William Stallings Computer Organization and Architecture

Chapter 3
Top Level View of Computer Function and Interconnection

Program Concept

- Hardwired systems are inflexible
- General purpose hardware can do different tasks, given correct control signals
- Instead of re-wiring, supply a new set of control signals

What is a program?

- A sequence of steps
- For each step, an arithmetic or logical operation is done
- For each operation, a different set of control signals is needed

Function of Control Unit

- For each operation a unique code is provided
  - e.g. ADD, MOVE
- A hardware segment accepts the code and issues the control signals

- We have a computer!

Components

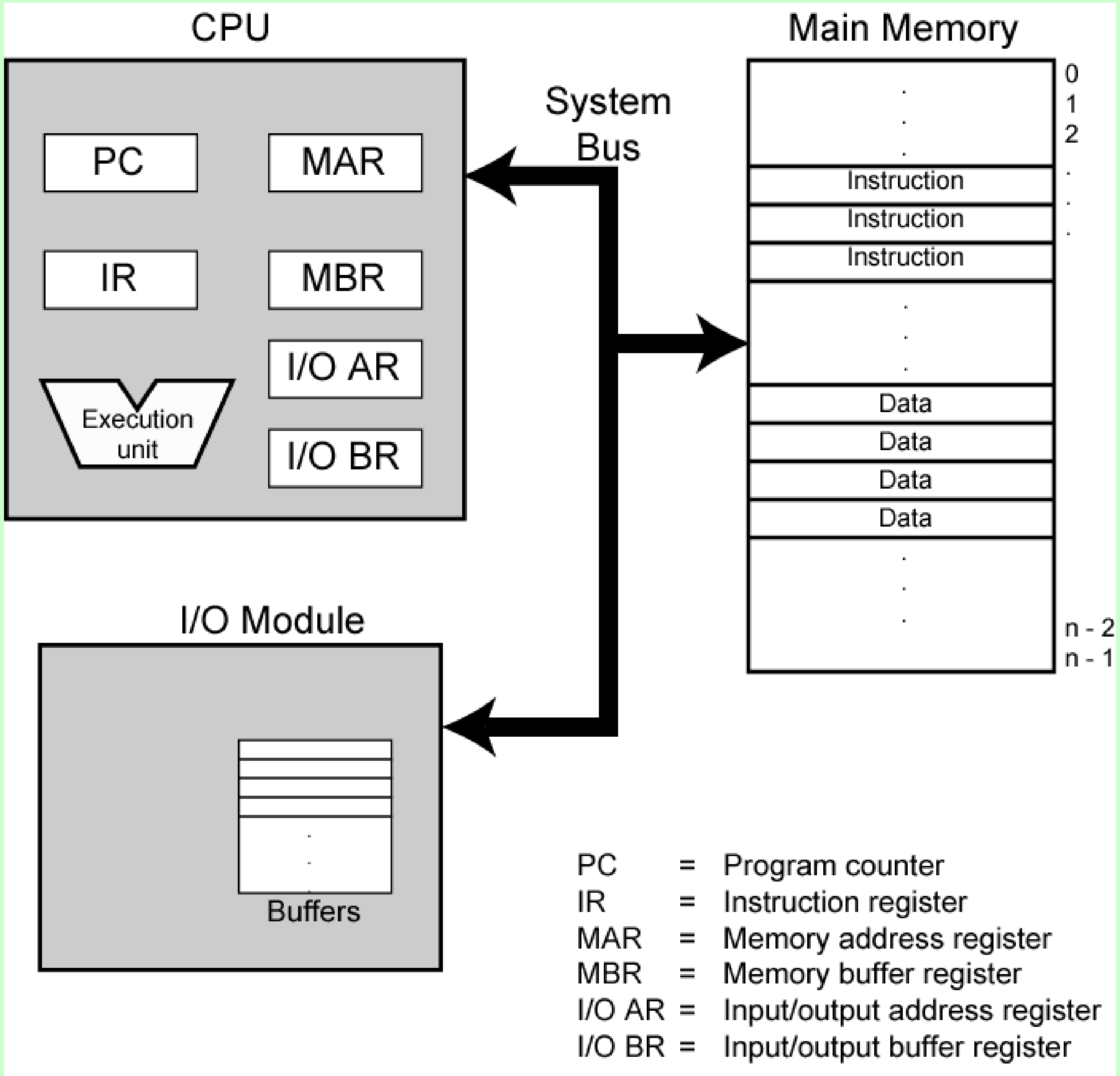
- The Control Unit and the Arithmetic and Logic Unit constitute the Central Processing Unit
- Data and instructions need to get into the system and results out
  - —Input/output
- Temporary storage of code and results is needed
  - —Main memory

Arsitektur Von Neumann

Virtually all contemporary computer designs are based on concepts developed by John von Neumann at the Institute for Advanced Studies, Princeton. Such a design is referred to as the *von Neumann architecture.*

1. Data and instructions are stored in a single read–write memory.
2. The contents of this memory are addressable by location, without regard to the type of data contained there.
3. Execution occurs in a sequential fashion (unless explicitly modified) from one instruction to the next.

Computer Components: Top Level View



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

Komponen (1)

- *Memory, or main memory* to distinguish it from external storage or peripheral devices. Von Neumann pointed out that the same memory could be used to store both instructions and data
- 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.

Komponen (2)

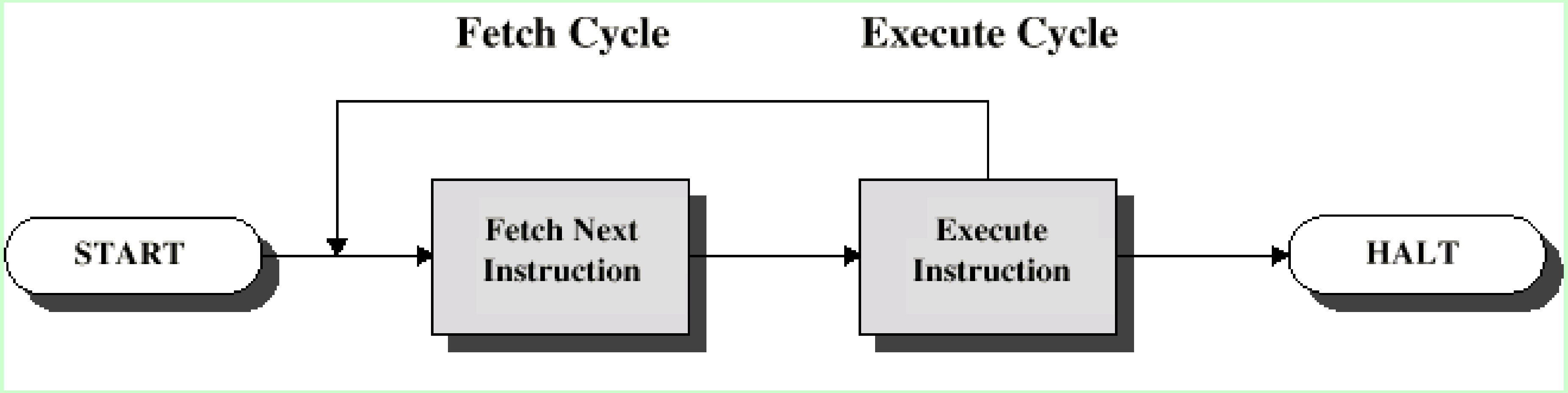
- Two internal (to the CPU) registers:
  - a memory address register (MAR), which specifies the address in memory for the next read or write, and a
  - memory buffer register (MBR), which contains the data to be written into memory or receives the data read from memory.

Serupa pula dengan IO

  - 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.

Instruction Cycle

- The processing required for a single instruction is called an instruction cycle
- The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory
- Two steps:
  - Fetch
  - Execute



Fetch Cycle

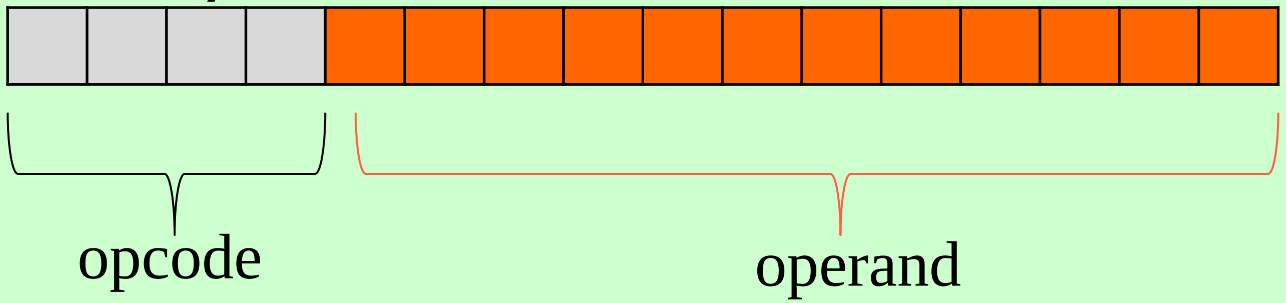
- Program Counter (PC) holds address of next instruction to fetch
- Processor fetches instruction from memory location pointed to by PC
- Increment PC
  - —Unless told otherwise
- Instruction loaded into Instruction Register (IR)
- Processor interprets instruction and performs required actions

