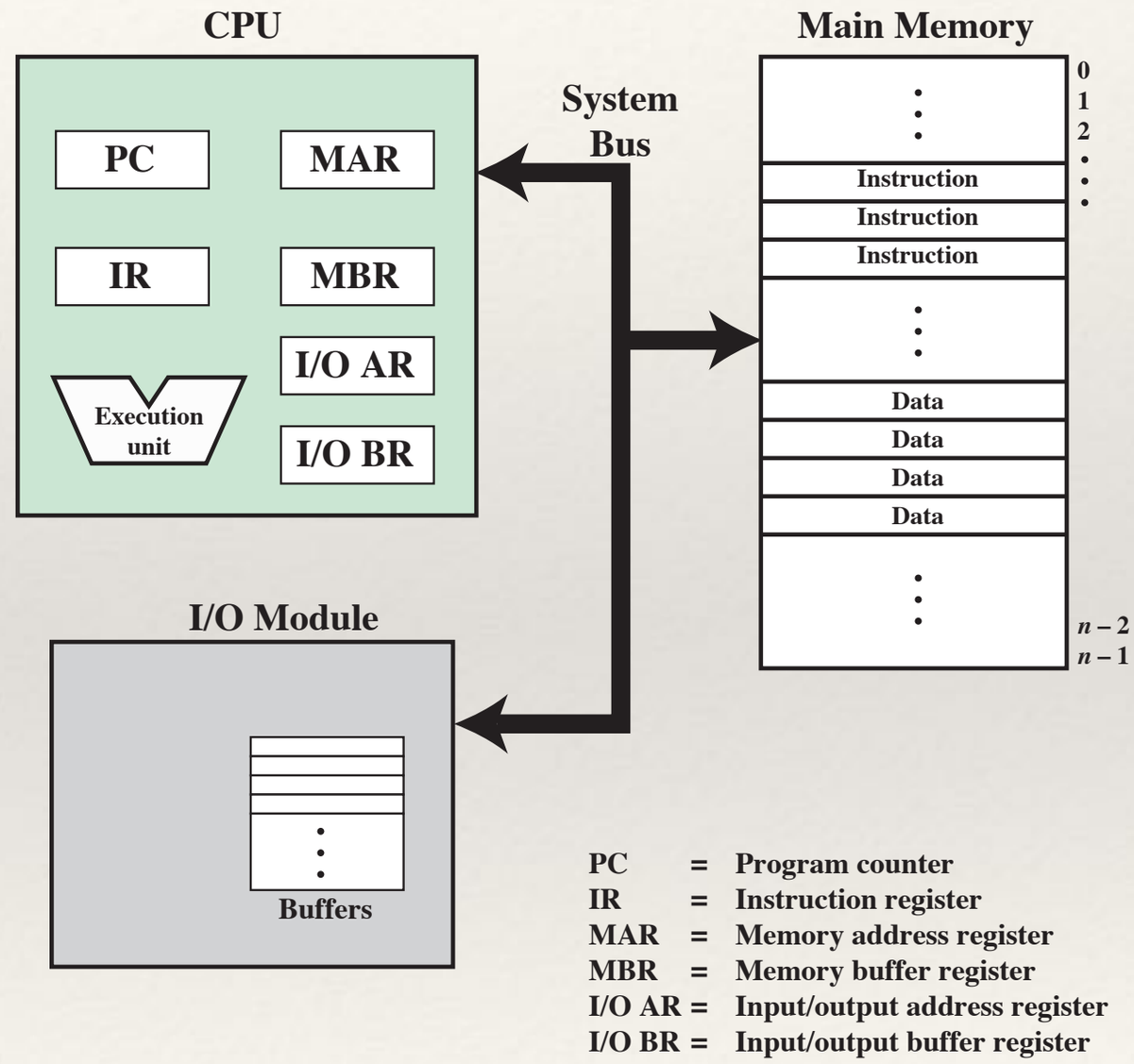


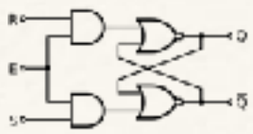
(b) Programming in software



Memory

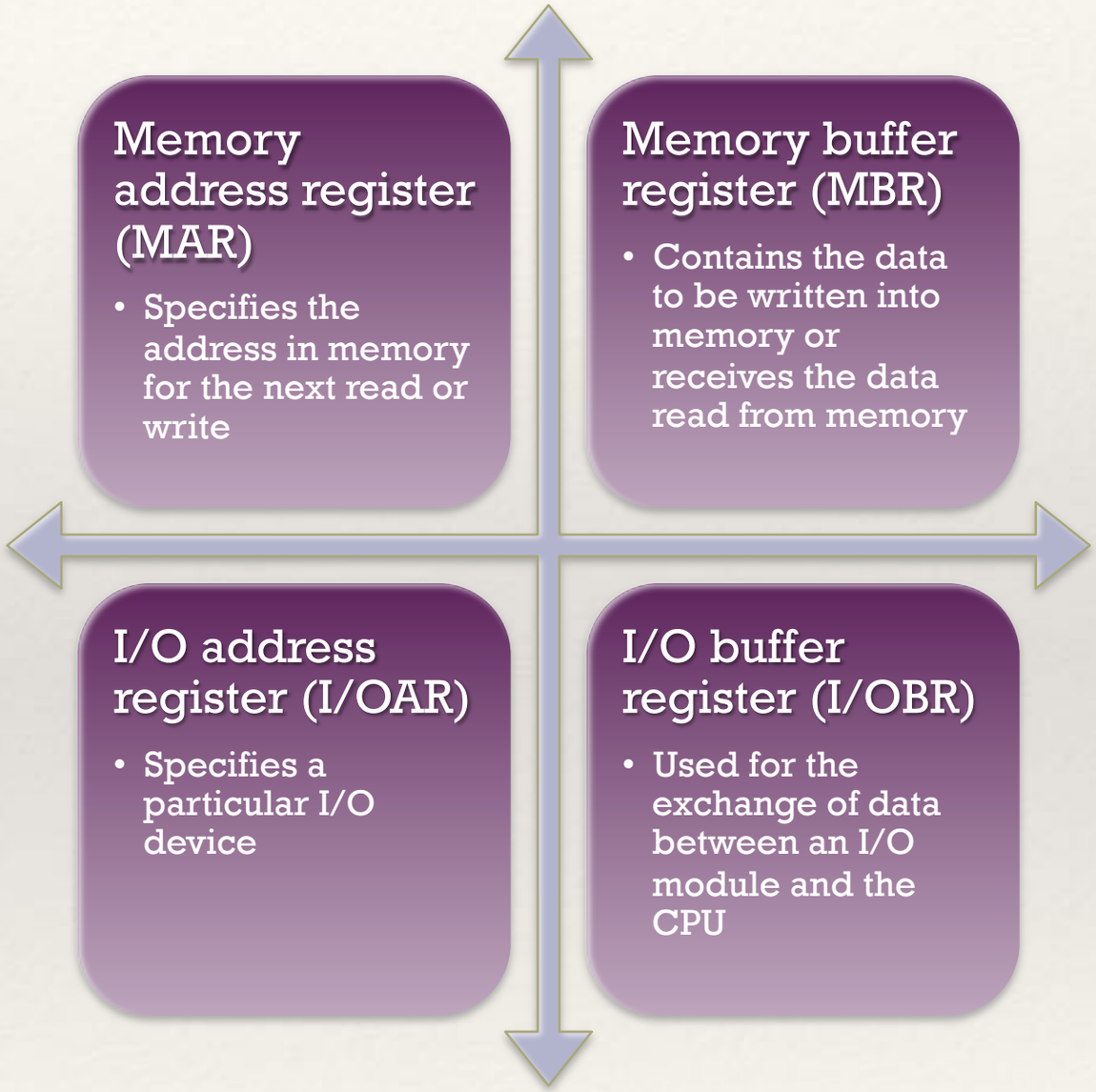
- A memory module consists of a set of locations, defined by sequentially numbered addresses.
- Each location contains a binary number that can be interpreted as either an instruction or data.
- An I/O module transfers data from external devices to CPU and memory, and vice versa.
- It contains internal buffers for temporarily holding these data until they can be sent on.

