Lecture B-1

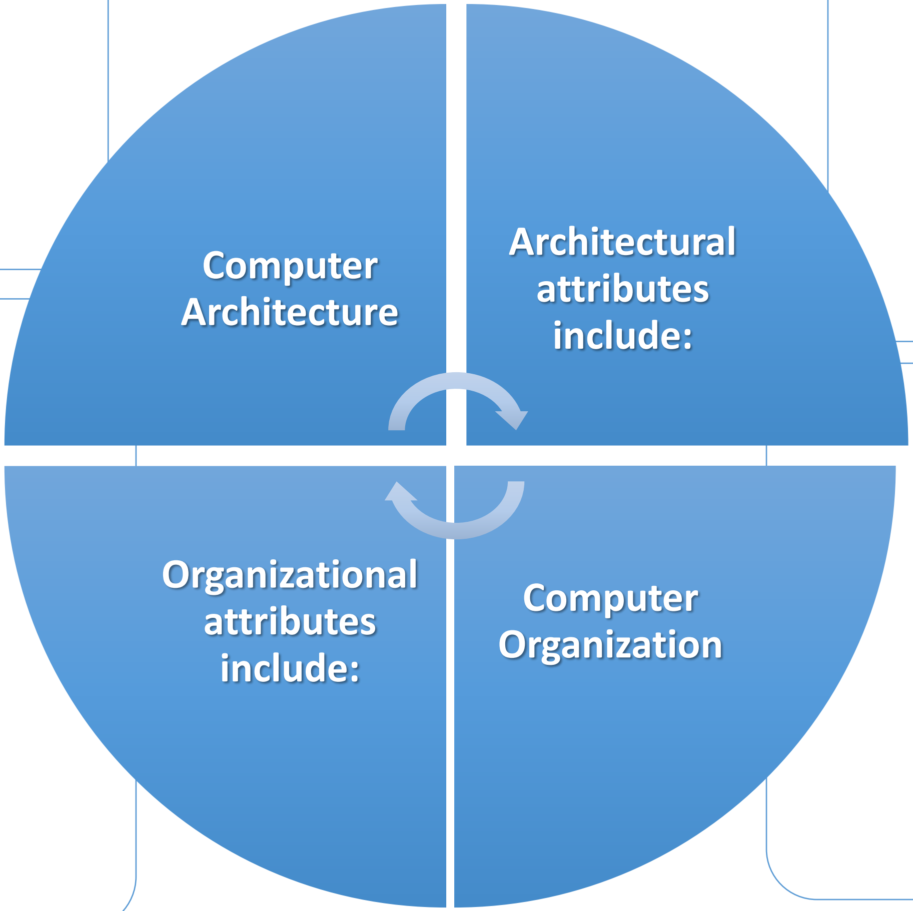
Basic Concepts and System Buses

Computer Architecture

Computer Organization

- Attributes of a system visible to the programmer
- Have a direct impact on the logical execution of a program

- Instruction set, number of bits used to represent various data types, I/O mechanisms, techniques for addressing memory



Computer Architecture
Architectural attributes include:
Organizational attributes include:
Computer Organization

- Hardware details transparent to the programmer, control signals, interfaces between the computer and peripherals, memory technology used

- The operational units and their interconnections that realize the architectural specifications

Structure and Function

- Hierarchical system
  - Set of interrelated subsystems
- Hierarchical nature of complex systems is essential to both their design and their description
- Designer need only deal with a particular level of the system at a time
  - Concerned with structure and function at each level

- Structure
  - The way in which components relate to each other
- Function
  - The operation of individual components as part of the structure

Function

- There are four basic functions that a computer can perform:
  - Data processing
    - Data may take a wide variety of forms and the range of processing requirements is broad
  - Data storage
    - Short-term
    - Long-term
  - Data movement
    - Input-output (I/O) - when data are received from or delivered to a device (peripheral) that is directly connected to the computer
    - Data communications – when data are moved over longer distances, to or from a remote device
  - Control
    - A control unit manages the computer’s resources and orchestrates the performance of its functional parts in response to instructions

Structure

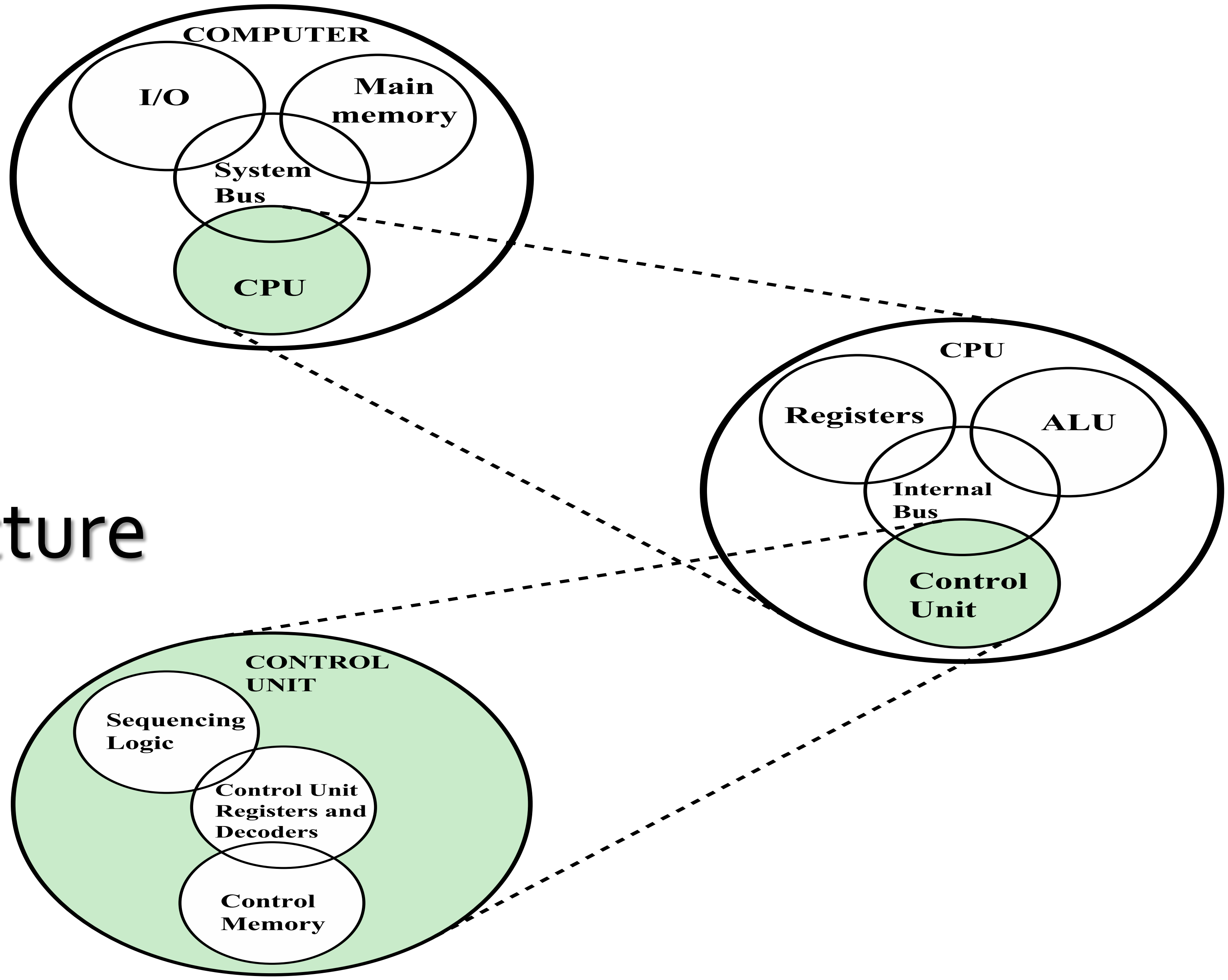


Figure 1.1 A Top-Down View of a Computer

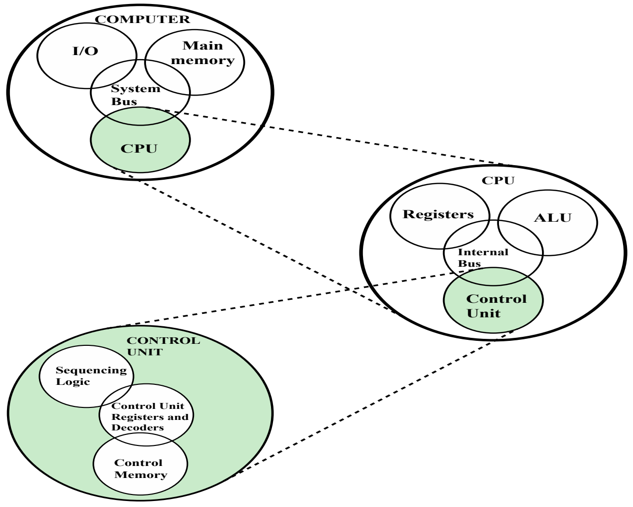
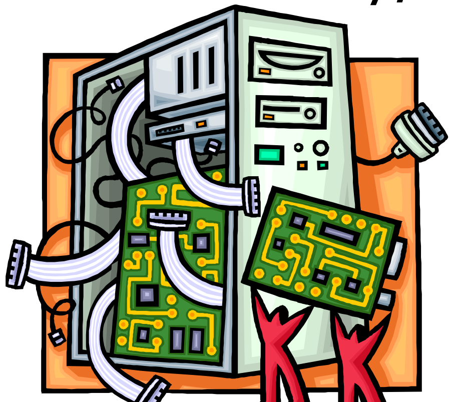


Figure 1.1 A Top-Down View of a Computer

There are four main structural components of the computer

- CPU – controls the operation of the computer and performs its data processing functions
- Main Memory – stores data
- I/O – moves data between the computer and its external environment
- System Interconnection or system bus – some mechanism that provides for communication among CPU, main memory, and I/O



CPU

Major structural components:

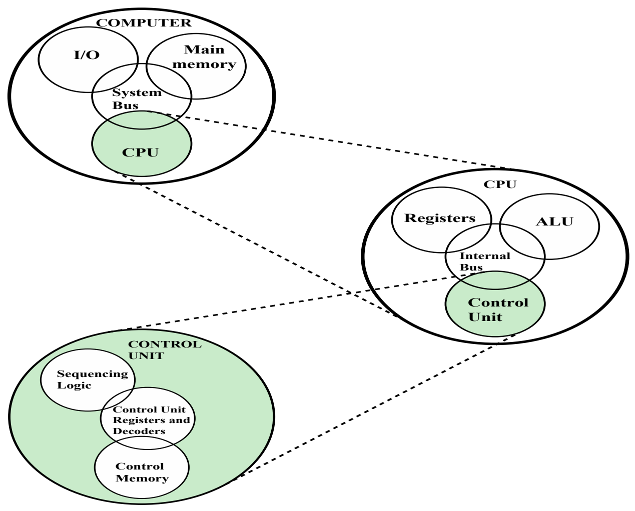
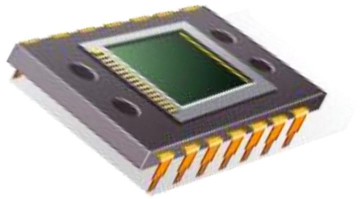
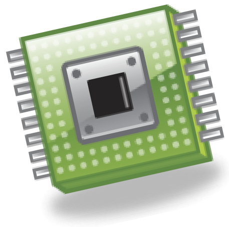