Execute Cycle

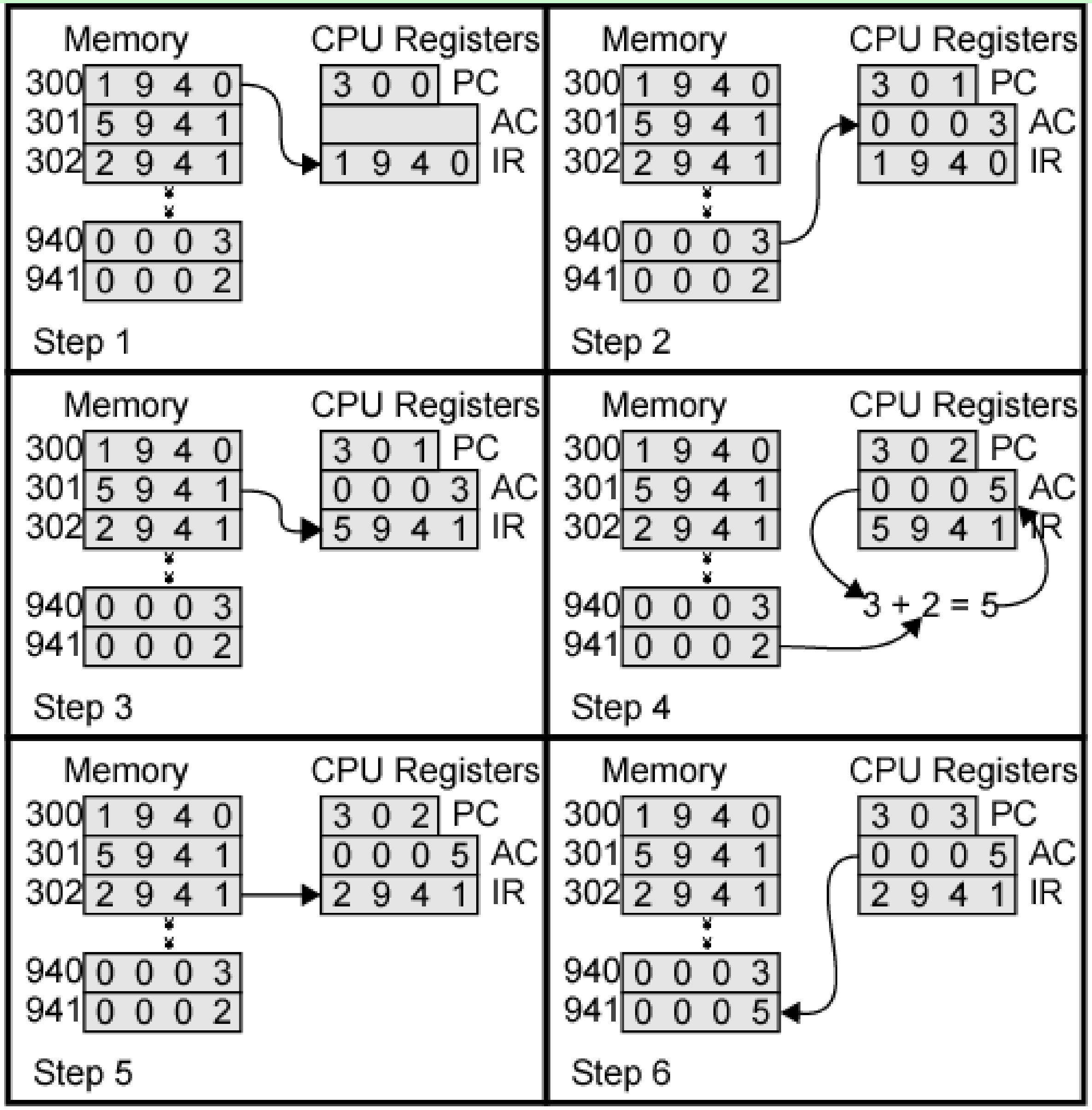
- Processor-memory
  - data transfer between CPU and main memory
- Processor I/O
  - Data transfer between CPU and I/O module
- Data processing
  - Some arithmetic or logical operation on data
- Control
  - Alteration of sequence of operations
  - e.g. jump. For example, the processor may fetch an instruction from location 149, which specifies that the next instruction be from location 182.
- Combination of above

Contoh

- 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.

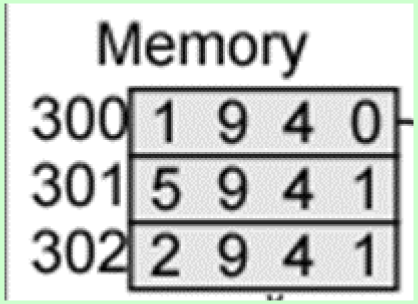


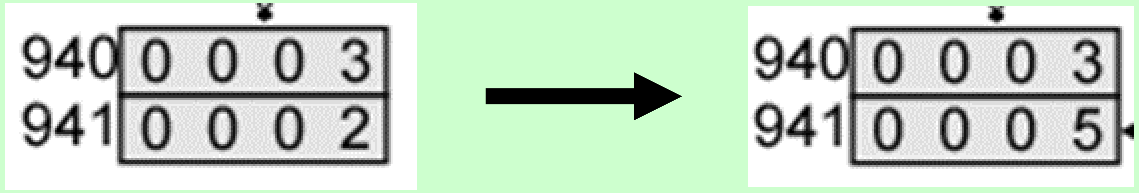
Example of Program Execution



Opcode, alamat dan hasil

- 0001 → 1 Load AC from memory
- 0101 → 5 Add to AC from memory
- 0010 → 2 Store AC to memory



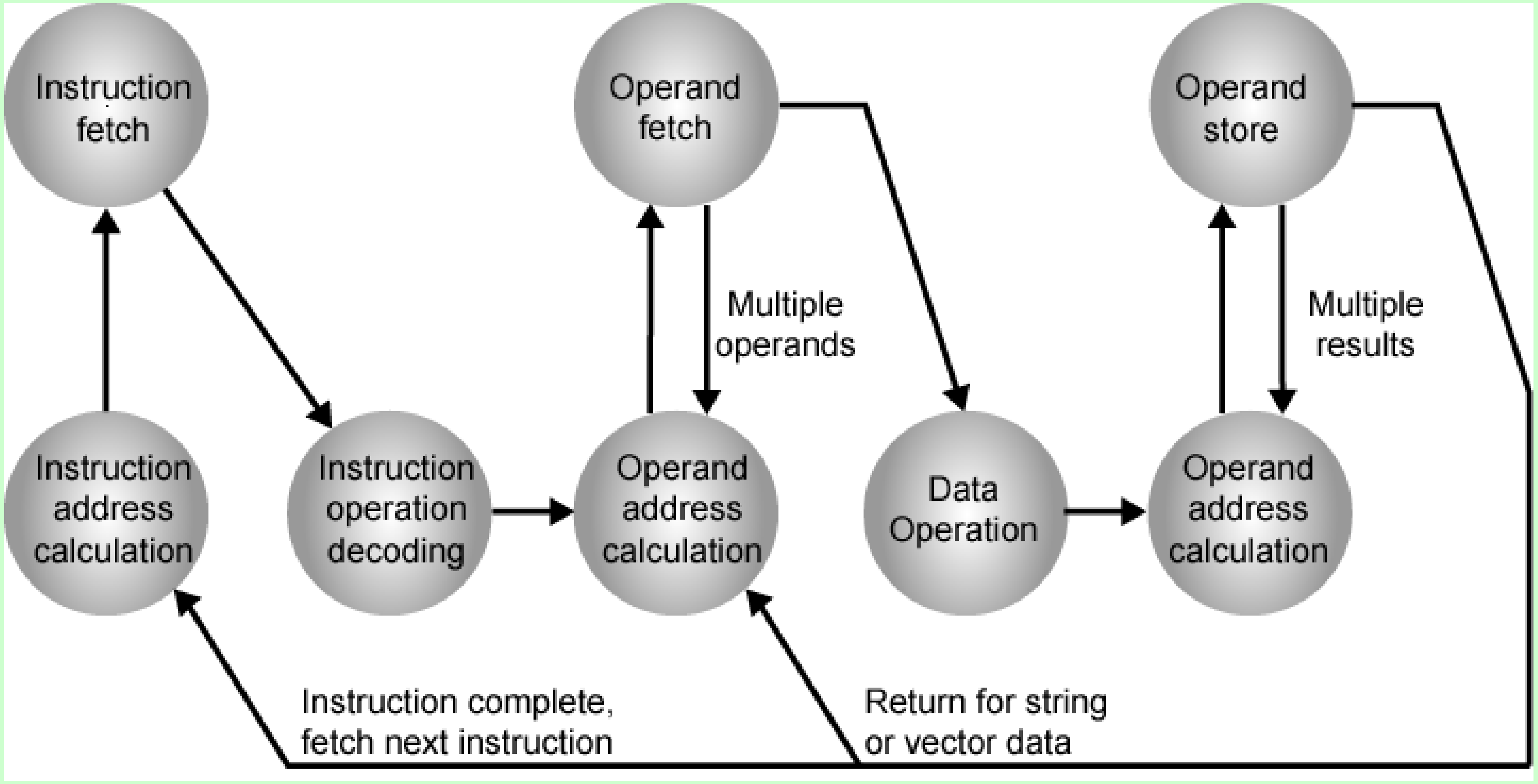


- three instruction cycles
- each consisting of a fetch cycle and an execute cycle, are needed to
- add the contents of location 940 to the contents of 941.

Contoh Lain

- With a more complex set of instructions, fewer cycles would be needed. Some older processors, for example, included instructions that contain more than one memory address. Thus the execution cycle for a particular instruction on such processors could involve more than one reference to memory. Also, instead of memory references, an instruction may specify an I/O operation.
- For example, the PDP-11 processor includes an instruction, expressed symbolically as ADD B,A, that stores the sum of the contents of memory locations B and A into memory location A.
- A single instruction cycle with the following steps occurs:
  - Fetch the ADD instruction.
  - Read the contents of memory location A into the processor.
  - Read the contents of memory location B into the processor. In order that the contents of A are not lost, the processor must have at least two registers for storing memory values, rather than a single accumulator.
  - Add the two values.
  - Write the result from the processor to memory location A.

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
Instruction complete, fetch next instruction
Return for string or vector data

States (1)

- **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.

States (2)

- **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.

States (3)

- **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.

States in the upper part involve an exchange between the processor and either memory or an I/O module. States in the lower part of the diagram involve only internal processor operations. The oac state appears twice, because an instruction may involve a read, a write, or both. However, the action performed during that state is fundamentally the same in both cases, and so only a single state identifier is needed.

Also note that the diagram allows for multiple operands and multiple results, because some instructions on some machines require this. For example, the PDP-11 instruction ADD A,B results in the following sequence of states: iac, if, iod, oac, of, oac, of, do, oac, os.