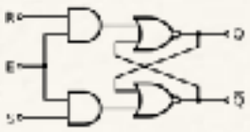




Memory (Cont'd)

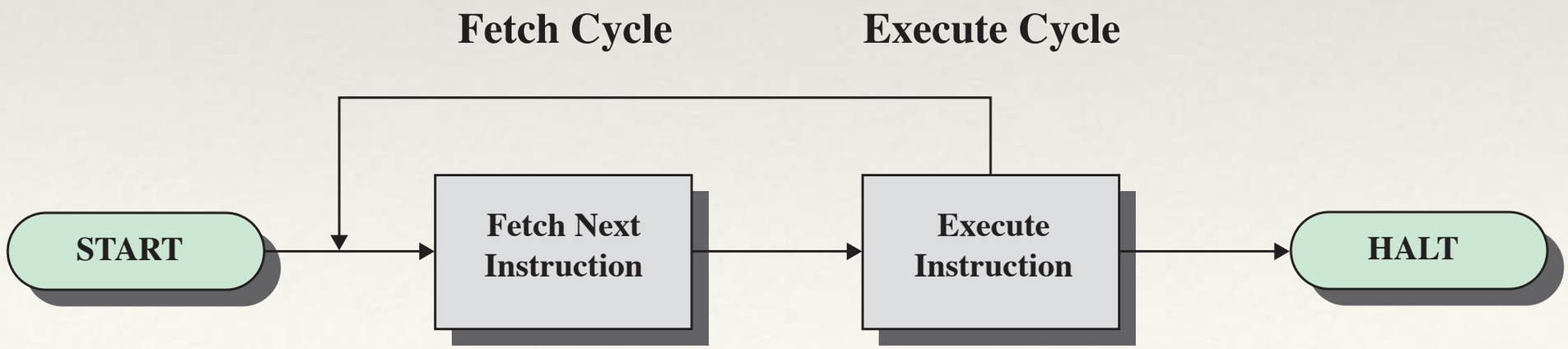
- The CPU exchanges data with memory
  - It typically makes use of two internal (to the CPU) registers:
    - A memory address register (**MAR**), which specifies the address in memory for the next read or write
    - A memory buffer/data register (**MBR**/**MDR**), which contains the data to be written into memory or receives the data read from memory
  - Likewise, an I/O address register (**I/O AR**) specifies a particular I/O device. An I/O buffer register (**I/O BR**) is used for the exchange of data between an I/O module and the CPU

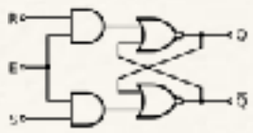




Instruction Fetch and Execute

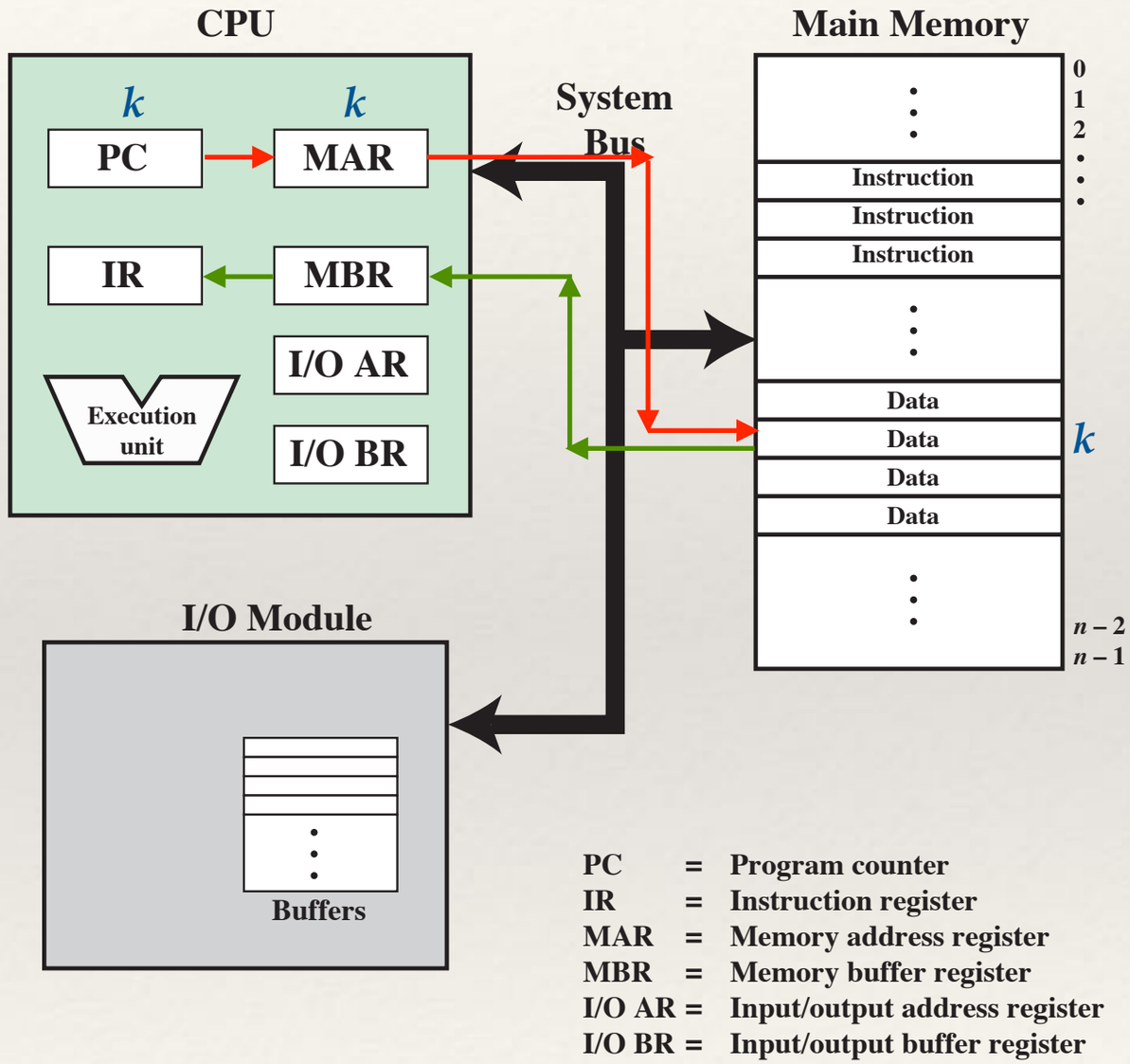
- The basic function performed by a computer is execution of a program
- Instruction processing consists of two steps
  - ***Fetch Cycle***: The processor reads (fetches) instructions from memory one at a time
  - ***Execute Cycle***: Execution of each instruction
- Program execution consists of repeating the process of instruction fetch and instruction execution
- The processing required for a single instruction is called an ***Instruction Cycle***