Figure 1.1 A Top-Down View of a Computer

- Control Unit
  - Controls the operation of the CPU and hence the computer
- Arithmetic and Logic Unit (ALU)
  - Performs the computer's data processing function
- Registers
  - Provide storage internal to the CPU
- CPU Interconnection
  - Some mechanism that provides for communication among the control unit, ALU, and registers



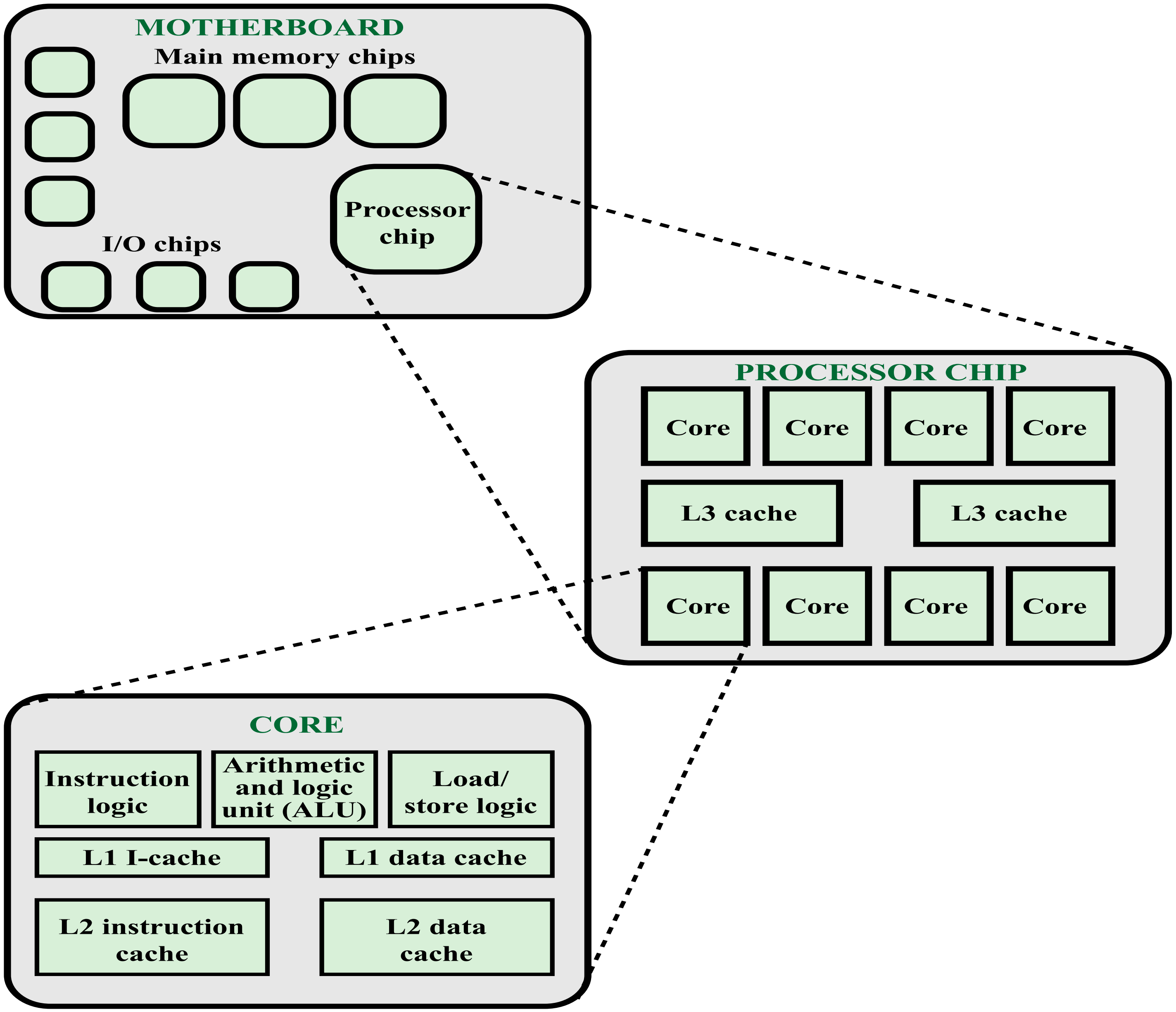


Multicore Computer Structure

- Central processing unit (CPU)
  - Portion of the computer that fetches and executes instructions
  - Consists of an ALU, a control unit, and registers
  - Referred to as a processor in a system with a single processing unit
- Core
  - An individual processing unit on a processor chip
  - May be equivalent in functionality to a CPU on a single-CPU system
  - Specialized processing units are also referred to as cores
- Processor
  - A physical piece of silicon containing one or more cores
  - Is the computer component that interprets and executes instructions
  - Referred to as a *multicore processor* if it contains multiple cores

Cache Memory

- Multiple layers of memory between the processor and main memory
- Is smaller and faster than main memory
- Used to speed up memory access by placing in the cache data from main memory that is likely to be used in the near future
- A greater performance improvement may be obtained by using multiple levels of cache, with level 1 (L1) closest to the core and additional levels (L2, L3, etc.) progressively farther from the core.


MOTHERBOARD
Main memory chips
I/O chips
Processor chip
PROCESSOR CHIP
Core
Core
Core
Core
L3 cache
L3 cache
Core
Core
Core
Core
CORE
Instruction logic
Arithmetic and logic unit (ALU)
Load/ store logic
L1 I-cache
L1 data cache
L2 instruction cache
L2 data cache

Multicore computer

Processor architecture

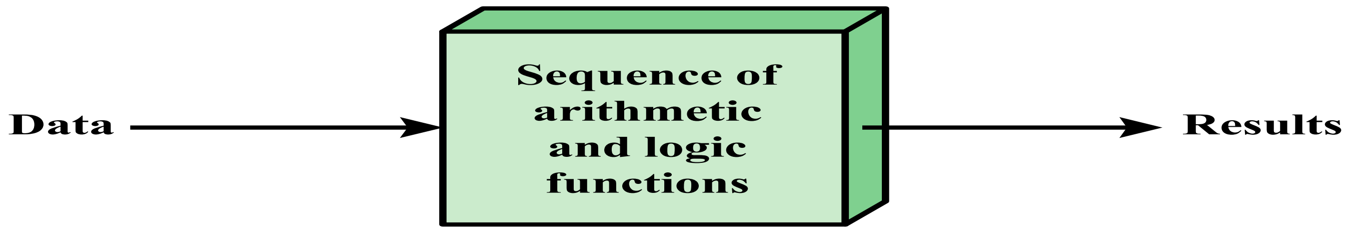
- The Figure in slide 11 shows a processor chip that contains eight cores and an L3 cache.
- Zooming in on the structure of a single core, which occupies a portion of the processor chip. In general terms, the functional elements of a core are:
  - Instruction logic: This includes the tasks involved in fetching instructions and decoding each instruction to determine the instruction operation and the memory locations of any operands.
  - Arithmetic and logic unit (ALU): Performs the operation specified by an instruction.
  - Load/store logic: Manages the transfer of data to and from main memory via cache.

Computer Components

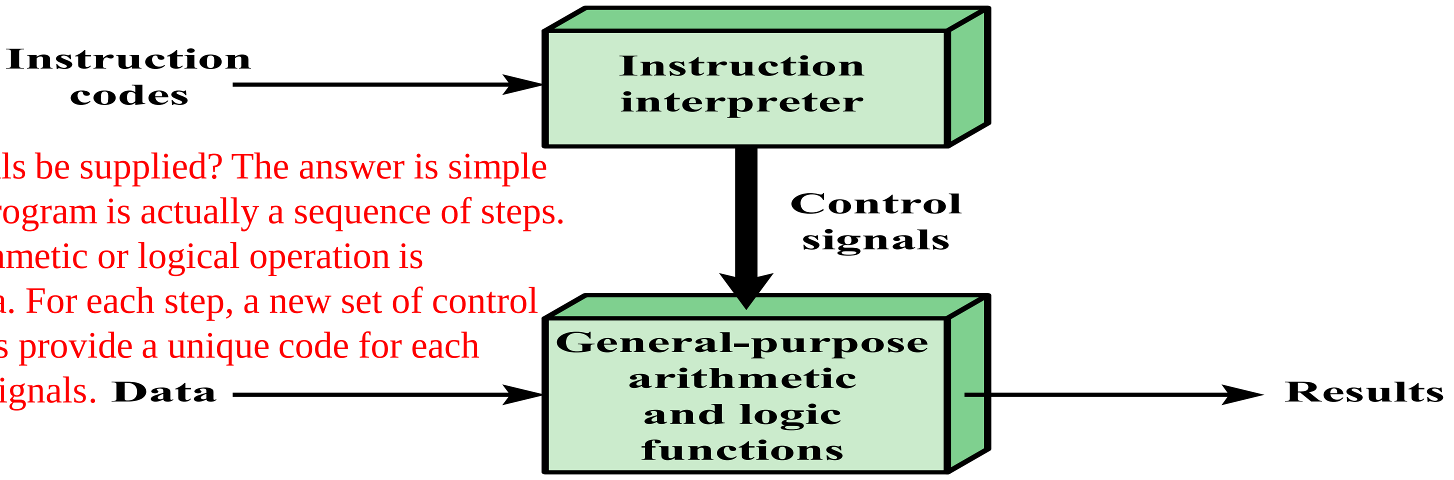
- Contemporary computer designs are based on concepts developed by ***John von Neumann*** at the Institute for Advanced Studies, Princeton
- Referred to as the *von Neumann architecture* and is based on three key concepts:
  - **Data and instructions are stored in a single read-write memory**
  - **The 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

Hardware and Software Approaches



(a) Programming in hardware



How shall control signals be supplied? The answer is simple but subtle. The entire program is actually a sequence of steps. At each step, some arithmetic or logical operation is performed on some data. For each step, a new set of control signals is needed. Let us provide a unique code for each possible set of control signals.

(b) Programming in software

With general-purpose hardware, the system accepts data and control signals and produces results

Figure 3.1 Hardware and Software Approaches

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

Software

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

I/O Components

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

MEMORY

I/O address register (I/OAR)

- Specifies a particular I/O device

I/O buffer register (I/OBR)