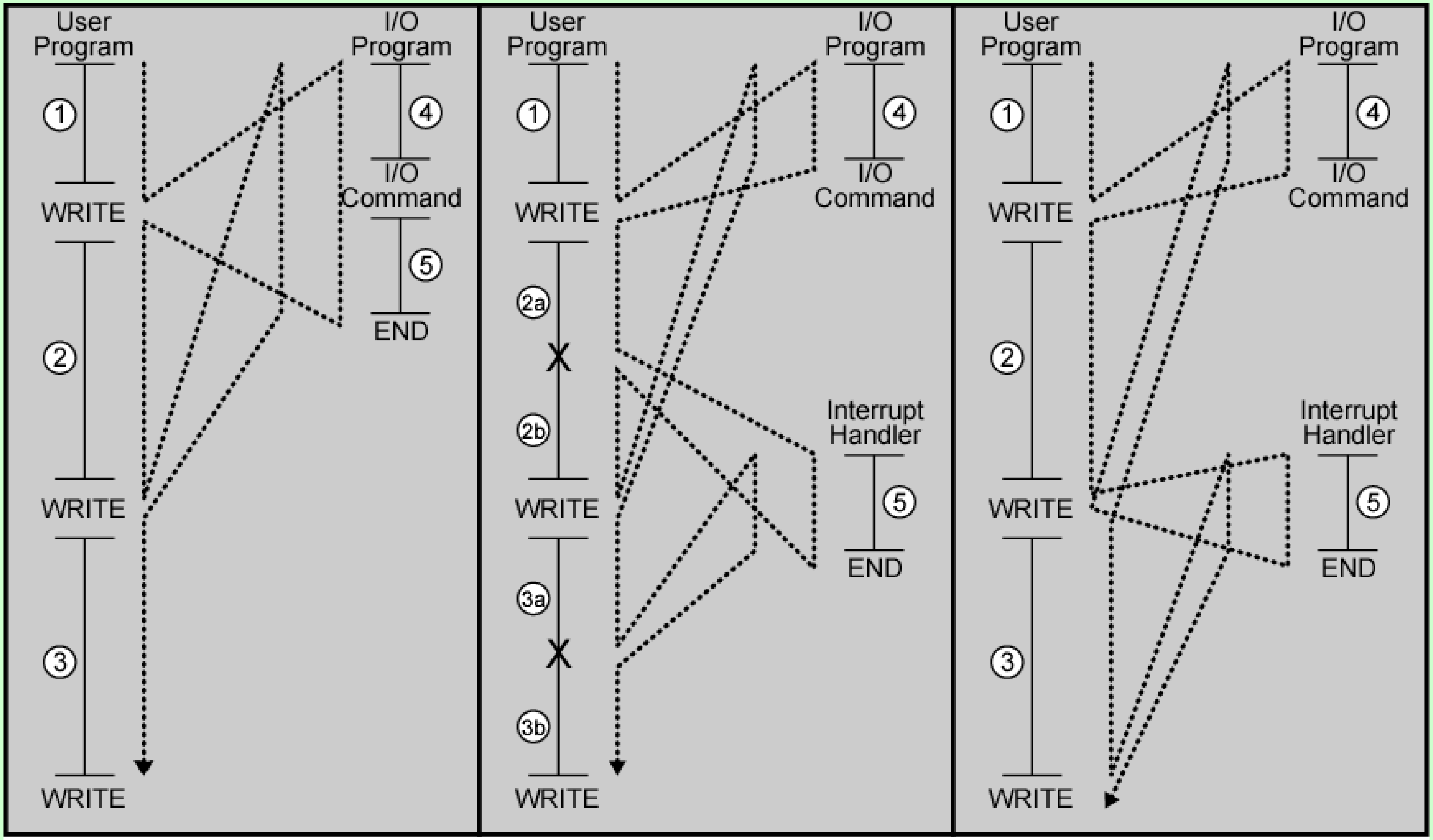
Interrupts

- Menonton Netflix, telp rumah, bel rumah berbunyi,

Mechanism by which other modules (e.g. I/O) may interrupt normal sequence of processing

- Program
  - e.g. overflow, division by zero
- Timer
  - Generated by internal processor timer
  - Used in pre-emptive multi-tasking
- I/O
  - from I/O controller
- Hardware failure
  - e.g. memory parity error

Program Flow Control

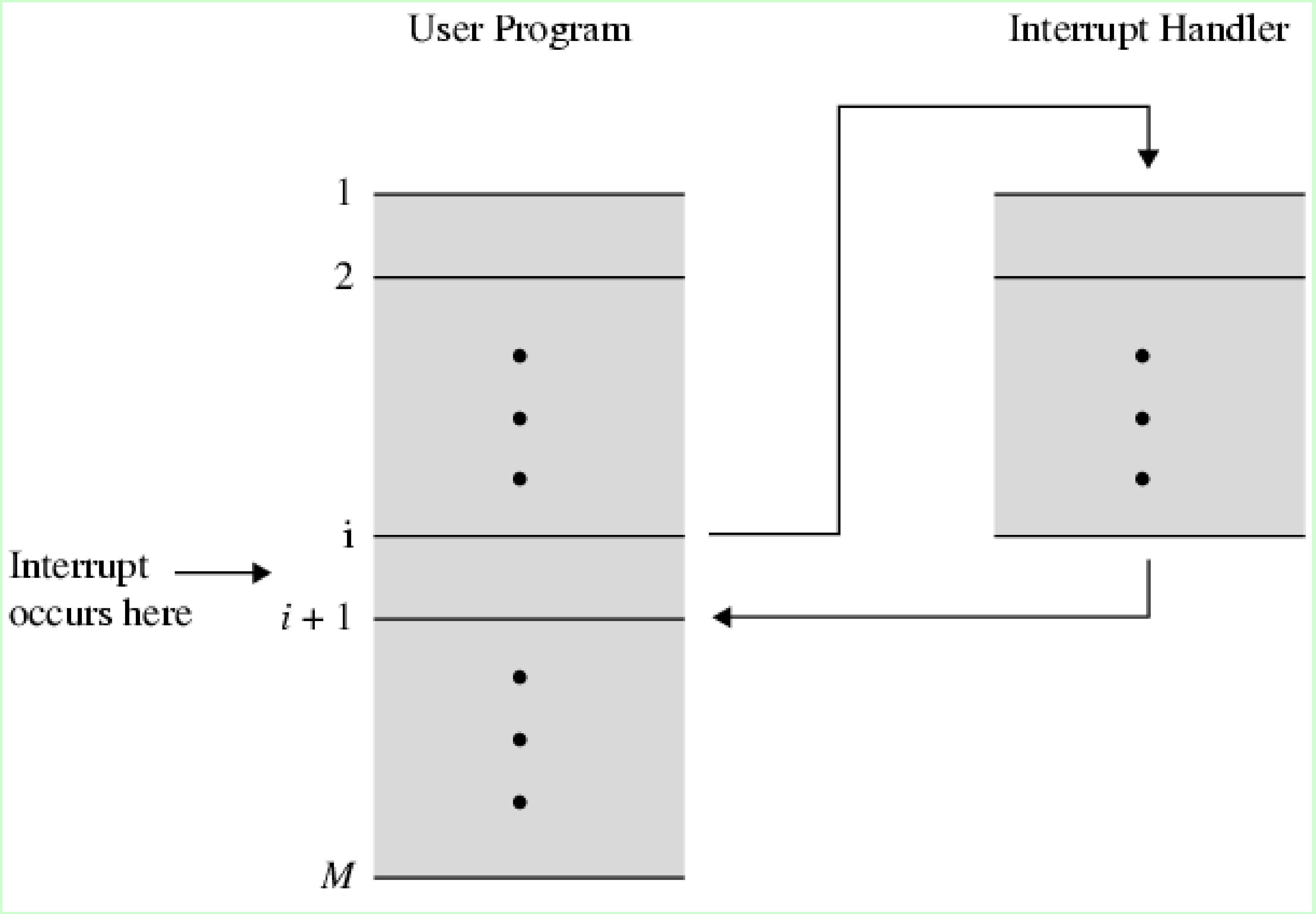


(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

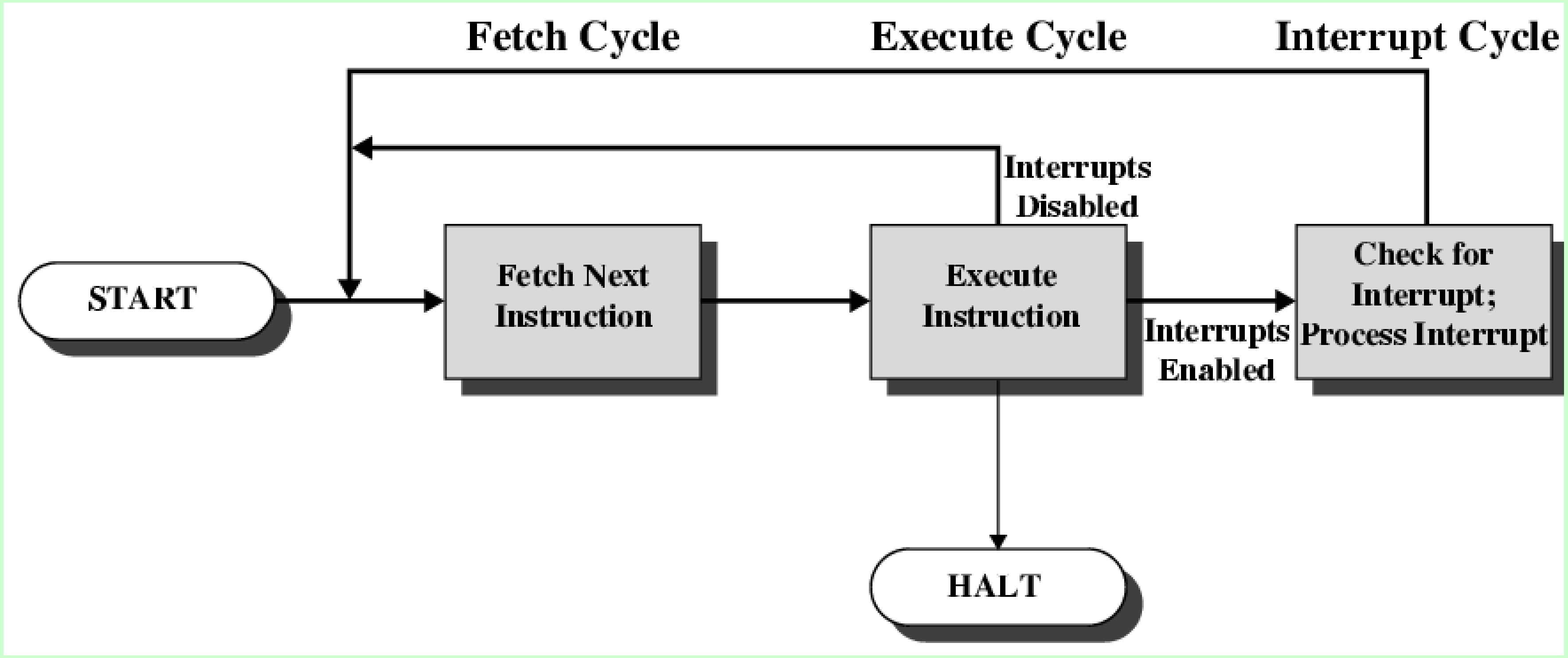
Interrupt Cycle

- Added to instruction cycle
- Processor checks for interrupt
  - Indicated by an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending:
  - Suspend execution of current program
  - Save context
  - Set PC to start address of interrupt handler routine
  - Process interrupt
  - Restore context and continue interrupted program