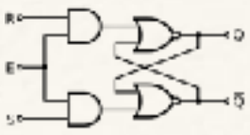




Fetch Cycle

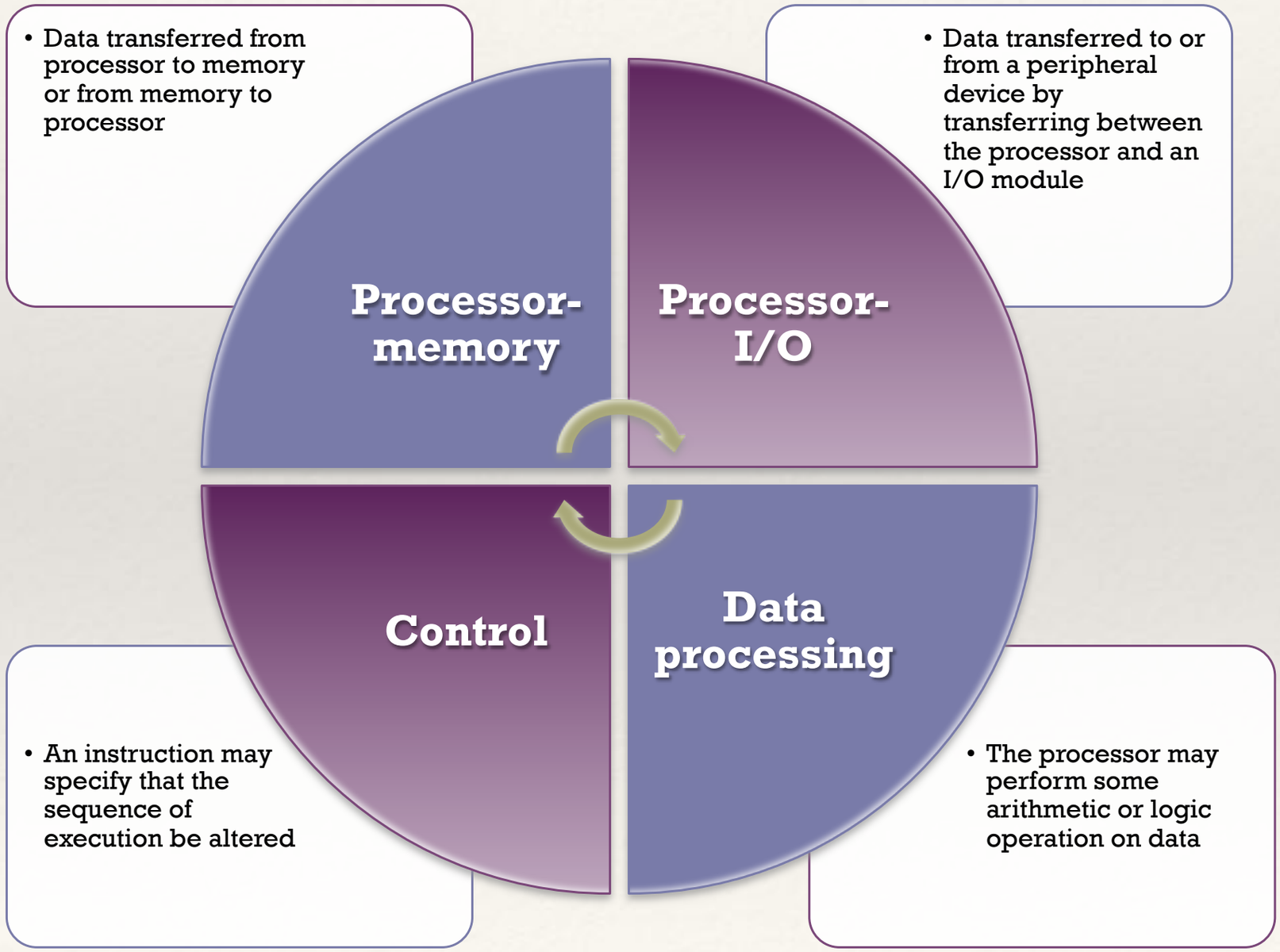
- At the beginning of each instruction cycle the processor fetches an instruction from memory
- The program counter (**PC**) holds the address of the instruction to be fetched next
- The processor increments the PC after each instruction fetch so that it will fetch the next instruction in sequence
-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

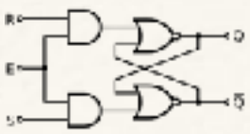




Execute Cycle

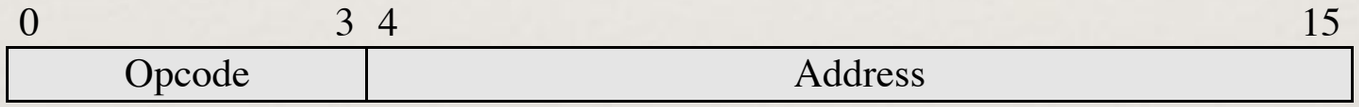
- Instruction's execution involve a combination of the following actions



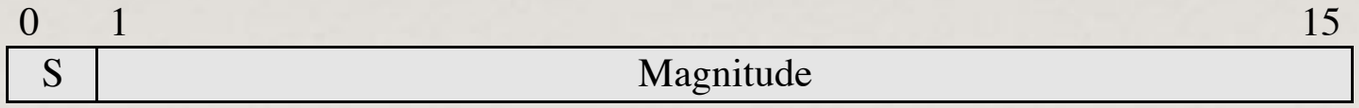


Example Machine

- The processor contains a single data register called an accumulator (AC)
- Both instructions and data are 16 bits long
- Instruction format
  - 4 bits for the opcode
  - Up to 2^{12} = 4096 (4K) words of memory can be directly addressed



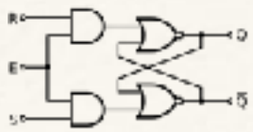
(a) Instruction format



(b) Integer format

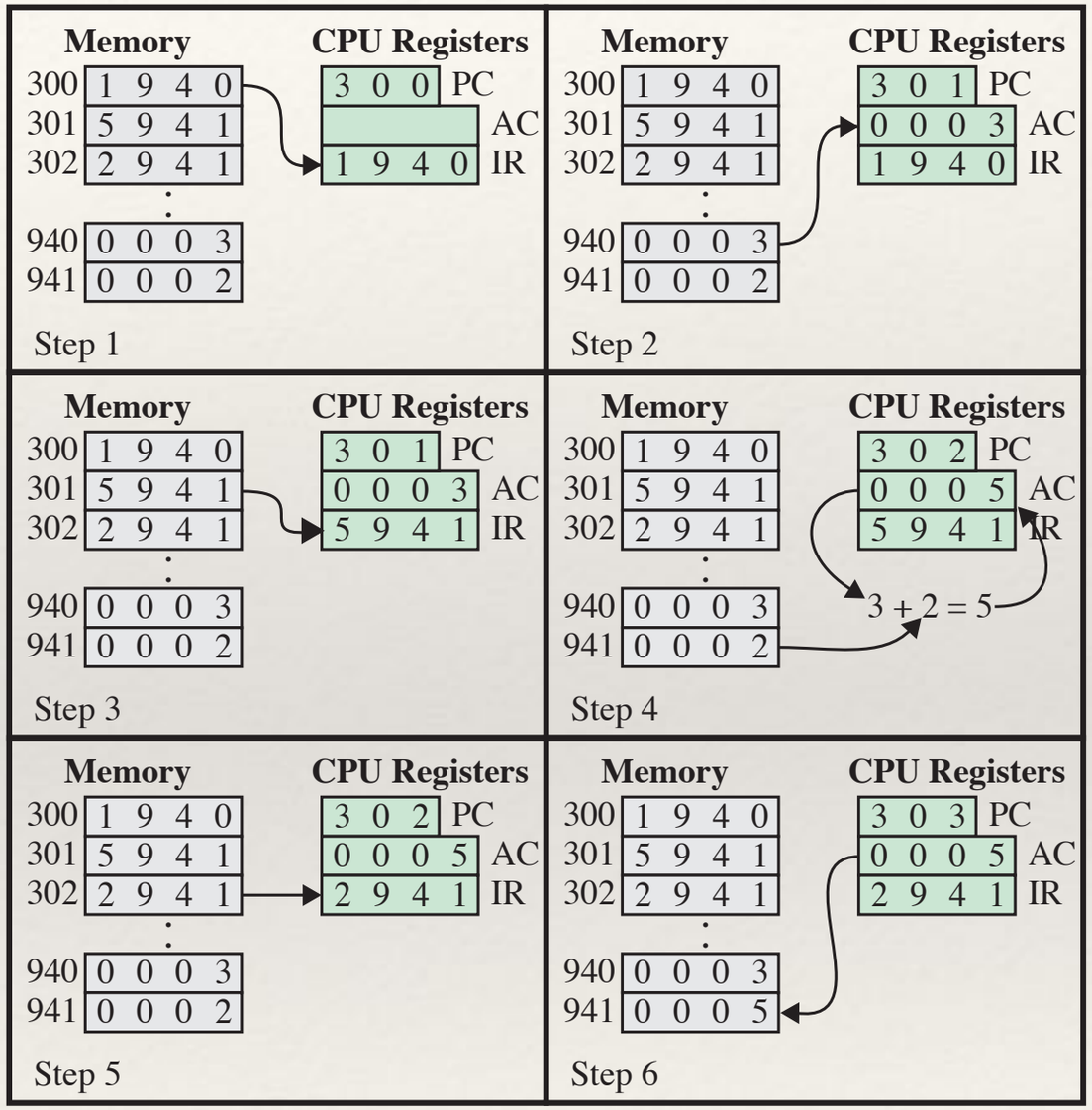
Program Counter (PC) = Address of instruction
Instruction Register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers



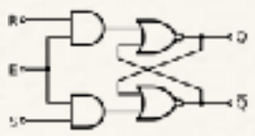
Program Execution Example

1. The PC contains 300, the address of the first instruction. This instruction (the value 1940 in hexadecimal) is loaded into the instruction register IR, and the PC is incremented. Note that this process involves the use of a memory address register and a memory buffer register. For simplicity, these intermediate registers are ignored.
2. The first 4 bits (first hexadecimal digit) in the IR indicate that the AC is to be loaded. The remaining 12 bits (three hexadecimal digits) specify the address (940) from which data are to be loaded.
3. The next instruction (5941) is fetched from location 301, and the PC is incremented.
4. The old contents of the AC and the contents of location 941 are added, and the result is stored in the AC.
5. The next instruction (2941) is fetched from location 302, and the PC is incremented.
6. The contents of the AC are stored in location 941.