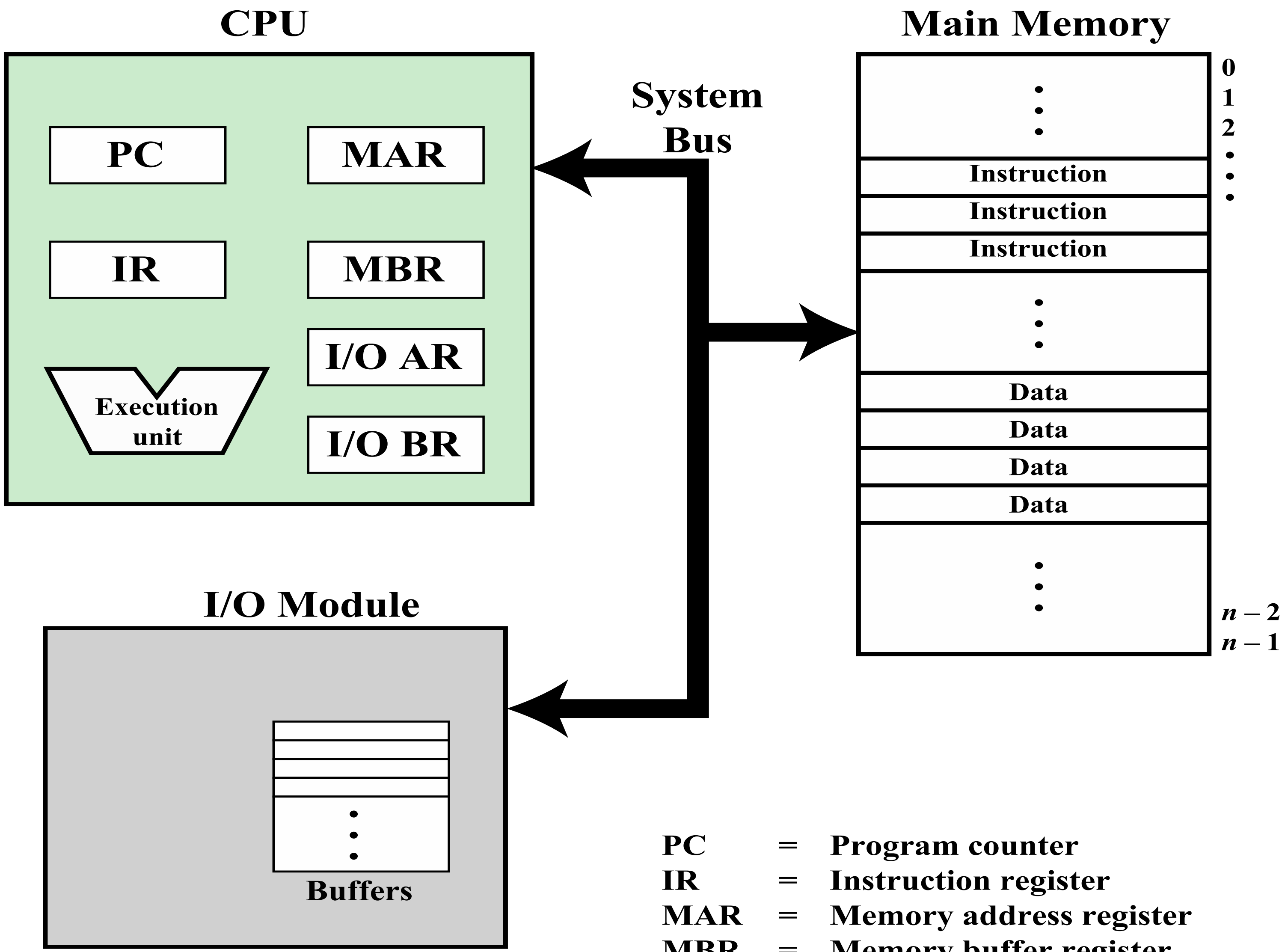
- Used for the exchange of data between an I/O module and the CPU

I/O

Memory & I/O

- The CPU exchanges data with memory. For this purpose, 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 register (MBR),** which contains the data to be written into memory or receives the data read from memory.
  - an **I/O address register (I/OAR)** specifies a particular I/O device.
  - an **I/O buffer (I/OBR) register** is used for the exchange of data between an I/O module and the CPU.

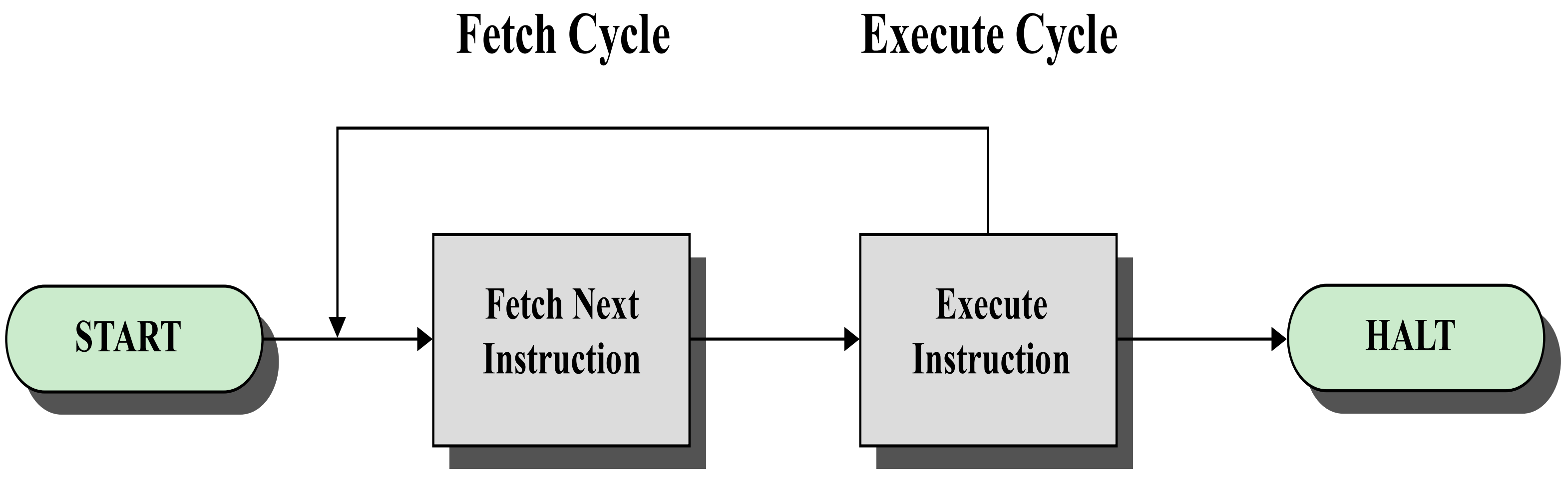
- A memory module consists of a set of locations, defined by sequentially numbered addresses. Each location contains a binary number that can be interpreted as either an instruction or data.
- An I/O module transfers data from external devices to CPU and memory, and vice versa. It contains internal buffers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



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

Program execution

- The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory.
- The processor does the actual work by executing instructions specified in the program.
- Instruction processing consists of two steps:
    - The processor reads (fetches) instructions from memory one at a time and executes 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.
- The two steps are referred to as the fetch cycle and the execute cycle.
- Program execution halts only if the machine is turned off, some sort of unrecoverable error occurs, or a program instruction that halts the computer is encountered.


Fetch Cycle
Execute Cycle
START
Fetch Next Instruction
Execute Instruction
HALT

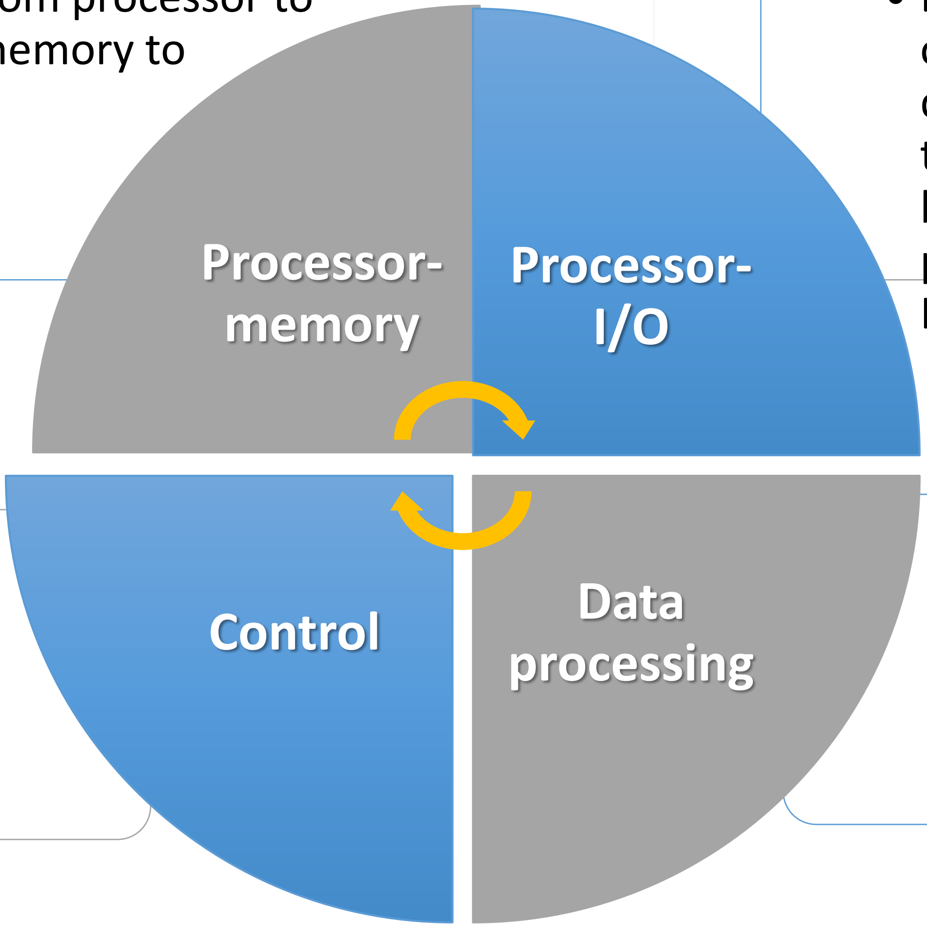
Fetch Cycle

- 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

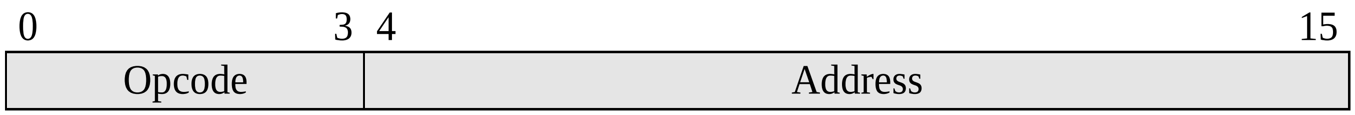


- An instruction may specify that the sequence of execution be altered

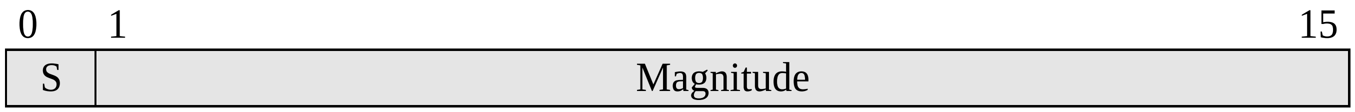
- The processor may perform some arithmetic or logic operation on data

Program execution example (1)

- The processor contains a single data register, called an accumulator (AC).
- Both instructions and data are 16 bits long. Thus, it is convenient to organize memory using 16-bit words.
- The instruction format provides 4 bits for the opcode, so that there can be as many as $2^4 = 16$ different opcodes, and up to $2^{12} = 4096$ (4K) words of memory can be directly addressed.
- The following slides illustrate a partial program execution, showing the relevant portions of memory and processor registers.
  - The program fragment shown adds the contents of the memory word at address 940 to the contents of the memory word at address 941 and stores the result in the latter location.