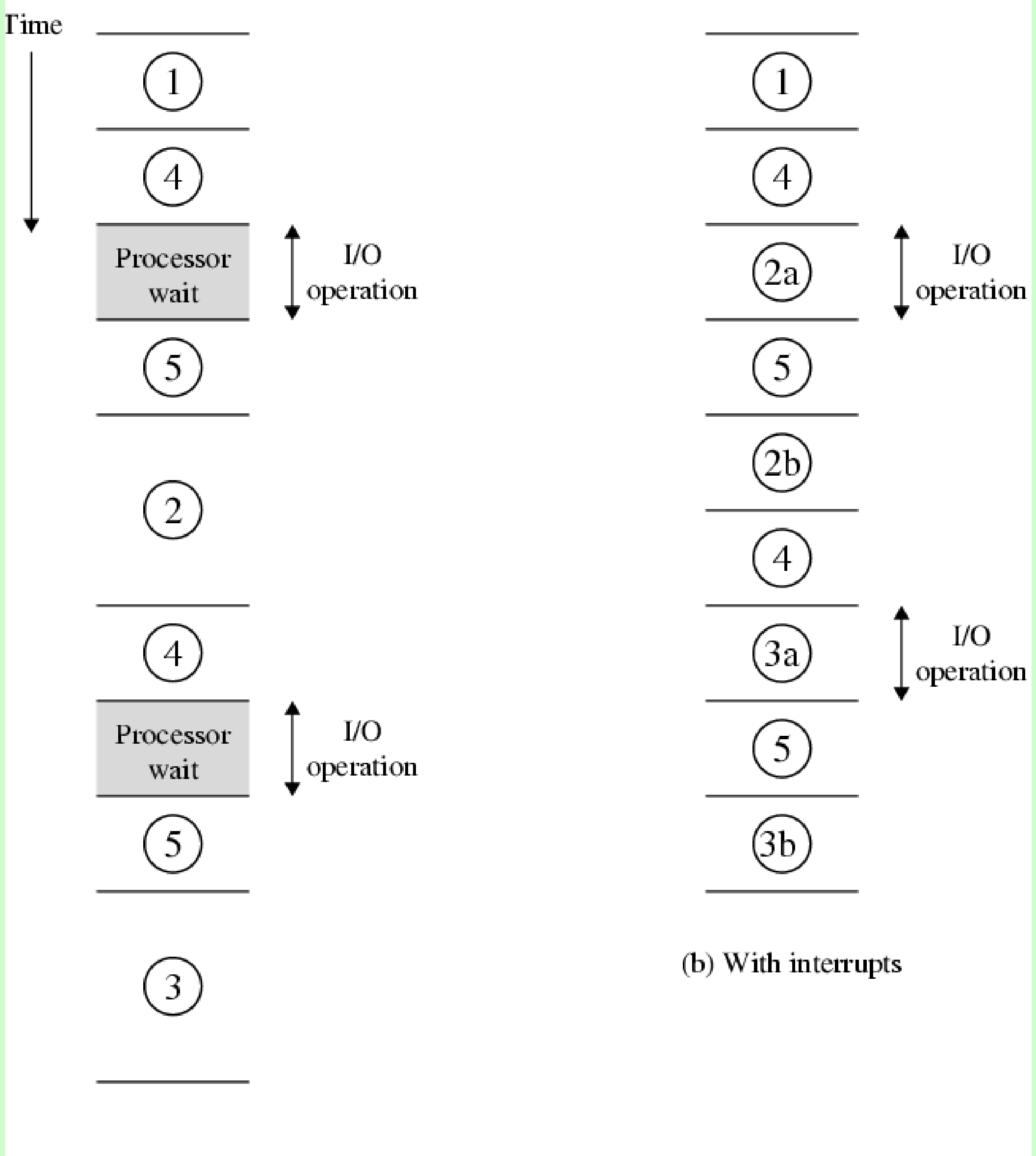
Transfer of Control via Interrupts



Instruction Cycle with Interrupts



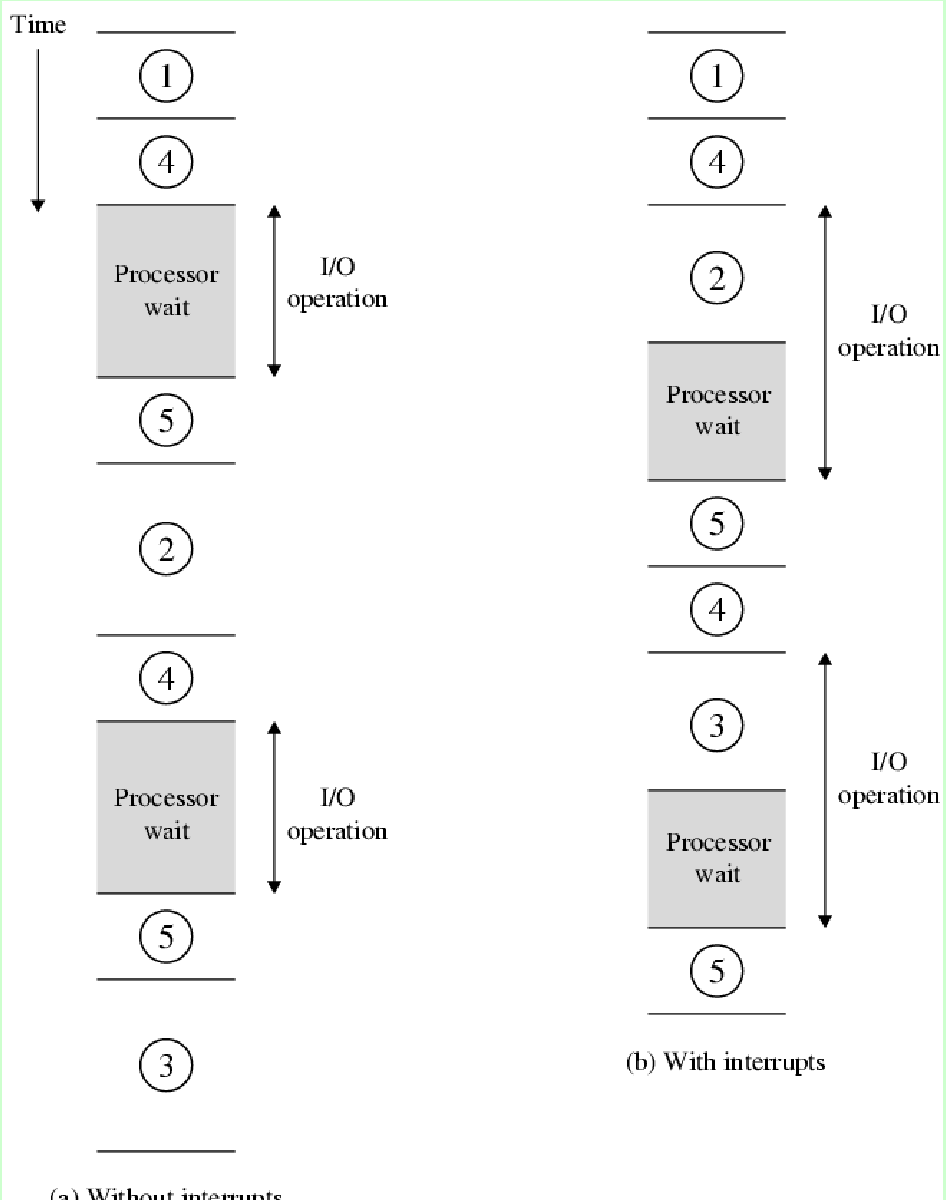
Program Timing
Short I/O Wait



(a) Without interrupts

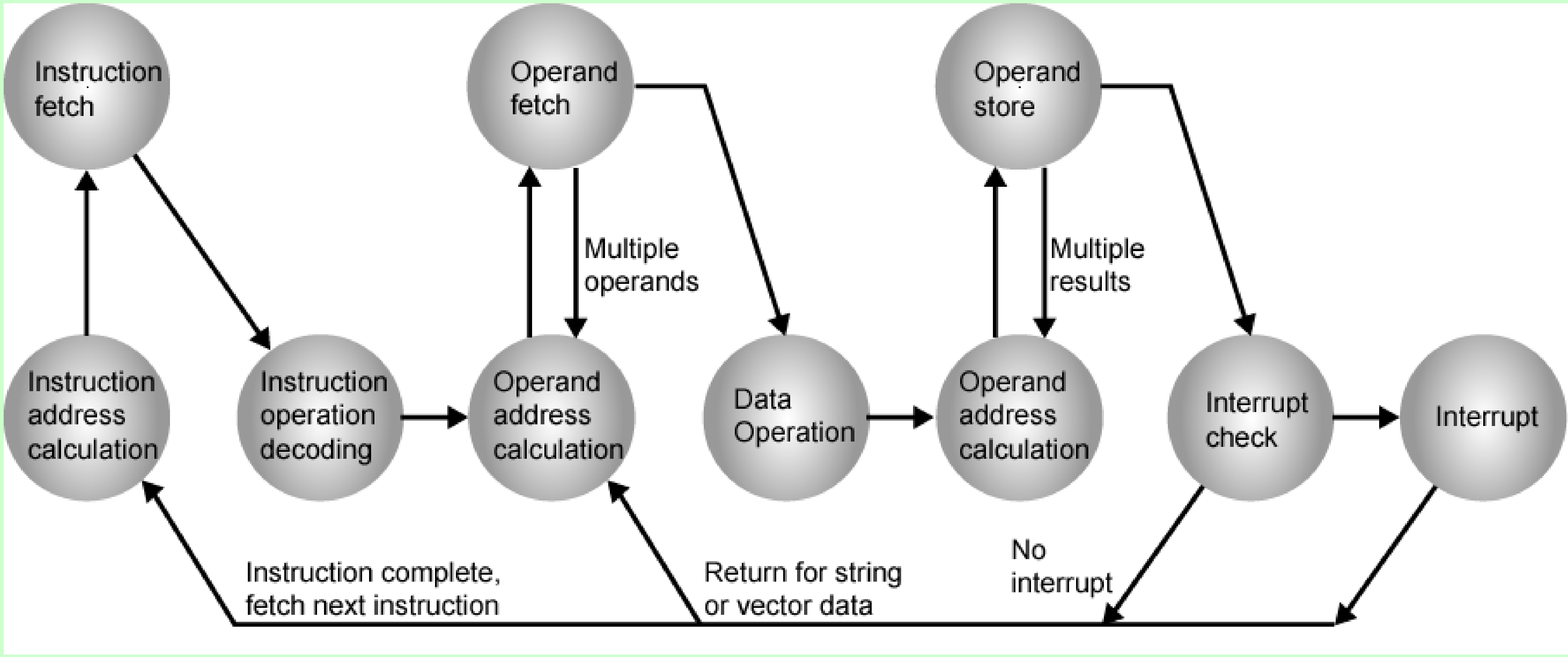
(b) With interrupts

Program Timing
Long I/O Wait



(a) Without interrupts
(b) With interrupts

Instruction Cycle (with Interrupts) - State Diagram

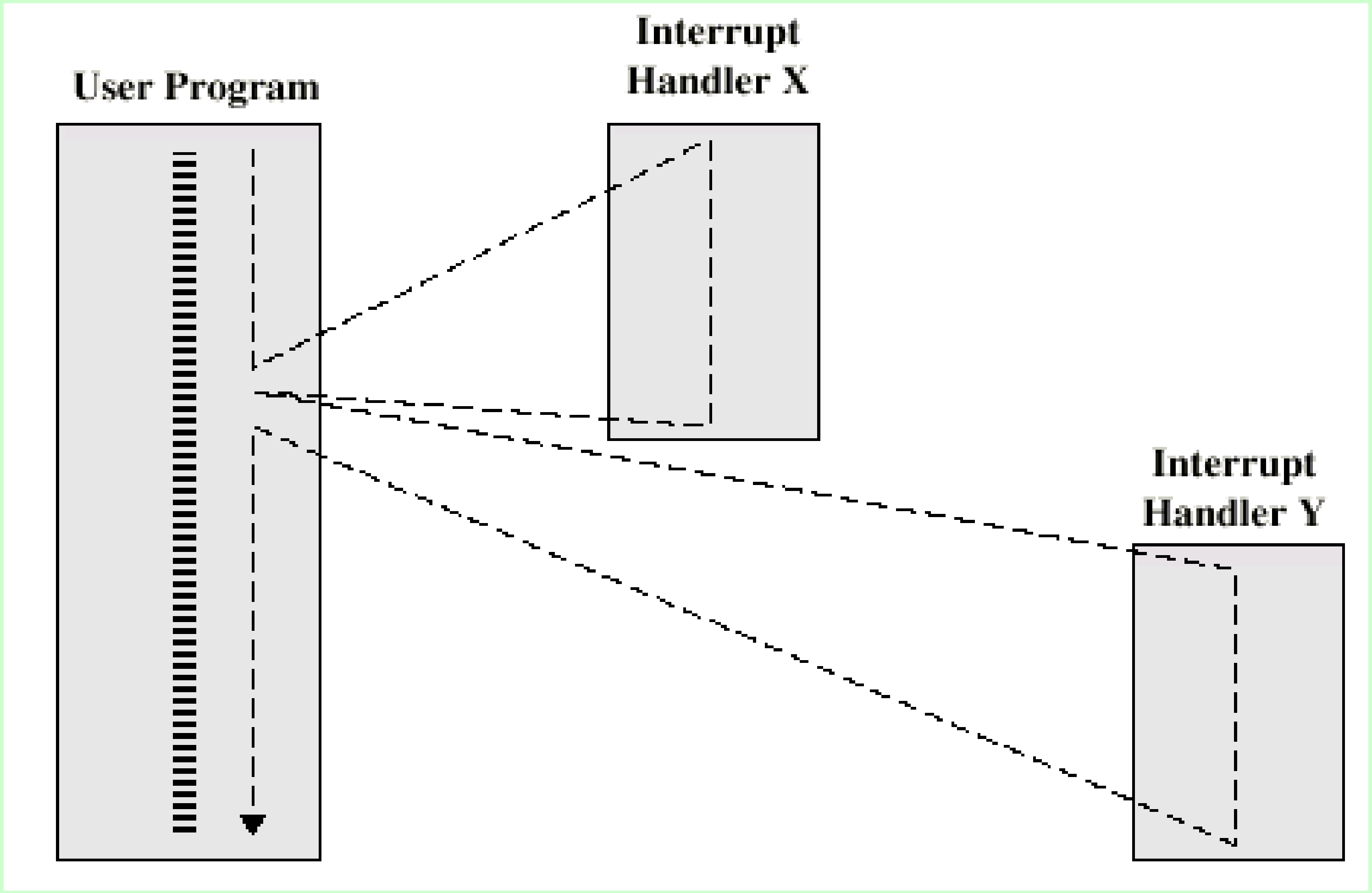


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Interrupt check
Interrupt
No interrupt
Instruction complete, fetch next instruction