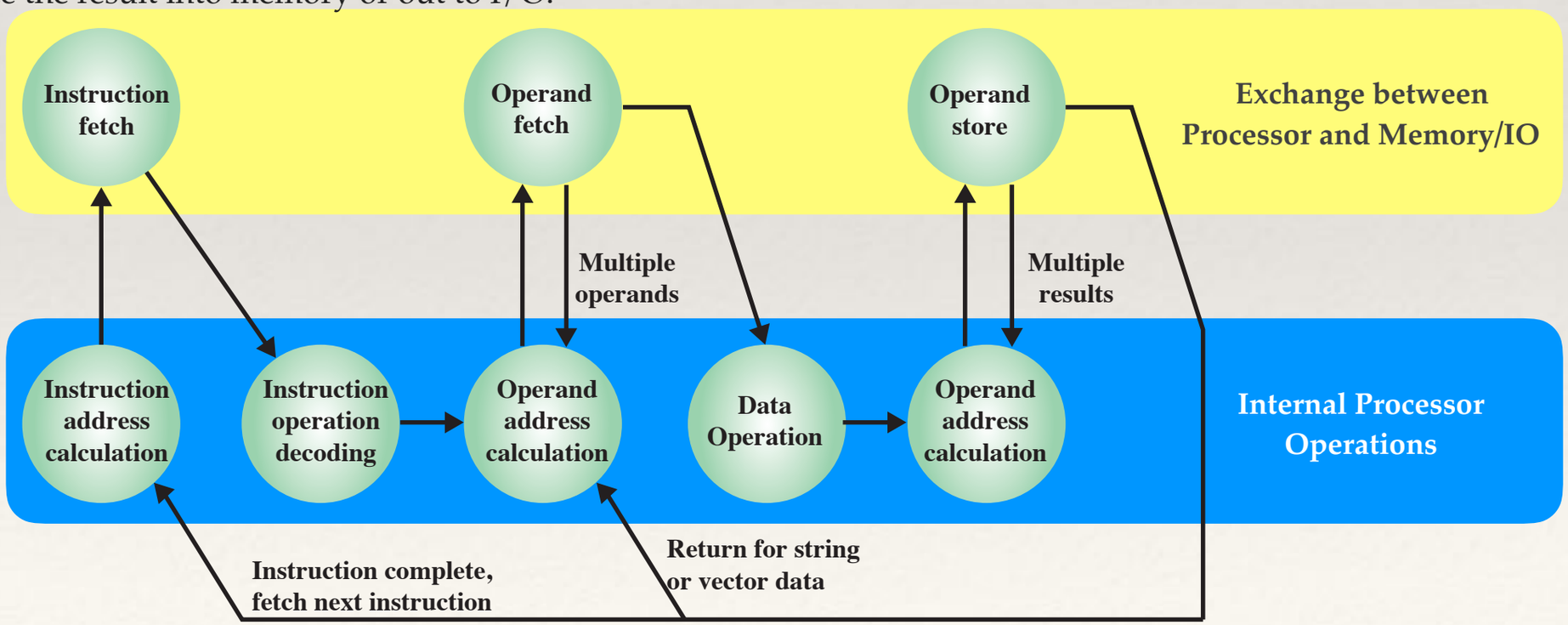
0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

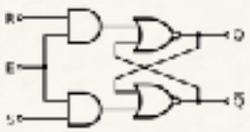
Partial list of opcodes



Instruction Cycle States

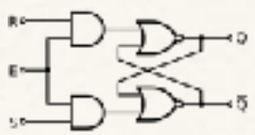
- **Instruction address calculation (iac)**: Determine the address of the next instruction to be executed. Usually, this involves adding a fixed number to the address of the previous instruction.
- **Instruction fetch (if)**: Read instruction from its memory location into the processor.
- **Instruction operation decoding (iod)**: Analyze instruction to determine type of operation to be performed and operand(s) to be used.
- **Operand address calculation (oac)**: If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand.
- **Operand fetch (of)**: Fetch the operand from memory or read it in from I/O.
- **Data operation (do)**: Perform the operation indicated in the instruction.
- **Operand store (os)**: Write the result into memory or out to I/O.





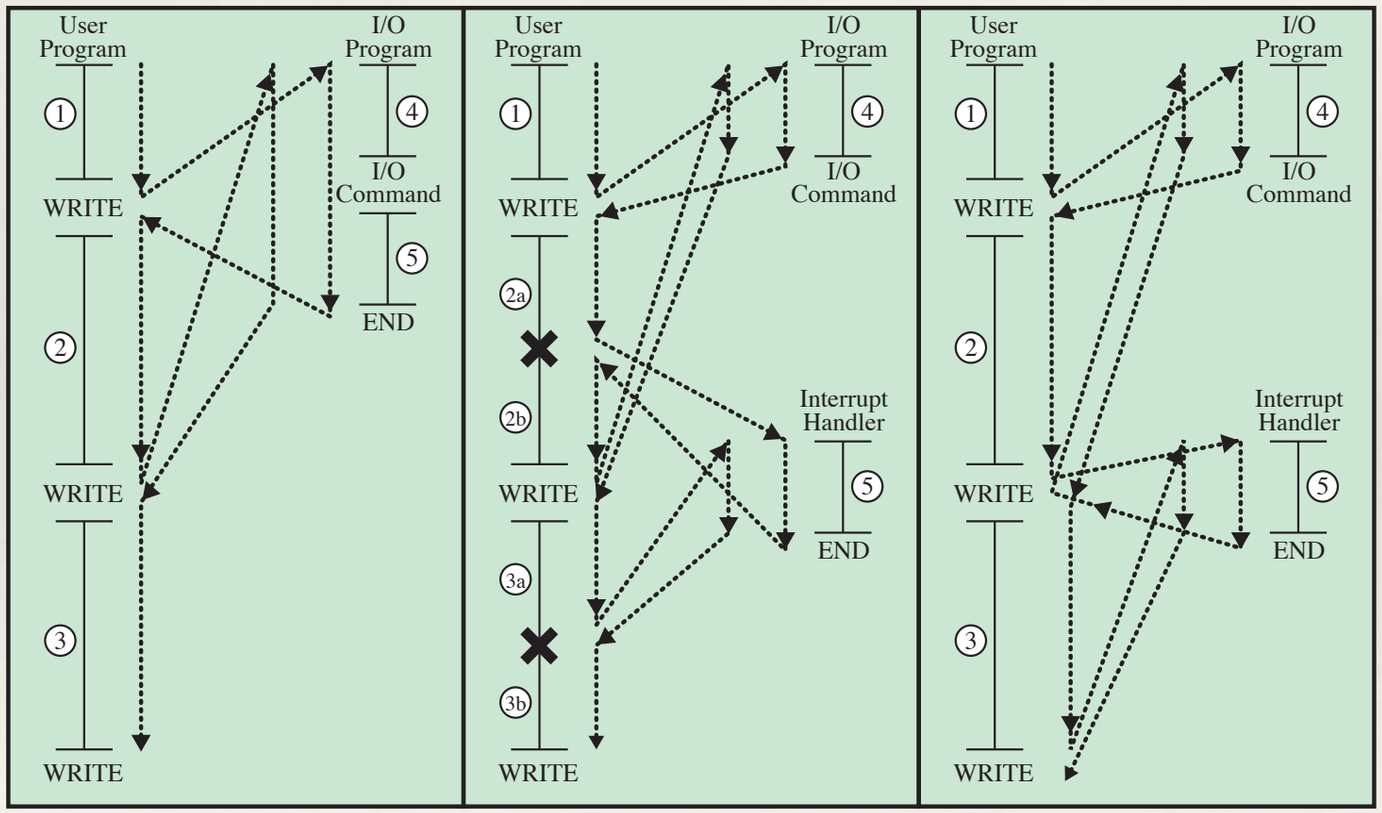
Interrupts

- Virtually all computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal processing of the processor
- Interrupts are provided primarily as a way to improve processing efficiency
- 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 exe- cute 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.