(a) Instruction format



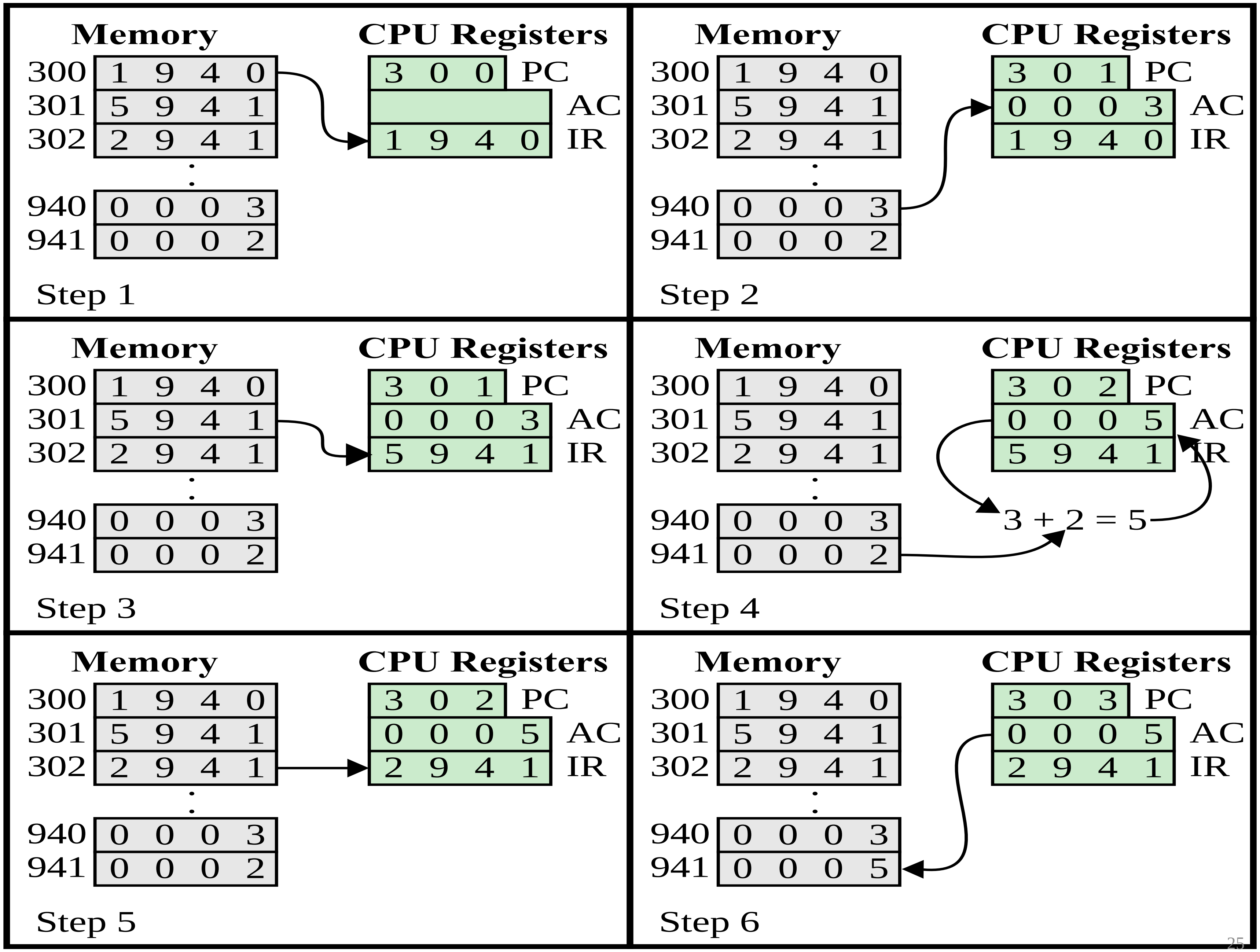
(b) Integer format

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


Memory
CPU Registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 0 PC
AC
1 9 4 0 IR
Step 1
Memory
CPU Registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 1 PC
0 0 0 3 AC
1 9 4 0 IR
Step 2
Memory
CPU Registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 1 PC
0 0 0 3 AC
5 9 4 1 IR
Step 3
Memory
CPU Registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 2 PC
0 0 0 5 AC
5 9 4 1 IR
3 + 2 = 5
Step 4
Memory
CPU Registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 2 PC
0 0 0 5 AC
2 9 4 1 IR
Step 5
Memory
CPU Registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 5
3 0 3 PC
0 0 0 5 AC
2 9 4 1 IR
Step 6

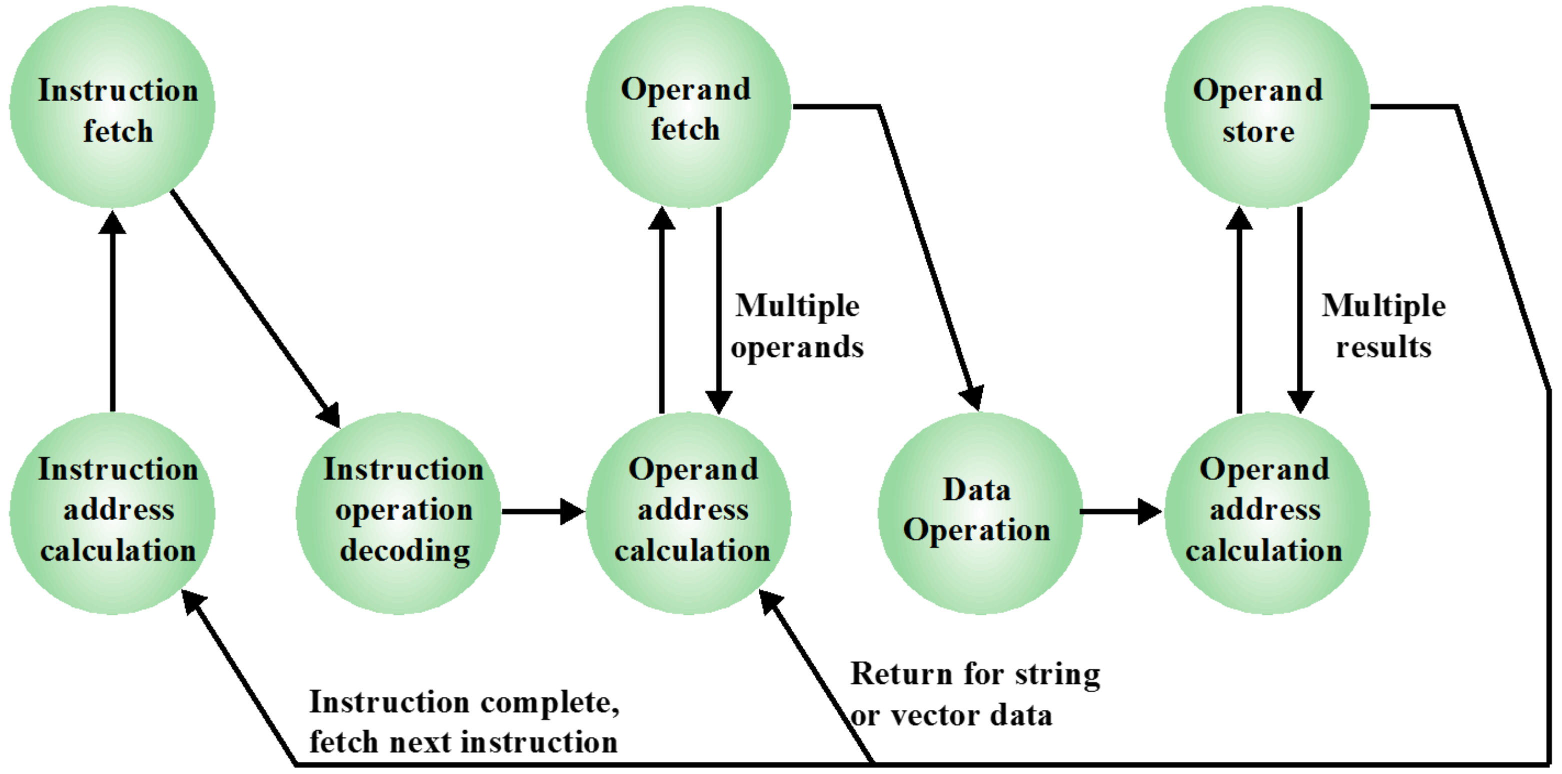
Program execution example (2)

- Three instructions, which can be described as three fetch and three execute cycles, are required:
  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.

Instruction cycle

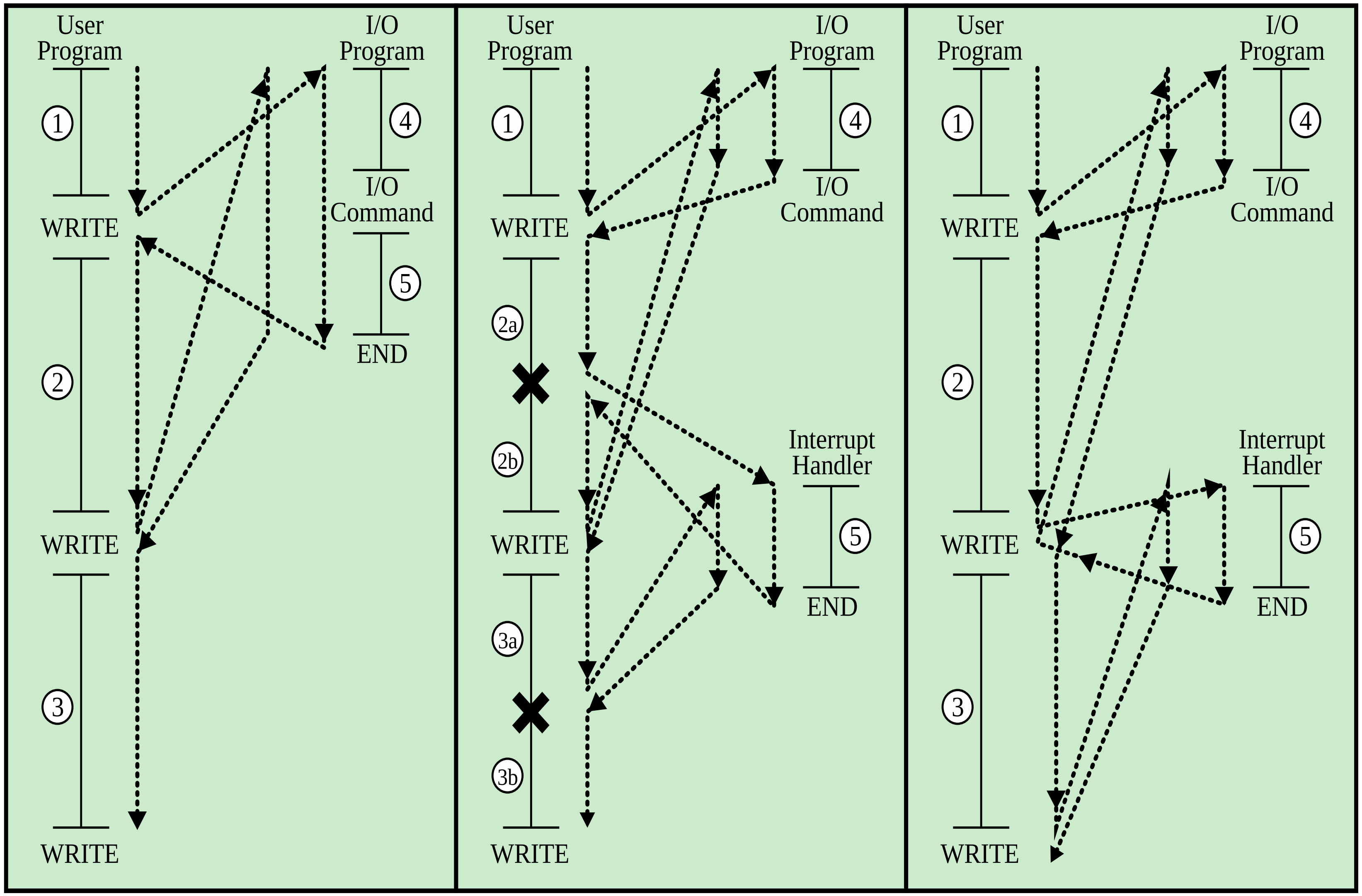
- The execution cycle for a particular instruction may involve more than one reference to memory. Also, instead of memory references, an instruction may specify an I/O operation.
- For any given instruction cycle, some states may be null and others may be visited more than once. The states can be described as follows:
  - **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. For example, if each instruction is 16 bits long and memory is organized into 16-bit words, then add 1 to the previous address. If, instead, memory is organized as individually addressable 8-bit bytes, then add 2 to the previous address.
  - **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.