Return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

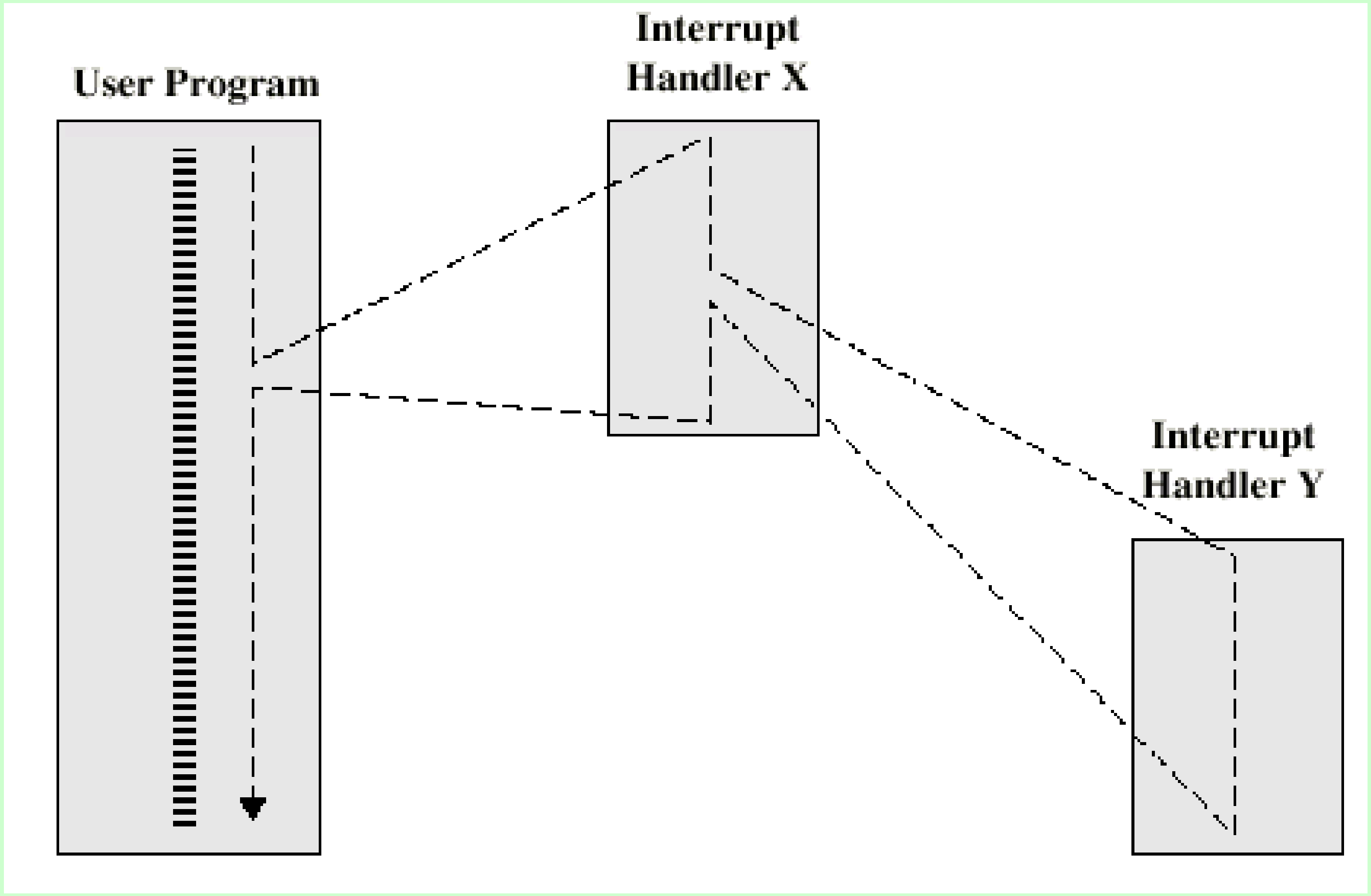
Multiple Interrupts

- Disable interrupts
  - Processor will ignore further interrupts whilst processing one interrupt
  - Interrupts remain pending and are checked after first interrupt has been processed
  - Interrupts handled in sequence as they occur
- Define priorities
  - Low priority interrupts can be interrupted by higher priority interrupts
  - When higher priority interrupt has been processed, processor returns to previous interrupt
- Mengapa perlu ada interrupt?
- Mengapa perlu ada prioritas interrupt?