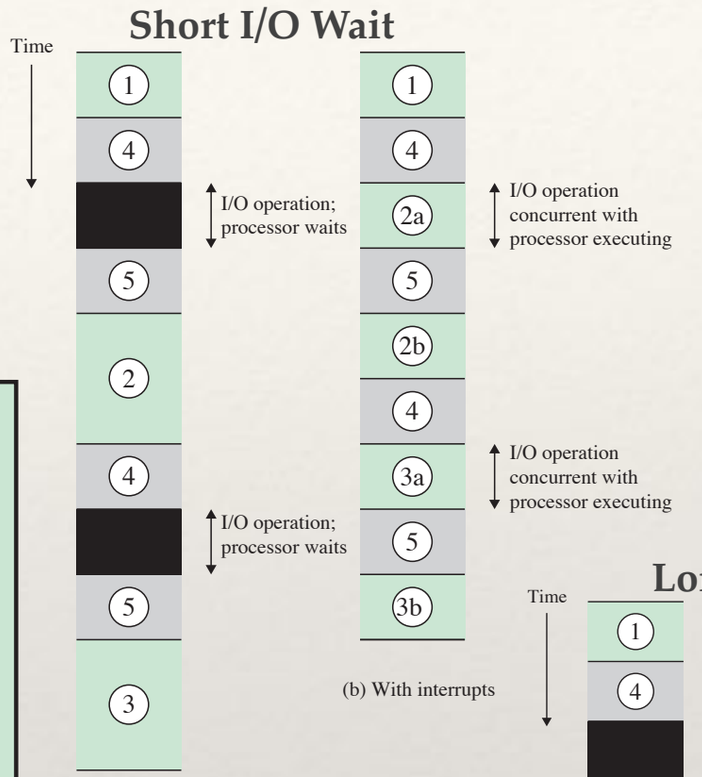
Program Flow of Control w/wo Interrupts

- Code segments 1, 2, and 3 refer to sequences of instructions that do not involve I/O
- Segment 4 refers to the preparation of actual I/O operation
- Segment 5 refers to the finalisation of the I/O operation (e.g. setting a flag)

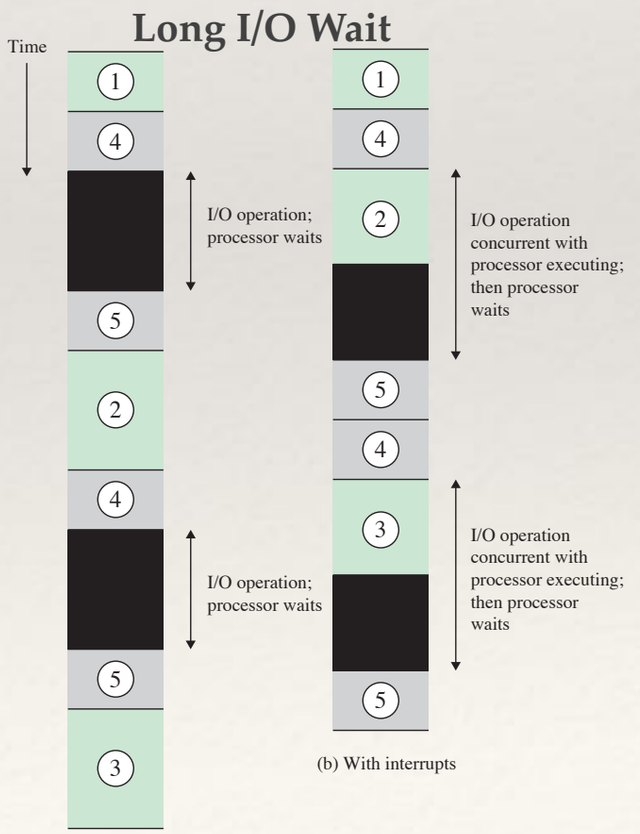


(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

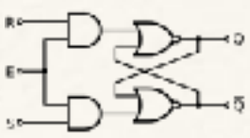
✖ = interrupt occurs during course of execution of user program



(a) Without interrupts (b) With interrupts

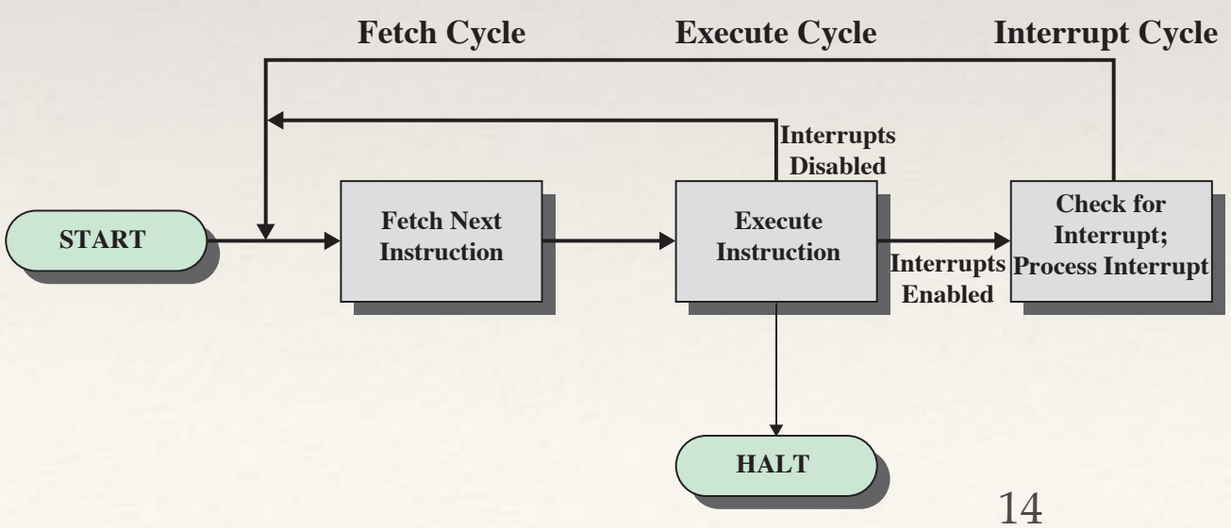


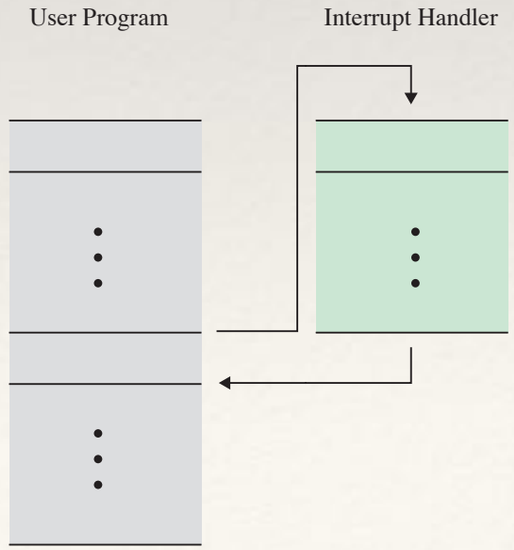
(a) Without interrupts (b) With interrupts