Instruction cycle state diagram


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

Interrupts (1)

- Interrupts are provided primarily as a way to improve processing efficiency. For example, most external devices are much slower than the processor.
- Suppose that the processor is transferring data to a printer. After each write operation, the processor must pause and remain idle until the printer catches up.
- The length of this pause may be on the order of many hundreds or even thousands of instruction cycles that do not involve memory. This is a very wasteful use of the processor.
- The Figure (a) in the next slide illustrates this state of affairs. The user program performs a series of WRITE calls interleaved with processing.
  - Code segments 1, 2, and 3 refer to sequences of instructions that do not involve I/O.
  - The WRITE calls are to an I/O program that is a system utility and that will perform the actual I/O operation.



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

✖ = interrupt occurs during course of execution of user program

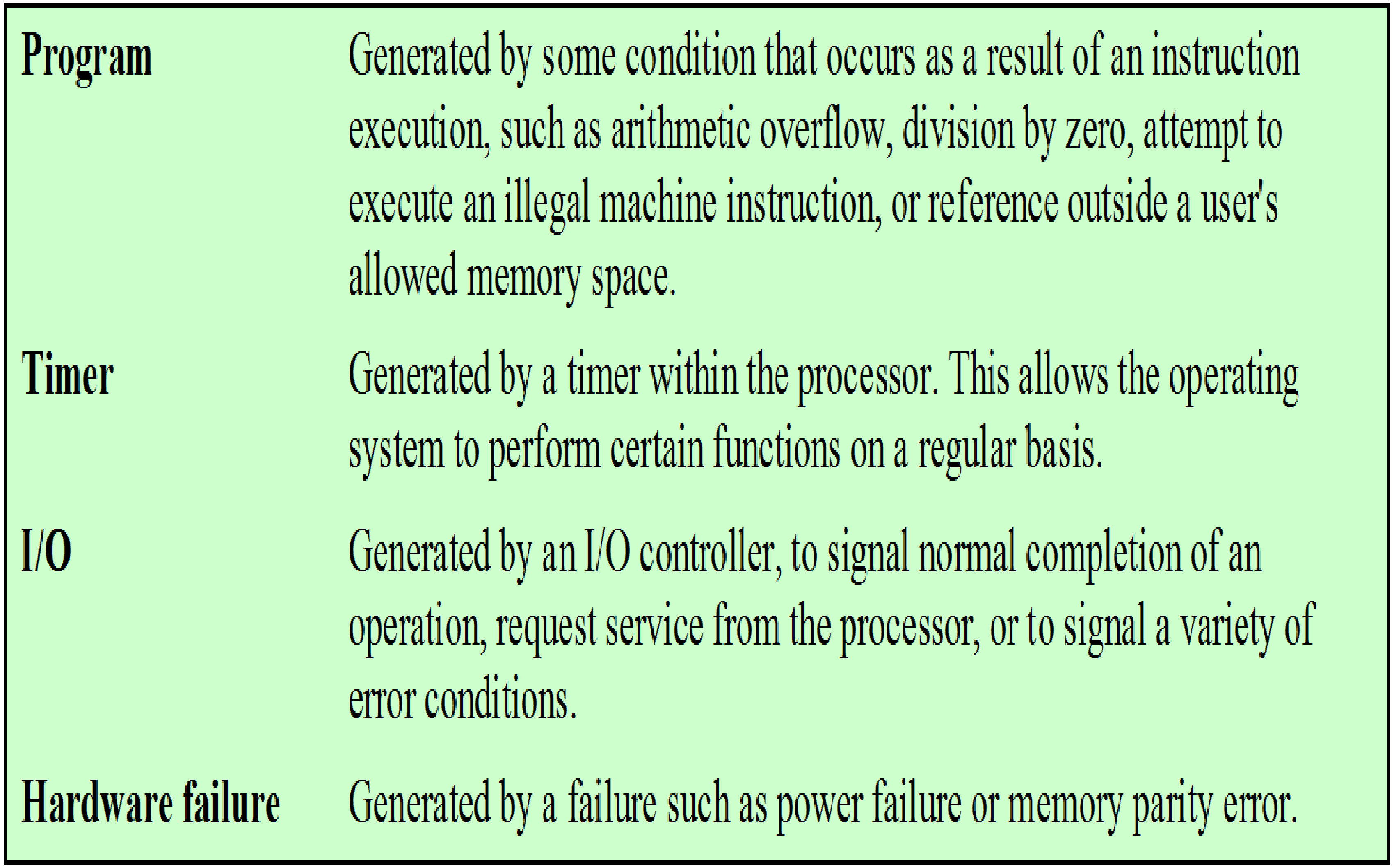
Interrupts (2)

- The I/O program consists of three sections:
  - A sequence of instructions, labeled 4 in the Figure, to prepare for the actual I/O operation. This may include copying the data to be output into a special buffer and preparing the parameters for a device command.
  - The actual I/O command. Without the use of interrupts, once this command is issued, the program must wait for the I/O device to perform the requested function (or periodically poll the device). The program might wait by simply repeatedly performing a test operation to determine if the I/O operation is done.
  - A sequence of instructions, labeled 5 in the Figure, to complete the operation. This may include setting a flag indicating the success or failure of the operation.
- Because the I/O operation may take a relatively long time to complete, the I/O program is hung up waiting for the operation to complete; hence, the user program is stopped at the point of the WRITE call for some considerable period of time.
- With interrupts, the processor can be engaged in executing other instructions while an I/O operation is in progress.

Interrupts (3)

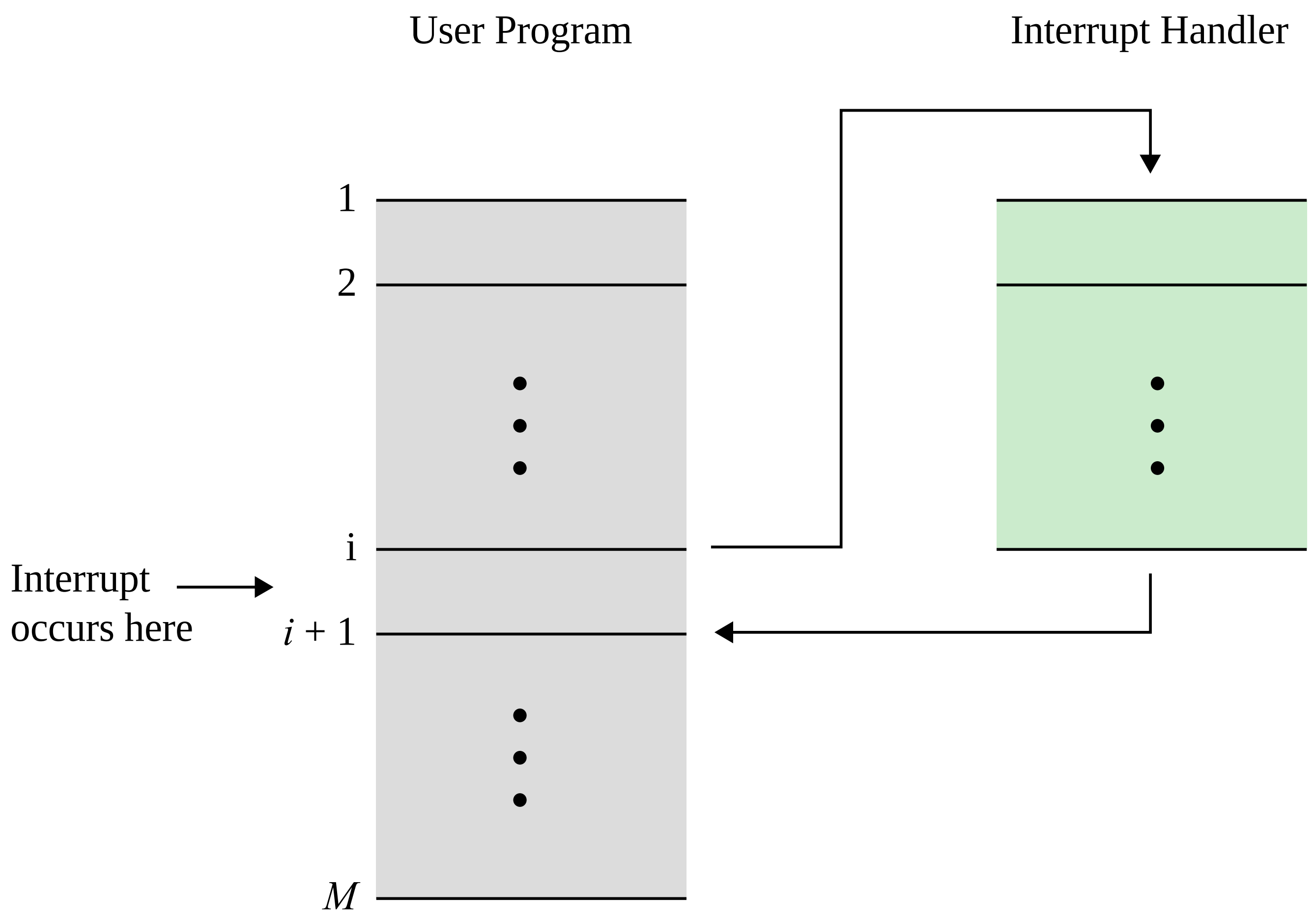
- Consider the flow of control in (b):
  - The user program reaches a point at which it makes a system call in the form of a WRITE call. The I/O program that is invoked in this case consists only of the preparation code and the actual I/O command.
  - After these few instructions have been executed, control returns to the user program. Meanwhile, the external device is busy accepting data from computer memory and printing it. This I/O operation is conducted concurrently with the execution of instructions in the user program.
  - When the external device becomes ready to be serviced—that is, when it is ready to accept more data from the processor—the I/O module for that external device sends an interrupt request signal to the processor.
  - The processor responds by suspending operation of the current program, branching off to a program to service that particular I/O device, known as an interrupt handler, and resuming the original execution after the device is serviced. The points at which such interrupts occur are indicated by an asterisk in (b).