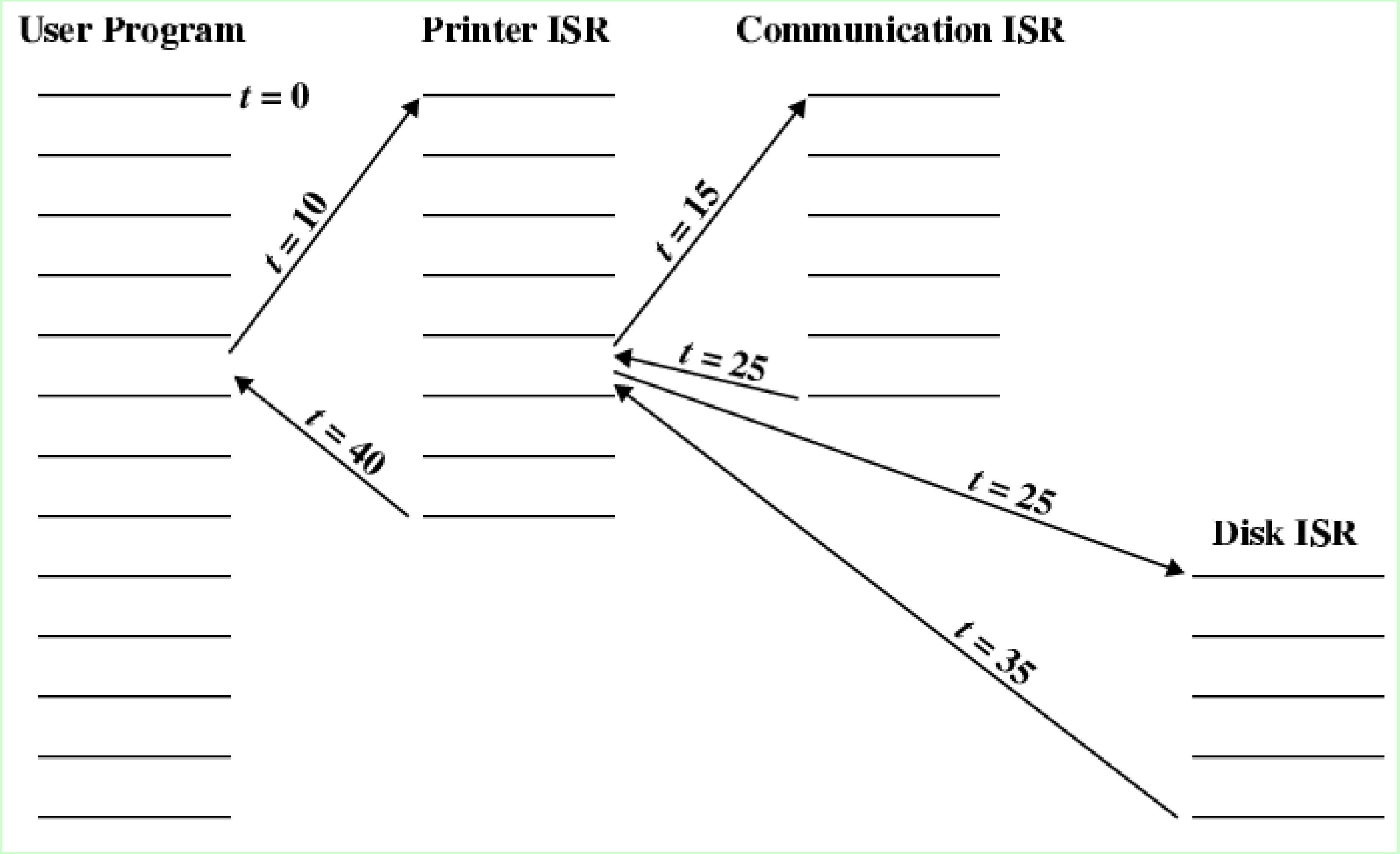
Multiple Interrupts - Sequential



Multiple Interrupts – Nested



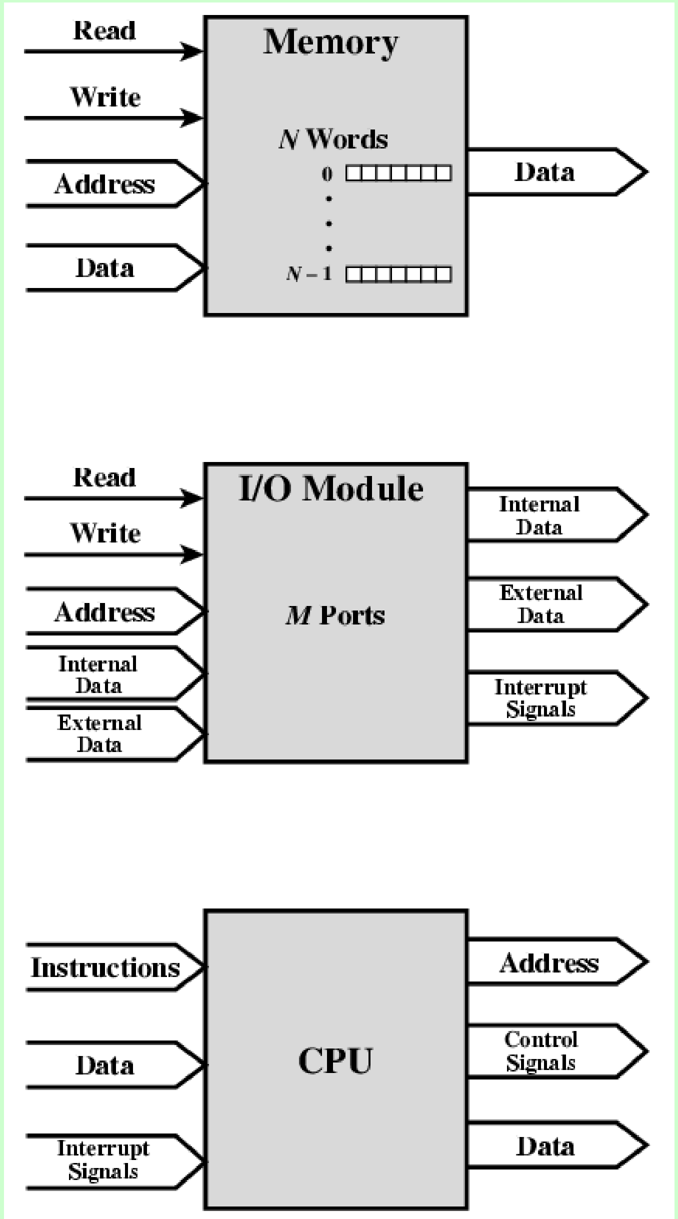
Time Sequence of Multiple Interrupts



Connecting

- All the units must be connected
- Different type of connection for different type of unit
  - Memory
    - R/W
    - alamat
  - Input/Output
    - R/W
    - Port
    - High speed, low speed
  - CPU
    - Control signal
    - interrupt

Computer Modules



Memory Connection

- Receives and sends data
- Receives addresses (of locations)
- Receives control signals
  - Read
  - Write
  - Timing

Input/Output Connection(1)

- Similar to memory from computer’s viewpoint
- Output
  - Receive data from computer
  - Send data to peripheral
- Input
  - Receive data from peripheral
  - Send data to computer

Input/Output Connection(2)

- Receive control signals from computer
- Send control signals to peripherals
    - e.g. spin disk
- Receive addresses from computer
    - e.g. port number to identify peripheral
- Send interrupt signals (control)

CPU Connection

- Reads instruction and data
- Writes out data (after processing)
- Sends control signals to other units
- Receives (& acts on) interrupts

Buses

- There are a number of possible interconnection systems
- Single and multiple BUS structures are most common
- e.g. Control/Address/Data bus (PC)
- e.g. Unibus (DEC-PDP)

What is a Bus?

- A communication pathway connecting two or more devices
- Usually broadcast
- Often grouped
  - A number of channels in one bus
  - e.g. 32 bit data bus is 32 separate single bit channels

Data Bus

- Carries data
  - Remember that there is no difference between “data” and “instruction” at this level
- Width is a key determinant of performance
  - 8, 16, 32, 64 bit

Address bus

- Identify the source or destination of data
- e.g. CPU needs to read an instruction (data) from a given location in memory
- Bus width determines maximum memory capacity of system
  - e.g. 8080 has 16 bit address bus giving 64k address space

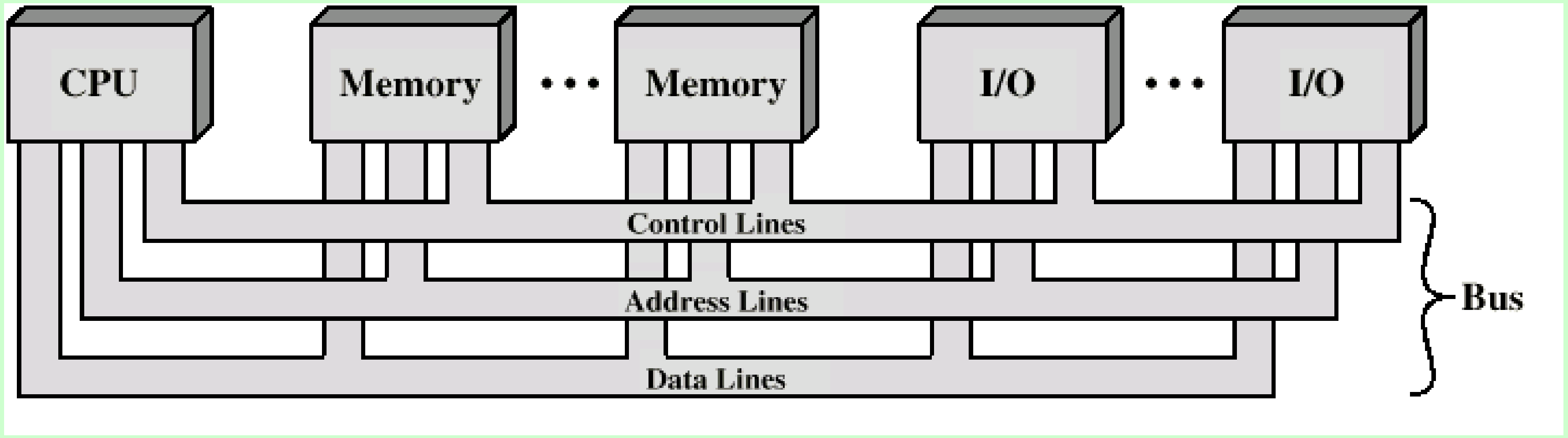
Bus Width

- The width of the data bus has an impact on system performance: The wider the data bus, the greater the number of bits transferred at one time.
- The width of the address bus has an impact on system capacity: the wider the address bus, the greater the range of locations that can be referenced

Control Bus

- Control and timing information
    - Memory read/write signal
    - Interrupt request
    - Clock signals

Bus Interconnection Scheme



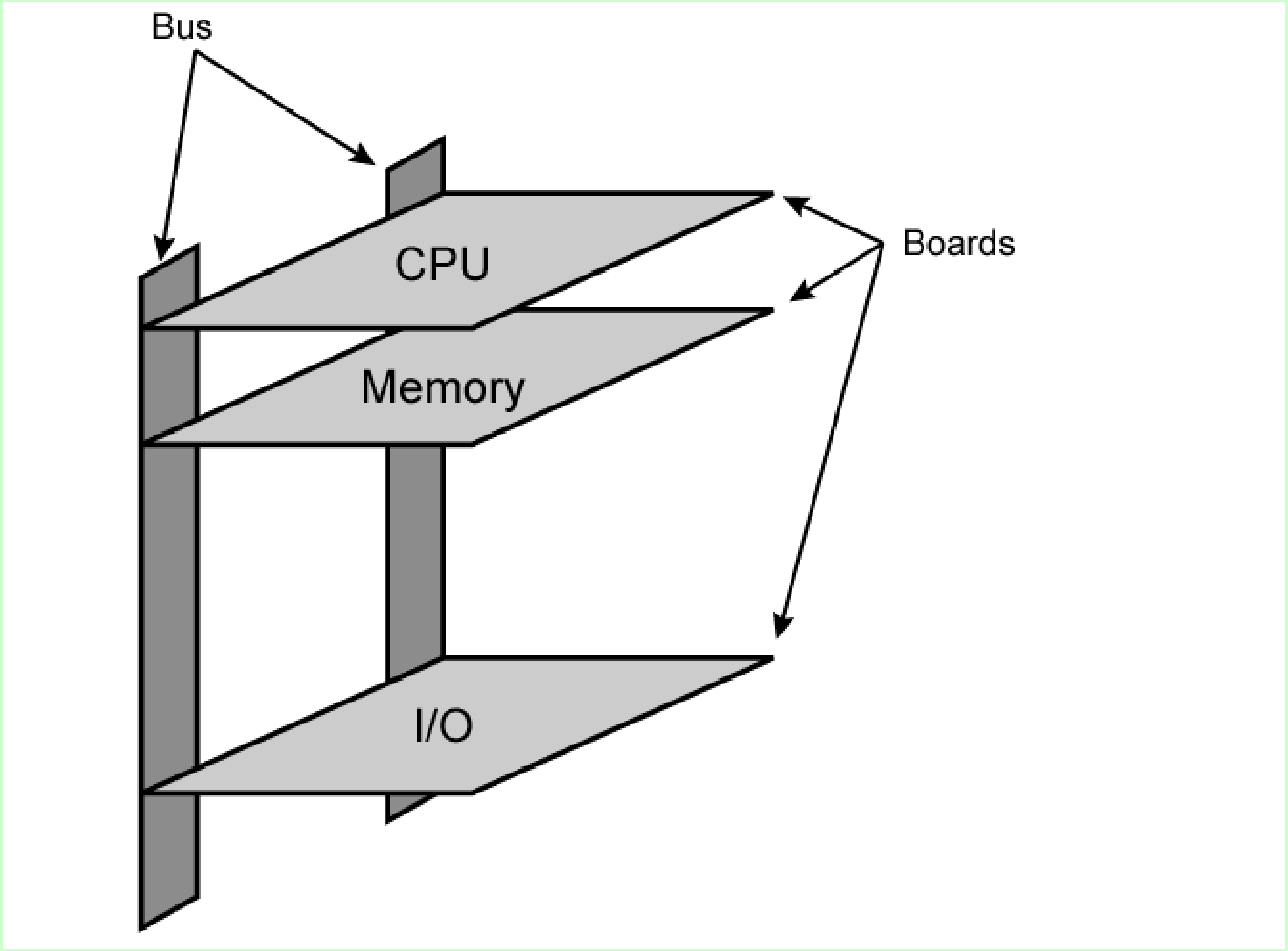
Big and Yellow?

What do buses look like?