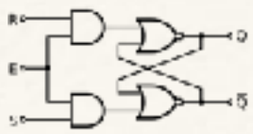


Instruction Cycle with Interrupts

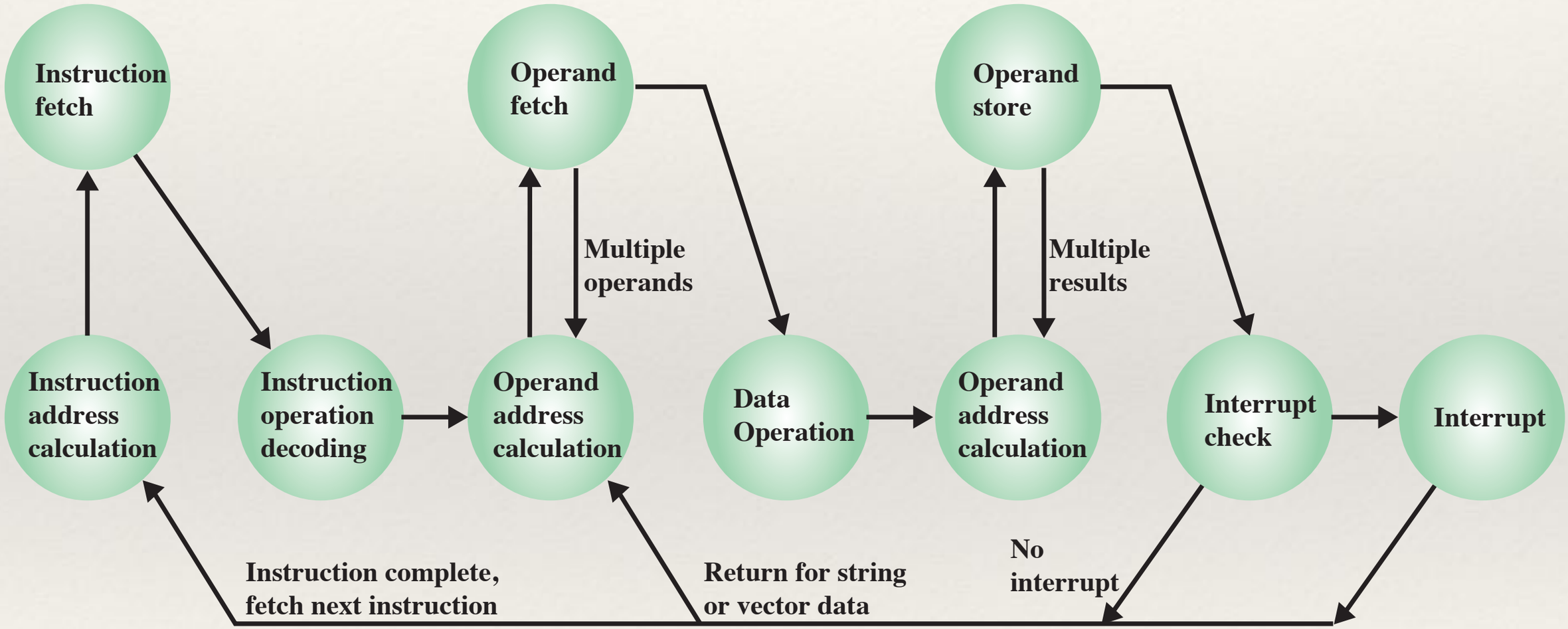
- To accommodate interrupts, an interrupt cycle is added to the instruction
- In the interrupt cycle, the processor checks to see if any interrupts have occurred, indicated by the presence of an interrupt signal
  - If no interrupts are pending, the processor proceeds to the fetch cycle and fetches the next instruction of the current program.
  - If an interrupt is pending, the processor does the following:
    - It suspends execution of the current program being executed and saves its context. This means saving the address of the next instruction to be executed (current contents of the program counter) and any other data relevant to the processor's current activity.
    - It sets the program counter to the starting address of an interrupt handler routine.
    - The processor now proceeds to the fetch cycle and fetches the first instruction in the interrupt handler program, which will service the interrupt. The interrupt handler program is generally part of the operating system. Typically, this program determines the nature of the interrupt and performs whatever actions are needed.
    - When the interrupt handler routine is completed, the processor can resume execution of the user program at the point of interruption.

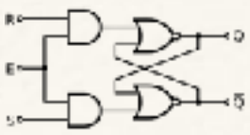






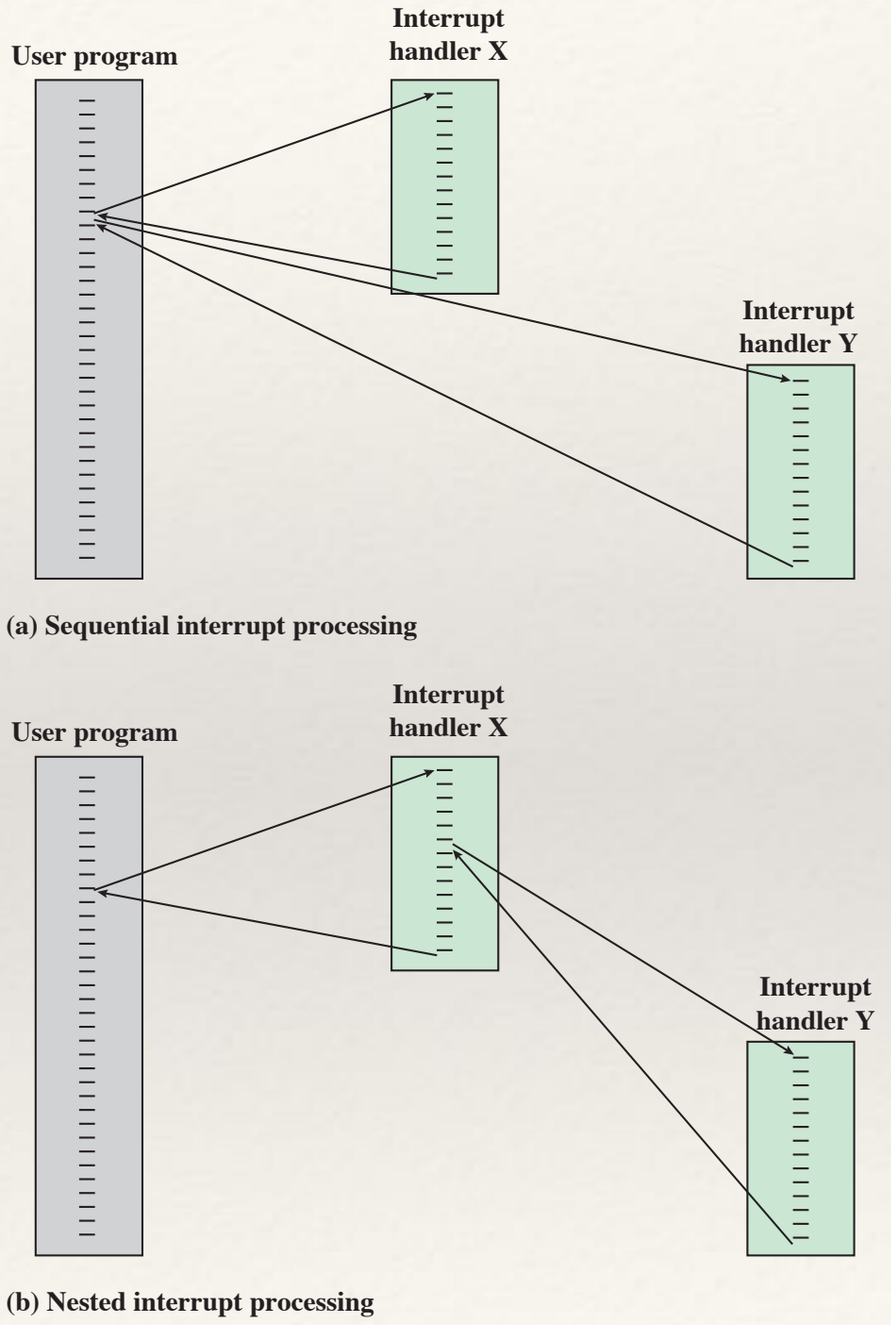
Instruction Cycle State Diagram, With Interrupts

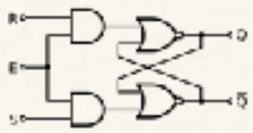




Multiple Interrupts

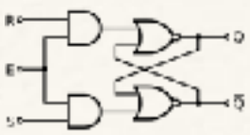
- Two approaches can be taken to dealing with multiple interrupts
  - Disable interrupts while an interrupt is being processed
    - A disabled interrupt simply means that the processor can and will ignore that interrupt request signal.
    - If an interrupt occurs during this time, it generally remains pending and will be checked by the processor after the processor has enabled interrupts.
    - After the interrupt handler routine completes, interrupts are enabled before resuming the user program, and the processor checks to see if additional interrupts have occurred.
    - Interrupts are handled in strict sequential order
    - The drawback to the preceding approach is that it does not take into account relative priority or time-critical needs
  - A second approach is to define priorities for interrupts and to allow an interrupt of higher priority to cause a lower-priority interrupt handler to be itself interrupted





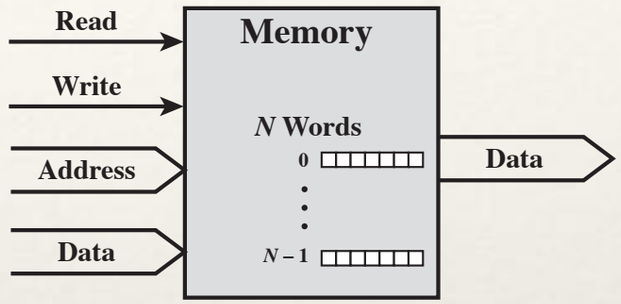
I/O Function

- 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)