Classes of interrupts

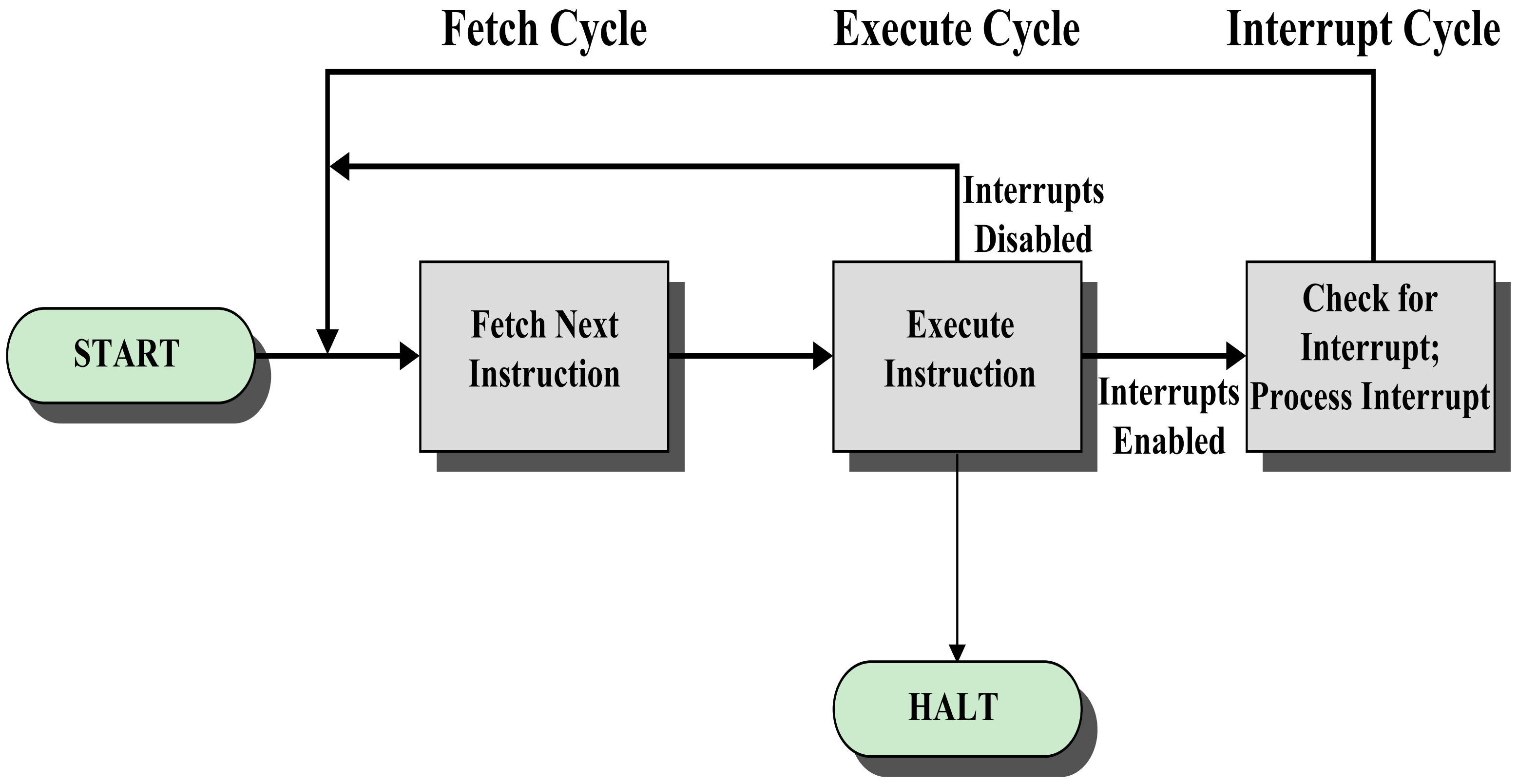


Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the 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.

Transfer of control via interrupts



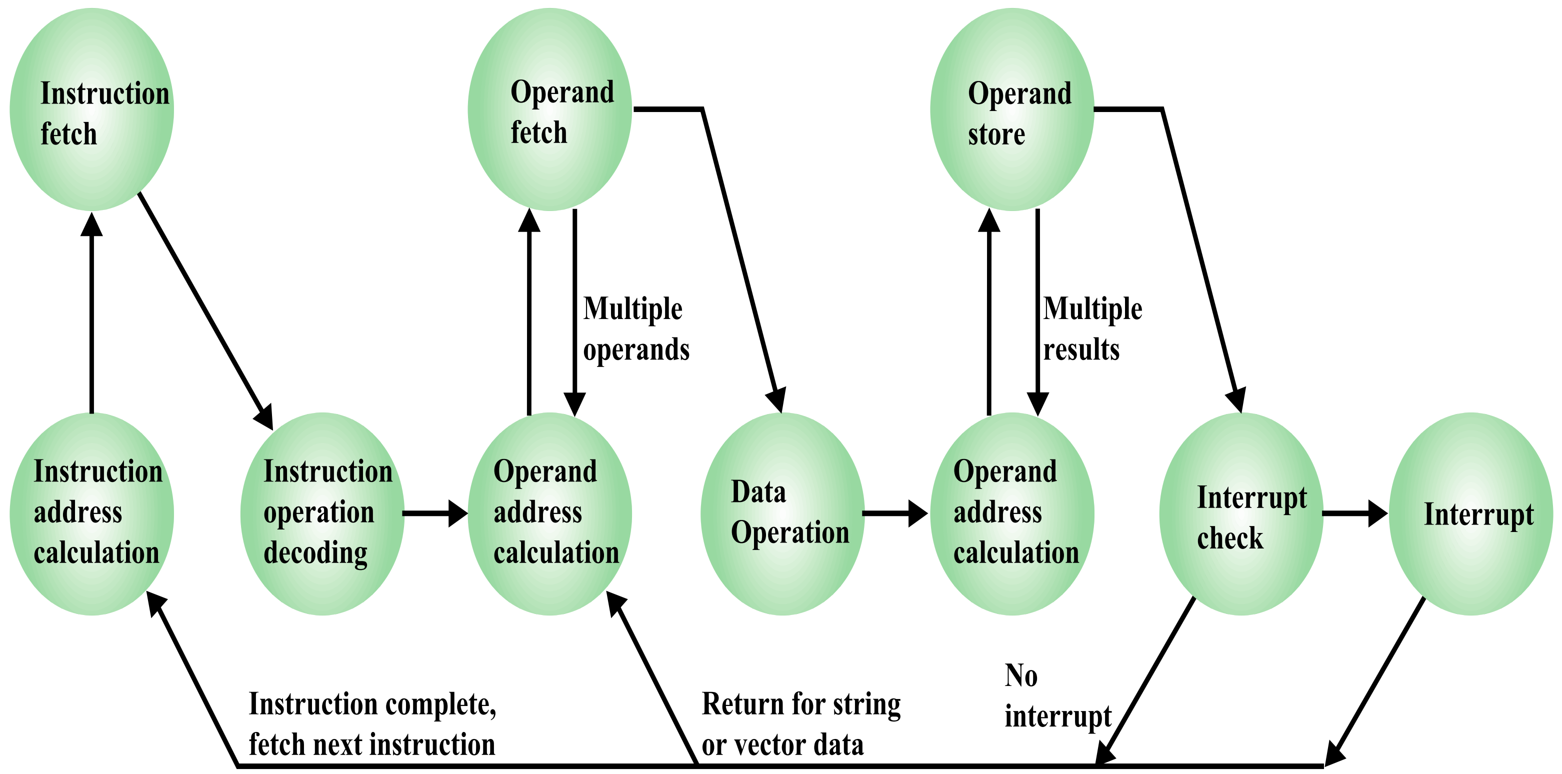
Instruction cycle with interrupts (1)



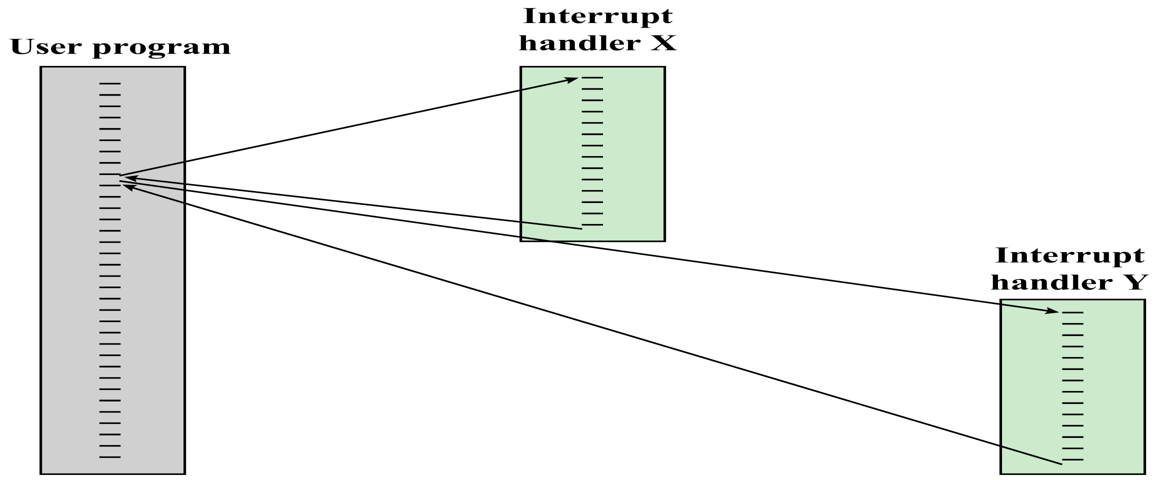
Instruction cycle with interrupts (2)

- 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.

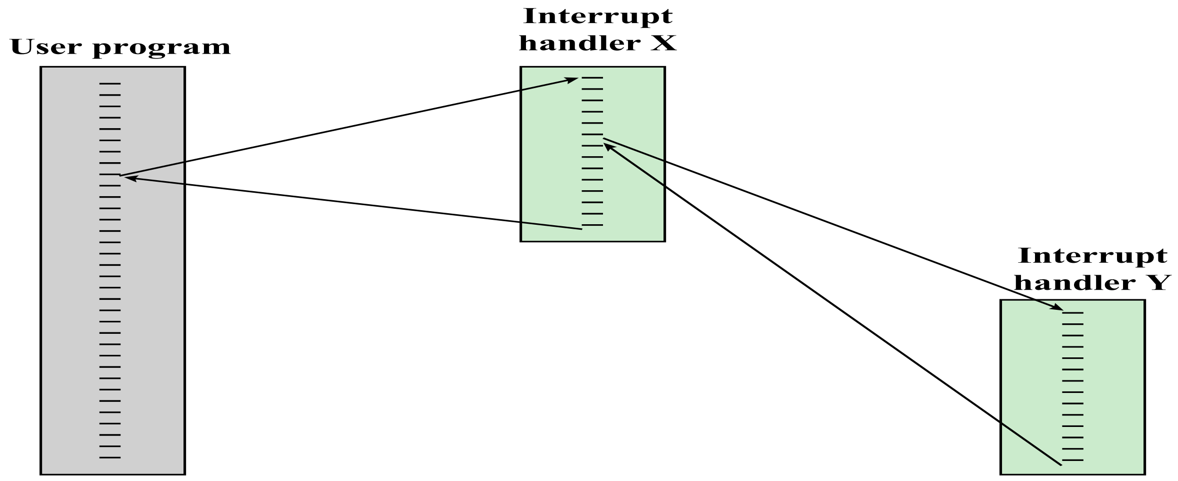
Instruction cycle state diagram with interrupts