- Parallel lines on circuit boards
- Ribbon cables
- Strip connectors on mother boards
  - e.g. PCI
- Sets of wires

- USB
  - Universal Serial Bus
  - USB-C → Type C

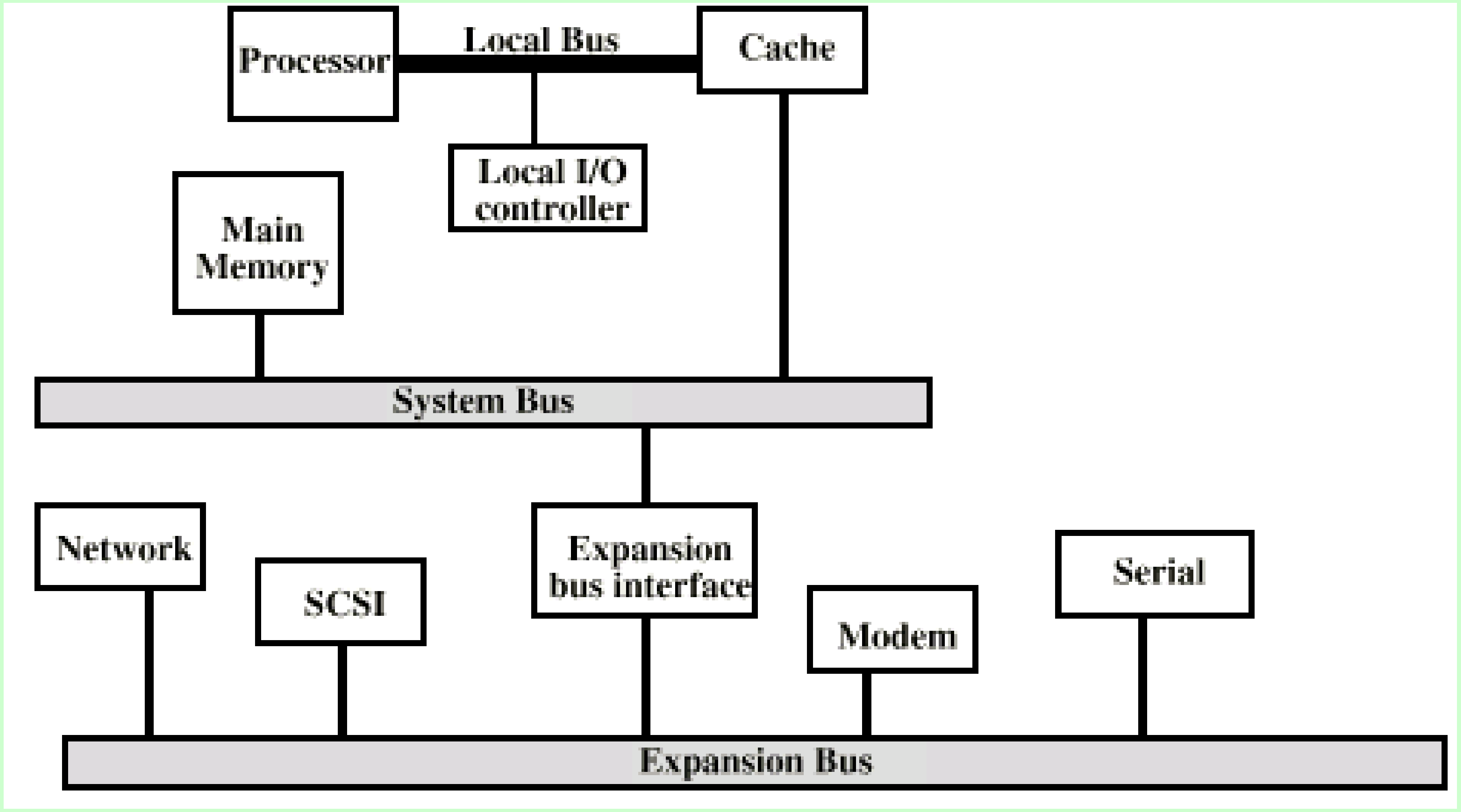
Physical Realization of Bus Architecture



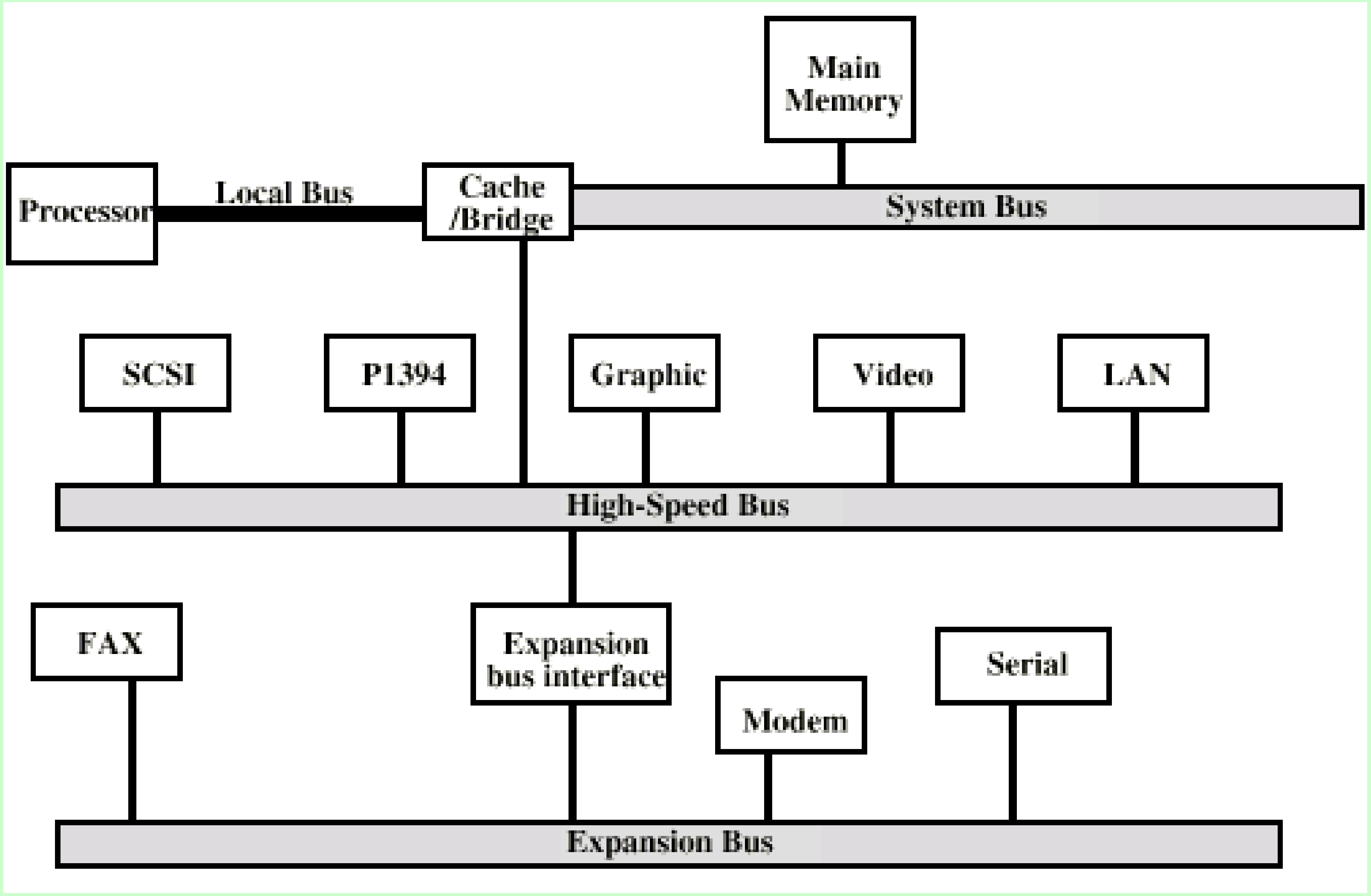
Single Bus Problems

- Lots of devices on one bus leads to:
  - Propagation delays
    - Long data paths mean that co-ordination of bus use can adversely affect performance
    - If aggregate data transfer approaches bus capacity
- Most systems use multiple buses to overcome these problems

Traditional (ISA) (with cache)



High Performance Bus



Bus Types

- Dedicated
  - Separate data & address lines
- Multiplexed
  - Shared lines
  - Address valid or data valid control line
  - Advantage - fewer lines
  - Disadvantages
    - More complex control
    - Ultimate performance

Single Bus Read Write

- All buses support both write (master to slave) and read (slave to master) transfers.
- In the case of a multiplexed address/data bus (single bus), the bus is first used for specifying the address and then for transferring the data.
- For a read operation, there is typically a wait while the data are being fetched from the slave to be put on the bus. For either a read or a write, there may also be a delay if it is necessary to go through arbitration to gain control of the bus for the remainder of the operation (i.e., seize the bus to request a read or write, then seize the bus again to perform a read or write)

Dedicated Bus.

- Data bus dan Adress bus. Bus data dan bus Alamat masing-masing
- In the case of dedicated address and data buses, the address is put on the address bus and remains there while the data are put on the data bus. For a write operation, the master puts the data onto the data bus as soon as the address has stabilized and the slave has had the opportunity to recognize its address. For a read operation, the slave puts the data onto the data bus as soon as it has recognized its address and has fetched the data.

Bus Arbitration

- More than one module controlling the bus
- e.g. CPU and DMA controller
- Only one module may control bus at one time
- Arbitration may be centralised or distributed