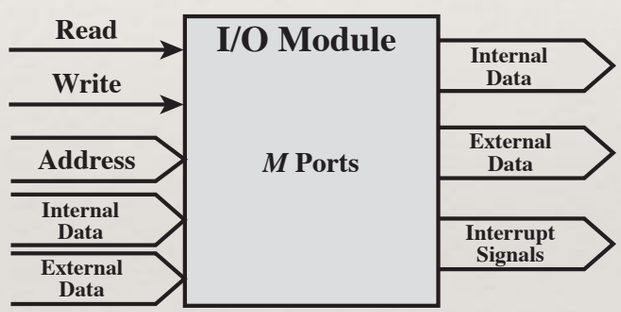


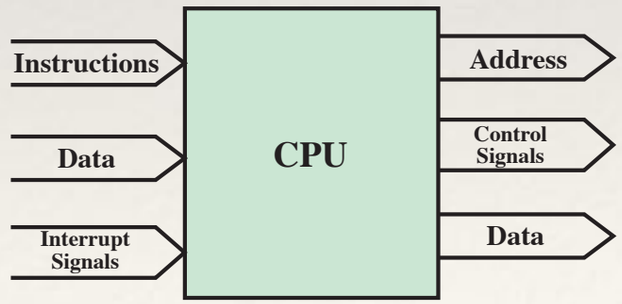
Interconnection Structures

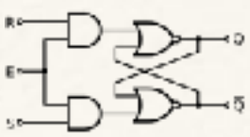
- A computer is a network of basic modules:
  - Processor
  - Memory
  - I/O
- The collection of paths connecting the various modules is called the interconnection structure



- Major forms of input and output for each module type:
  - Memory: Typically, a memory module will consist of *N* words of equal length. Each word is assigned a unique numerical address (0, 1, …, *N* - 1). The nature of the operation is indicated by read and write control signals. The location for the operation is specified by an address.
  - I/O module: I/O is functionally similar to memory. There are two operations, read and write. Further, an I/O module may control more than one external device. We can refer to each of the interfaces to an external device as a port and give each a unique address (e.g., 0, 1, …, *M* - 1). In addition, there are external data paths for the input and output of data with an external device. Finally, an I/O module may be able to send interrupt signals to the processor.
  - Processor: The processor reads in instructions and data, writes out data after processing, and uses control signals to control the overall operation of the system. It also receives interrupt signals.

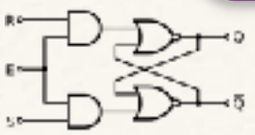






Types of Data Transfer

Memory to processor	Processor to memory	I/O to processor	Processor to I/O	I/O to or from memory
Processor reads an instruction or a unit of data from memory	Processor writes a unit of data to memory	Processor reads data from an I/O device via an I/O module	Processor sends data to the I/O device	An I/O module is allowed to exchange data directly with memory without going through the processor using direct memory access



Interconnection Bus (1)

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

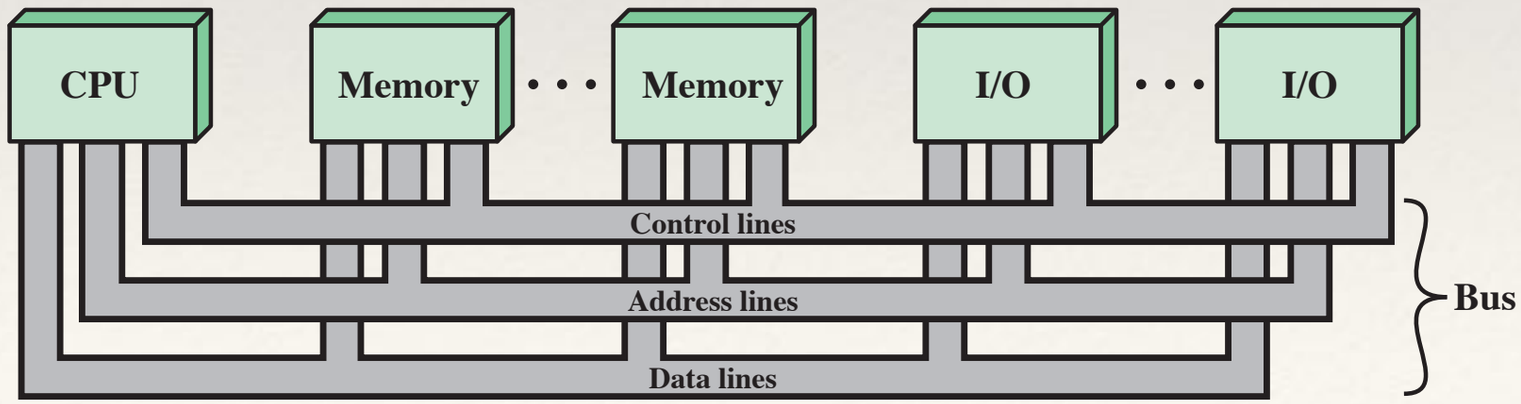
- Each line is capable of transmitting signals representing binary 1 and binary 0

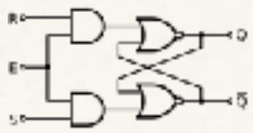
Computer systems contain a number of different buses that provide pathways between components at various levels of the computer system hierarchy

System bus

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

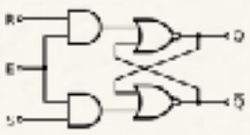
The most common computer interconnection structures are based on the use of one or more system buses





Interconnection Bus (2)

- ***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



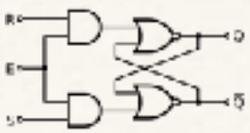
Point-to-Point Interconnect

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

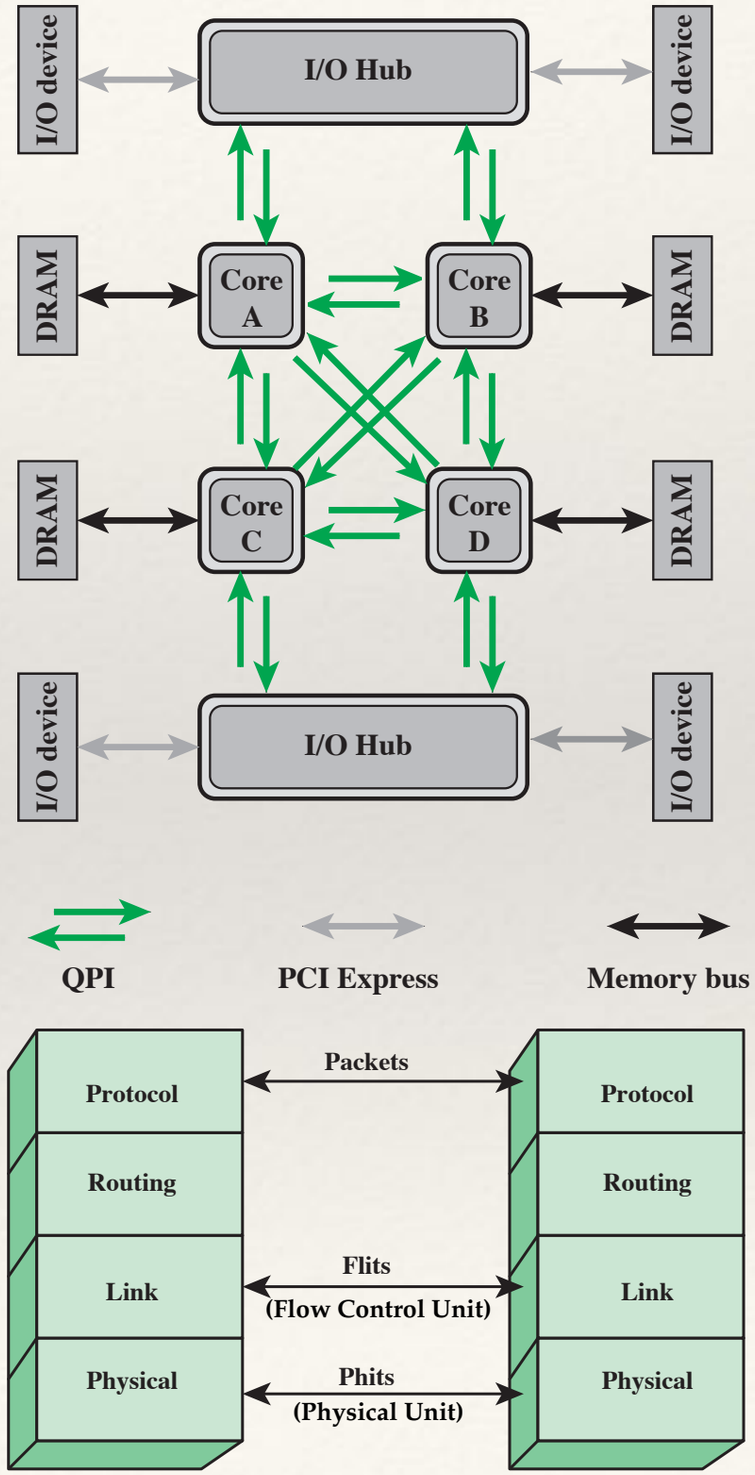
A conventional shared bus on the same chip magnified the difficulties of increasing bus data rate and reducing bus latency to keep up with the processors