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

Transfer of control for multiple interrupts

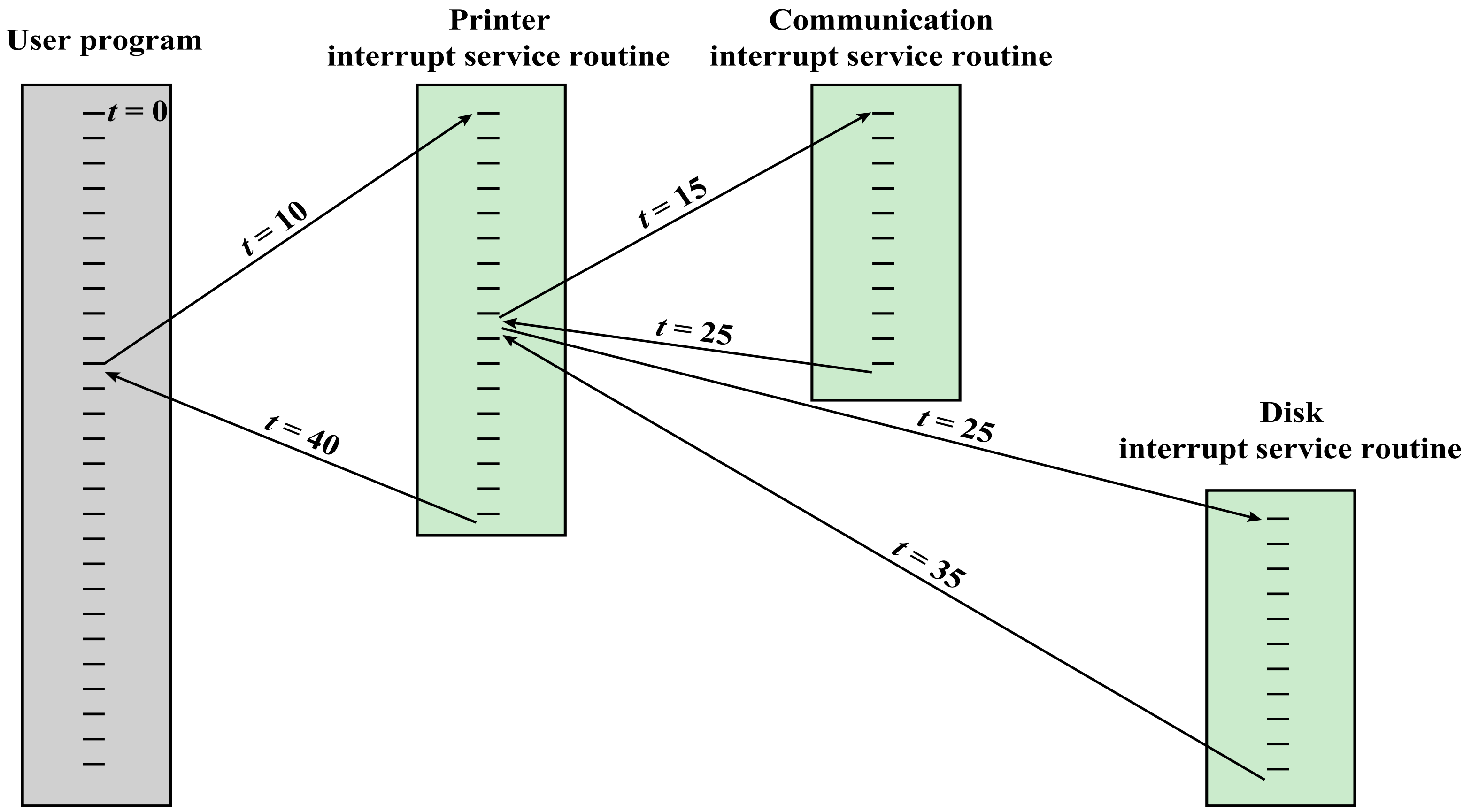


(a) Sequential interrupt processing



(b) Nested interrupt processing

Time sequence of 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

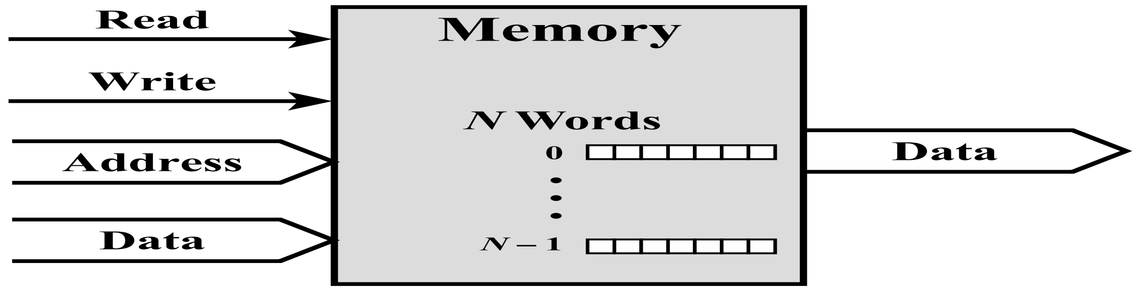
I/O Function

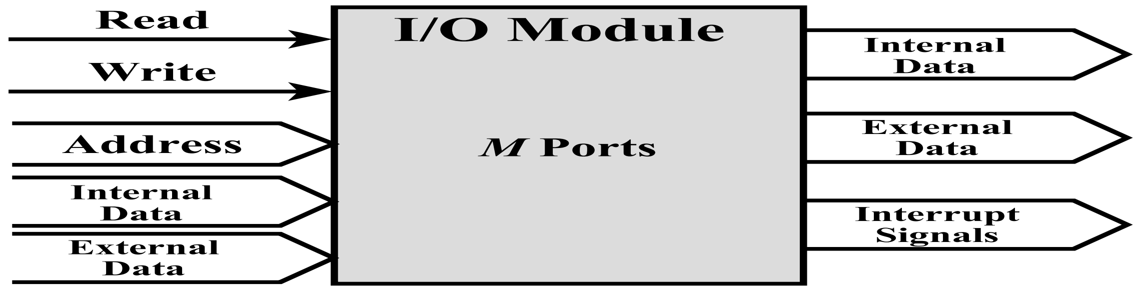
- I/O module can exchange data directly with the processor
- Processor can read data from or write data to an I/O module
  - Processor identifies a specific device that is controlled by a particular I/O module
  - I/O instructions rather than memory referencing instructions
- In some cases it is desirable 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)**

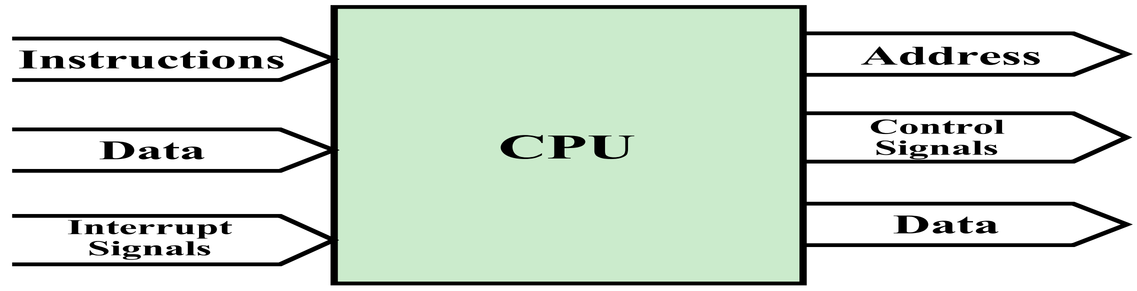
Computer modules (1)

- A computer consists of a set of components or modules of three basic types (processor, memory, I/O) that communicate with each other. In effect, a computer is a network of basic modules. Thus, there must be paths for connecting the modules.
- The collection of paths connecting the various modules is called the interconnection structure. The design of this structure will depend on the exchanges that must be made among modules.
- The 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). A word of data can be read from or written into the memory. 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**: From an internal (to the computer system) point of view, 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.

Computer modules (2)







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

I n t e r c o n n e c t i o n B u s

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

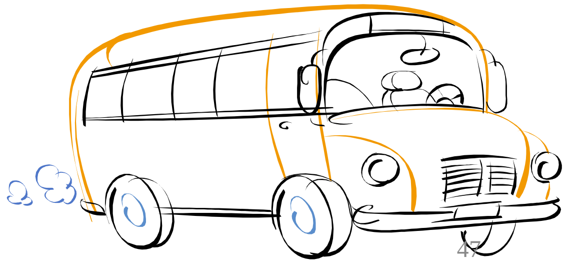
System bus

- 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



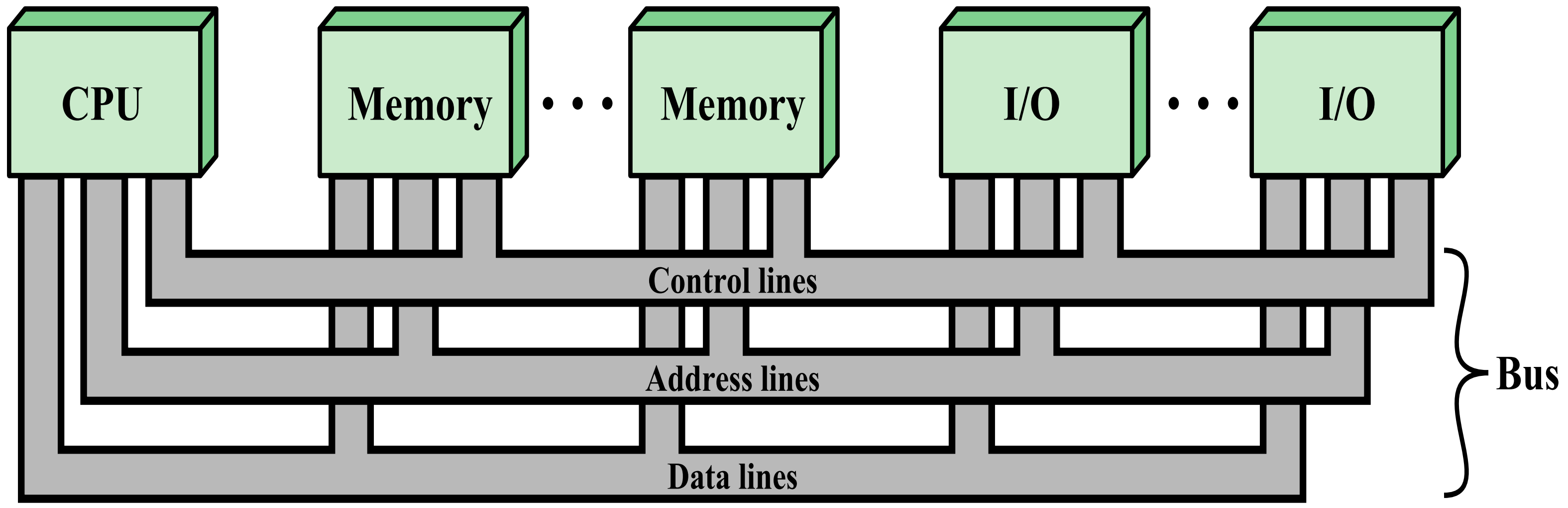
Address Bus

- **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

Bus interconnection scheme



Bus interconnection scheme

- The operation of the bus is as follows. If one module wishes to send data to another, it must do two things:
  1. obtain the use of the bus
  2. transfer data via the bus
- If one module wishes to request data from another module, it must:
  1. obtain the use of the bus
  2. transfer a request to the other module over the appropriate control and address lines
- It must then wait for that second module to send the data

Examples of processor architectures

- Two processor families are the Intel x86 and the ARM architectures
- Current x86 offerings represent the results of decades of design effort on complex instruction set computers (CISCs)
- An alternative approach to processor design is the reduced instruction set computer (RISC)
- ARM architecture is used in a wide variety of embedded systems and is one of the most powerful and best-designed RISC-based systems on the market

Highlights of the evolution of the Intel processors (1)

8080

- World's first general-purpose microprocessor
- 8-bit machine, 8-bit data path to memory
- Was used in the first personal computer (Altair)

8086

- A more powerful 16-bit machine
- Has an instruction cache, or queue, that prefetches a few instructions before they are executed
- The first appearance of the x86 architecture
- The 8088 was a variant of this processor and used in IBM's first personal computer (securing the success of Intel

80286

- Extension of the 8086 enabling addressing a 16-MB memory instead of just 1MB

80386

- Intel's first 32-bit machine
- First Intel processor to support multitasking

80486

- Introduced the use of much more sophisticated and powerful cache technology and sophisticated instruction pipelining
- Also offered a built-in math coprocessor

Highlights of the evolution of the Intel processors (2)

Pentium

- Intel introduced the use of superscalar techniques, which allow multiple instructions to execute in parallel

Pentium Pro

- Continued the move into superscalar organization with aggressive use of register renaming, branch prediction, data flow analysis, and speculative execution

Pentium II

- Incorporated Intel MMX technology, which is designed specifically to process video, audio, and graphics data efficiently

Pentium III

- Incorporated additional floating-point instructions
- Streaming SIMD Extensions (SSE)

Pentium 4

- Includes additional floating-point and other enhancements for multimedia

Core

- First Intel x86 micro-core

Core 2

- Extends the Core architecture to 64 bits
- Core 2 Quad provides four cores on a single chip
- More recent Core offerings have up to 10 cores per chip
- An important addition to the architecture was the Advanced Vector Extensions instruction set

ARM architecture

Refers to a processor architecture that has evolved ***from RISC design*** principles and is used in embedded systems

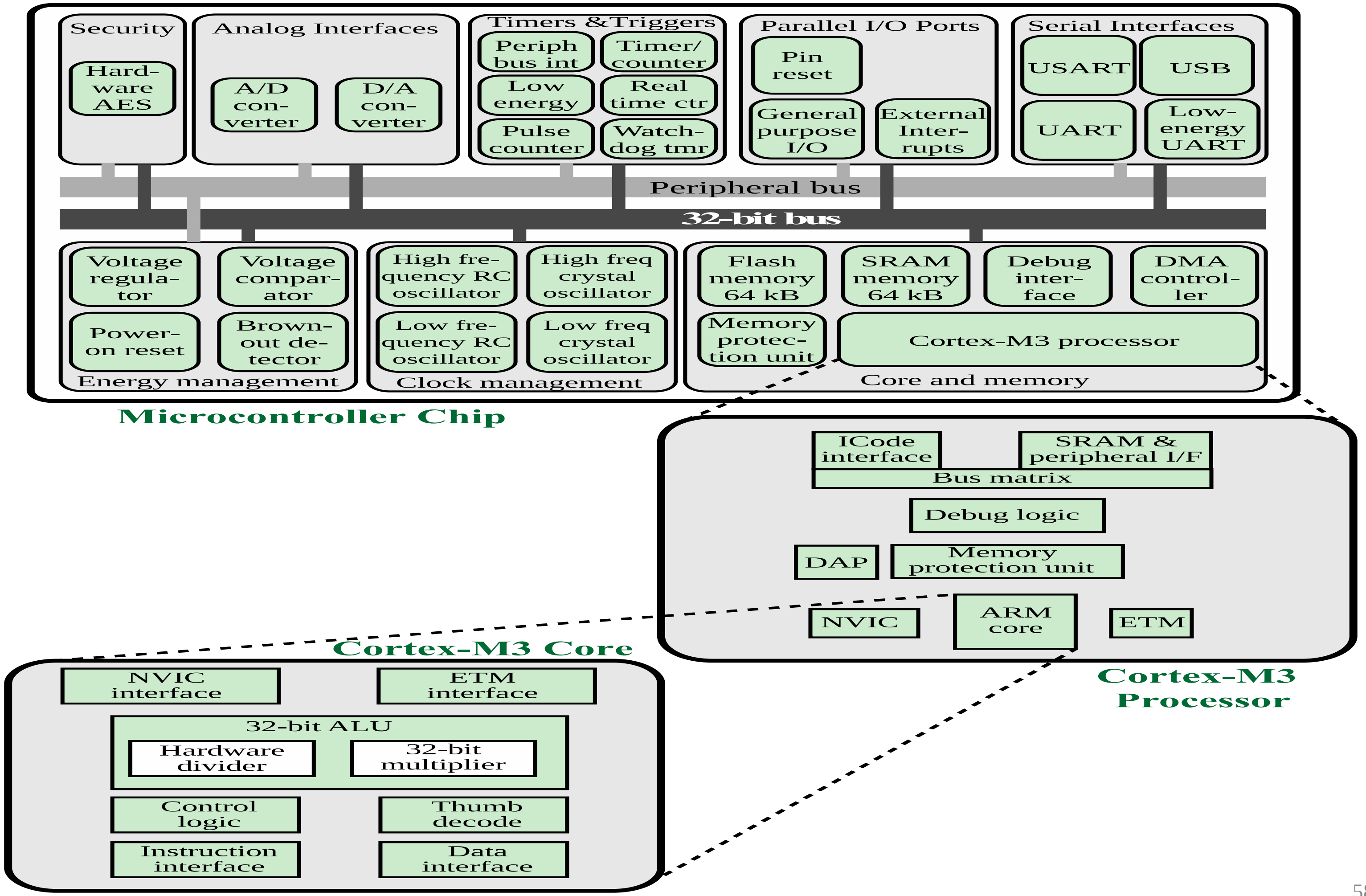
Family of RISC-based microprocessors and microcontrollers designed by ARM Holdings, Cambridge, England

Chips are high-speed processors that are known for their small die size and low power requirements

Probably the most widely used embedded processor architecture and indeed the most widely used processor architecture of any kind in the world

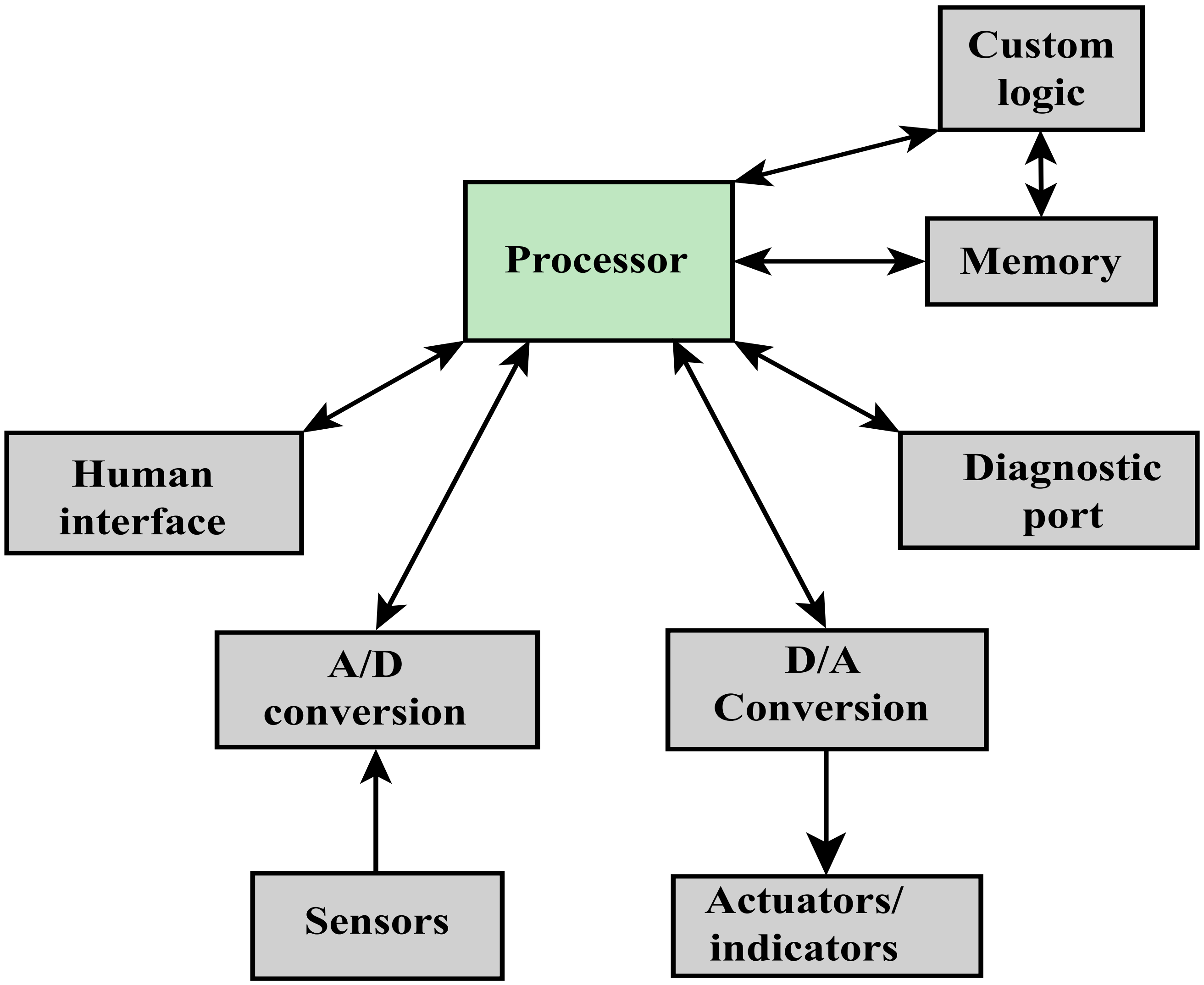
Acorn RISC Machine/Advanced RISC Machine

ARM processor based on Cortex-M3



Embedded Systems

- The use of electronics and software within a product
- Billions of computer systems are produced each year that are embedded within larger devices
- Today many devices that use electric power have an embedded computing system
- Often embedded systems are tightly coupled to their environment
  - This can give rise to real-time constraints imposed by the need to interact with the environment
    - Constraints such as required speeds of motion, required precision of measurement, and required time durations, dictate the timing of software operations
  - If multiple activities must be managed simultaneously this imposes more complex real-time constraints


Custom logic
Processor
Memory
Human interface
Diagnostic port
A/D conversion
D/A Conversion
Sensors
Actuators/ indicators

The Internet of Things (IoT) (1)

- **Term that refers to the expanding interconnection of smart devices, ranging from appliances to tiny sensors.**
- Is primarily driven by deeply embedded devices.
- Users interact with the device in a natural way, similar to their interactions with any other objects in the world.
- In this way, an embedded system has an interface that conforms to the expectations and needs of the users.
- Establishing a natural interface requires that the embedded system interface with the physical world directly through sensors, which read the state of the world, and actuators, which change the state of the world.
- It is marked by the use of billions of embedded devices

The Internet of Things (IoT) (2)

- IoT devices are implemented using both hardware and software components.
- Dedicated hardware components are used to implement the interface with the physical world, and to perform tasks which are more computationally complex.
- Microcontrollers are used to execute software that interprets inputs and controls the system.
- The functions of common hardware components are described and the interface between the software and hardware through the microcontroller is explained.
- IoT devices often use an operating system to support the interaction between the software and the microcontroller.

The Internet of Things (IoT) (3)

- An important aspect of the Internet of Things is that devices are networked in some way, and often connected to the Internet.
- Networking enables devices to communicate with other IoT devices and larger cloud-based servers.
- IoT devices can often be thought of as small parts of a much larger collective system which includes large servers based in the cloud.
- Eventually, most IoT devices are connected to the Internet, so understanding the protocols associated with the Internet is important to the design of IoT devices.
- The concept of a Mobile Ad Hoc Network, or MANET, which describes small, local networks of IoT devices.

The Internet of Things (IoT) (4)

- IoT is used in:
  - Environmental monitoring (air/water quality, atmospheric conditions, soil conditions)
  - Infrastructure management (monitoring structural conditions, bridges, railway tracks - facilitate repair, maintenance and safety)
  - Manufacturing (automate process controls, predictive maintenance, statistical evaluation)
  - Transportation (smart traffic control, logistic and fleet management)
  - Building and home automation (monitor and control electricity usage, temperature)