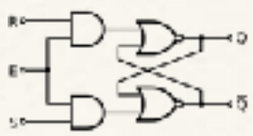
Has lower latency, higher data rate, and better scalability



Quick Path Interconnect (QPI)

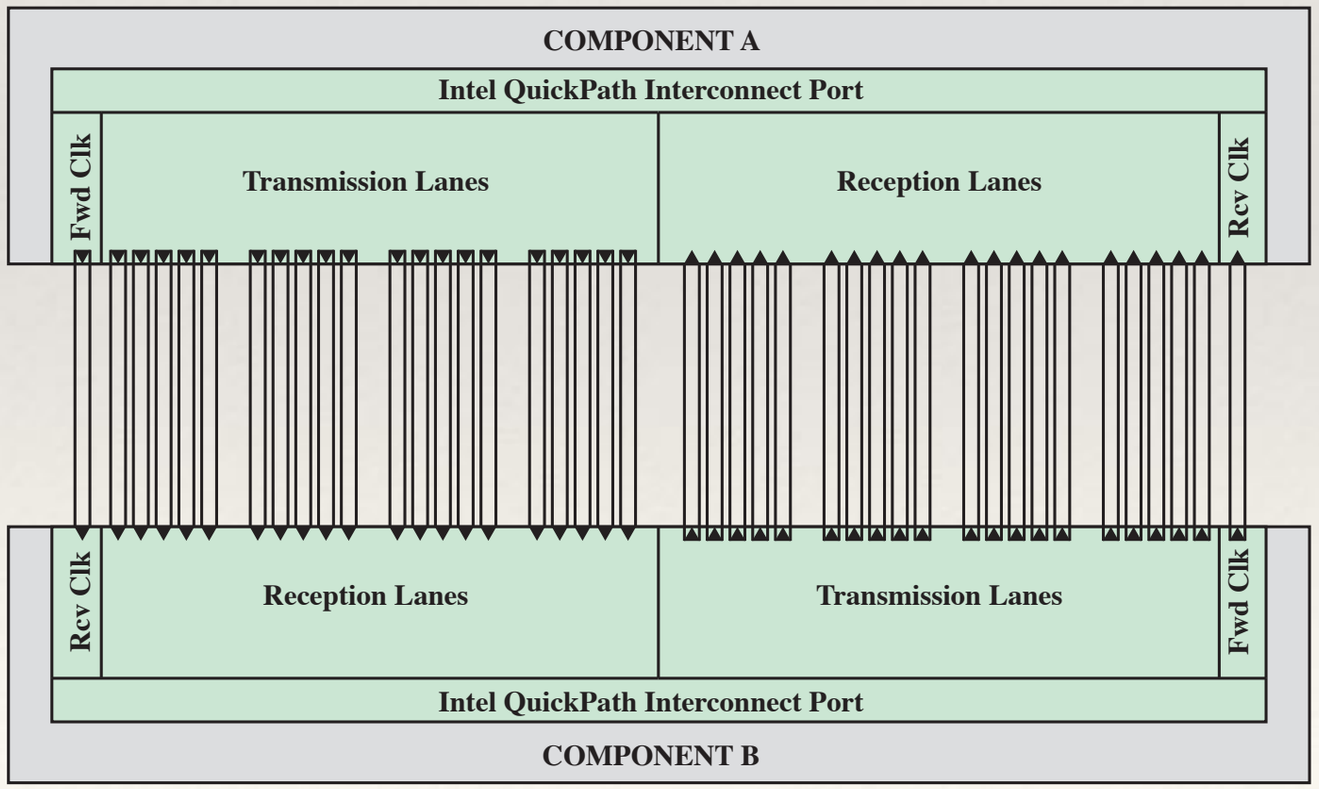
- Introduced in 2008
- Multiple direct connections
  - Direct pairwise connections to other components eliminating the need for arbitration found in shared transmission systems
- Layered protocol architecture
  - These processor level interconnects use a layered protocol architecture rather than the 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





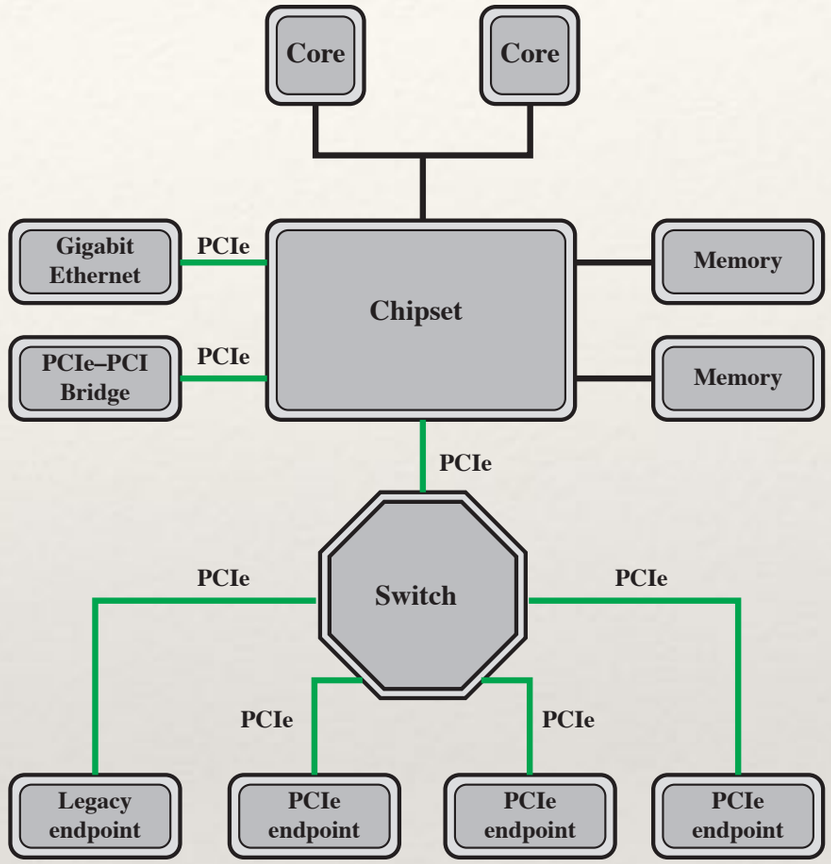
Physical Interface of the Intel QPI Interconnect

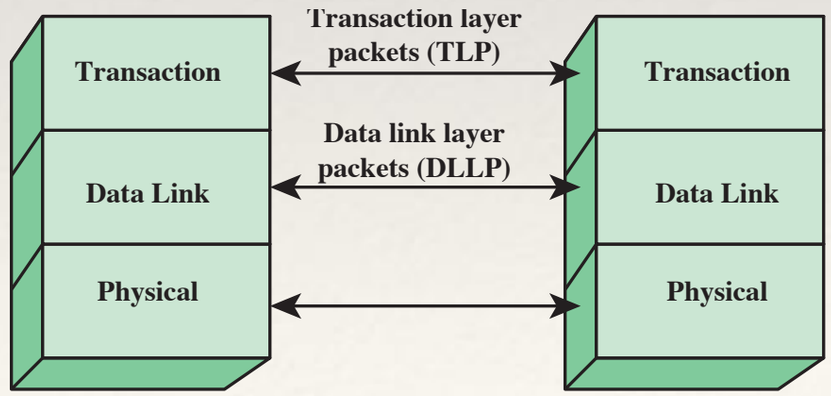
- The QPI port consists of 84 individual links
  - Each data path consists of a pair of wires that transmits data one bit at a time;
  - The pair is referred to as a lane.
  - There are 20 data lanes in each direction (transmit and receive), plus a clock lane in each direction.
  - Thus, QPI is capable of transmitting 20 bits in parallel in each direction. The 20-bit unit is referred to as a phit (physical unit).
  - Typical signaling speeds of the link in current products calls for operation at 6.4 GT/s (transfers per second). At 20 bits per transfer, that adds up to 16 GB/s, and since QPI links involve dedicated bidirectional pairs, the total capacity is 32 GB/s.
  - The lanes in each direction are grouped into four quadrants of 5 lanes each.

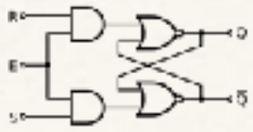


Peripheral Component Interconnect (PCI)

- 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







Chapter 3 - Summary

- Chapter 3 - A Top-Level View of Computer Function and Interconnection
  - Computer components
  - Computer function
    - Instruction fetch and execute
    - Interrupts
    - I/O function
  - Interconnection structures
  - Bus interconnection
  - Point-to-point interconnect
    - QPI
    - PCIe