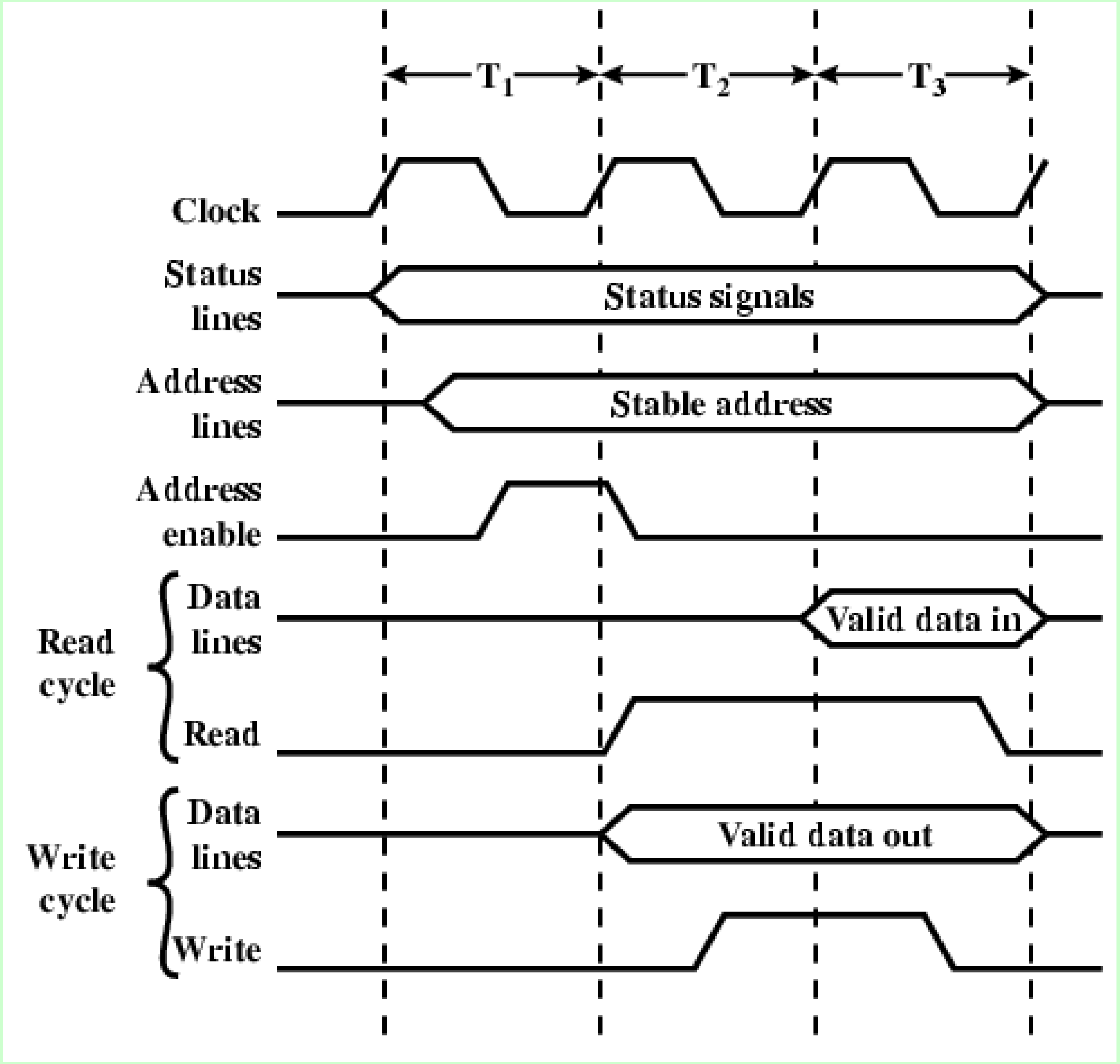
Centralised or Distributed Arbitration

- Centralised
  - Single hardware device controlling bus access
    - Bus Controller
    - Arbiter
  - May be part of CPU or separate
- Distributed
  - Each module may claim the bus
  - Control logic on all modules

Timing

- Co-ordination of events on bus
- Synchronous
  - Events determined by clock signals
  - Control Bus includes clock line
  - A single 1-0 is a bus cycle
  - All devices can read clock line
  - Usually sync on leading edge
  - Usually a single cycle for an event

Synchronous Timing Diagram



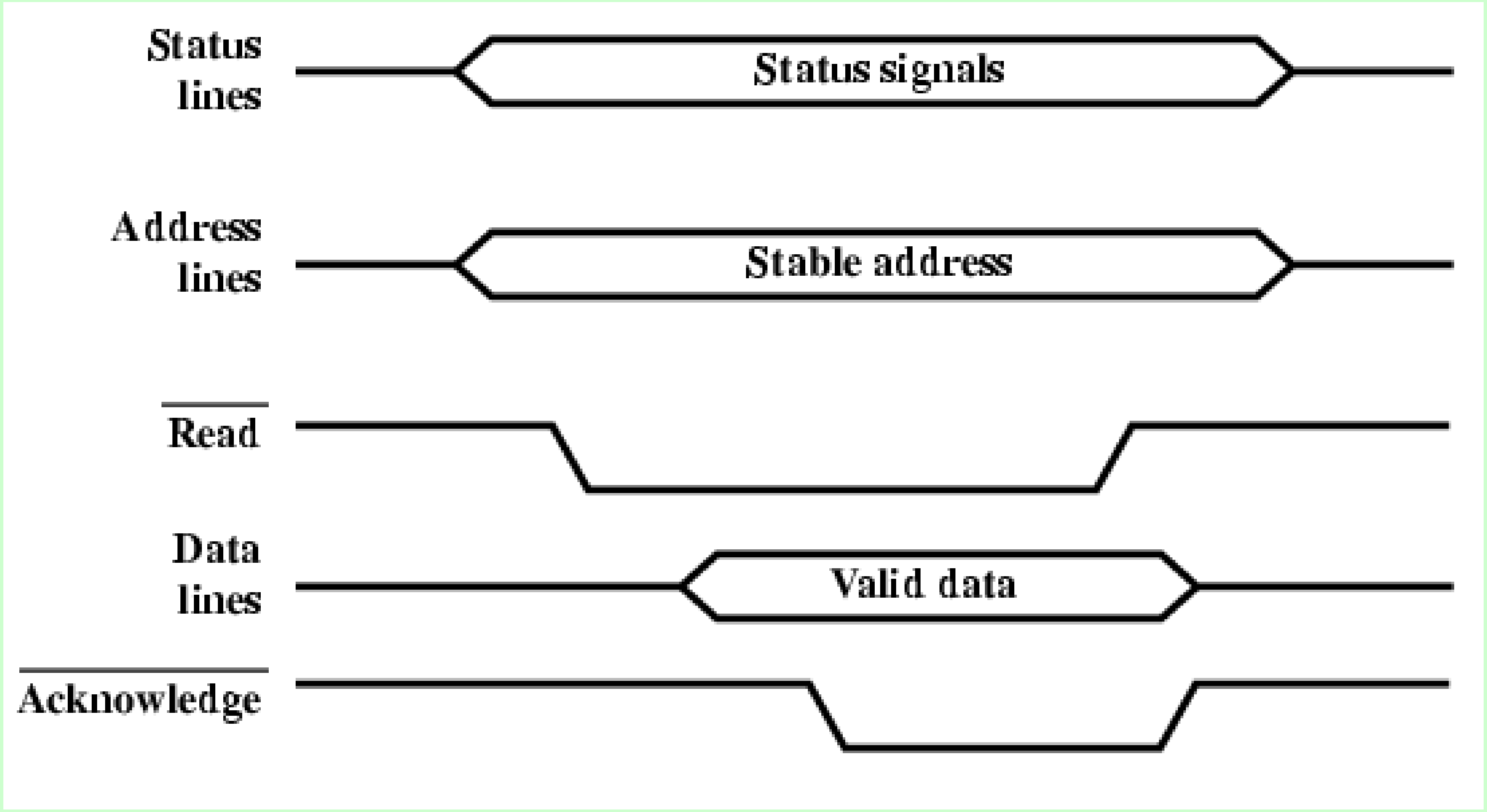
Pewaktuan Sinkron

- With synchronous timing, the occurrence of events on the bus is determined by a clock.
- The bus includes a clock line upon which a clock transmits a regular sequence of alternating 1s and 0s of equal duration. A single 1–0 transmission is referred to as a clock cycle or bus cycle and defines a time slot.

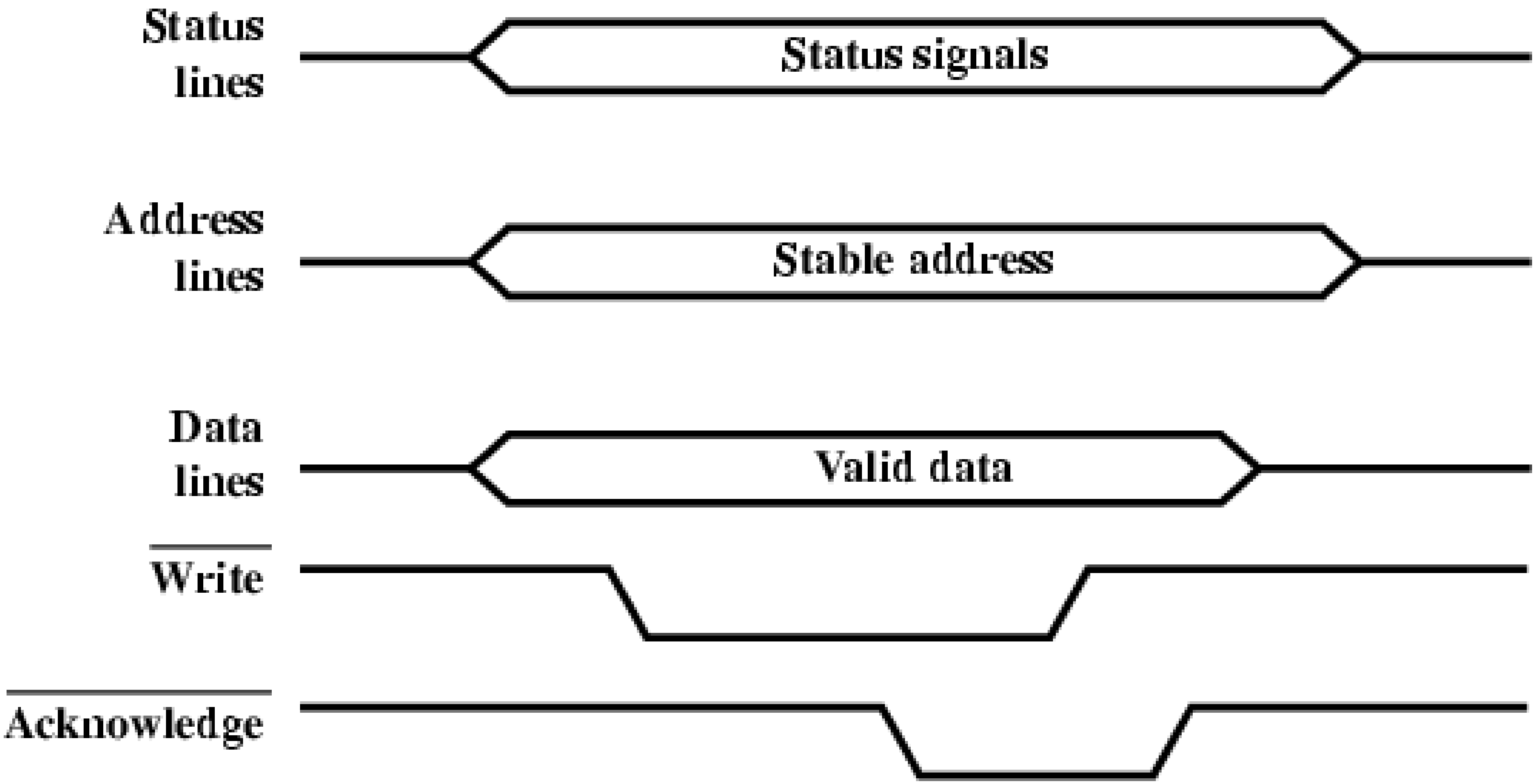
Pewaktuan Asinkron

- With asynchronous timing, the occurrence of one event on a bus follows and depends on the occurrence of a previous event.
- In the simple read example async timing, the processor places address and status signals on the bus. After pausing for these signals to stabilize, it issues a read command,indicating the presence of valid address and control signals. The appropriate memory decodes the address and responds by placing the data on the data line.
- Once the data lines have stabilized, the memory module asserts the acknowledged line to signal the processor that the data are available. Once the master has read the data from the data lines, it deasserts the read signal. This causes the memory module to drop the data and acknowledge lines. Finally, once the acknowledge line is dropped, the master removes the address information

Asynchronous Timing – Read Diagram



Asynchronous Timing – Write Diagram



PCI Bus

- Peripheral Component Interconnection
- Intel released to public domain
- 32 or 64 bit
- 50 lines

PCI Bus Lines (required)

- Systems lines
  - Including clock and reset
- Address & Data
  - 32 time mux lines for address/data
  - Interrupt & validate lines
- Interface Control
- Arbitration
  - Not shared
  - Direct connection to PCI bus arbiter
- Error lines

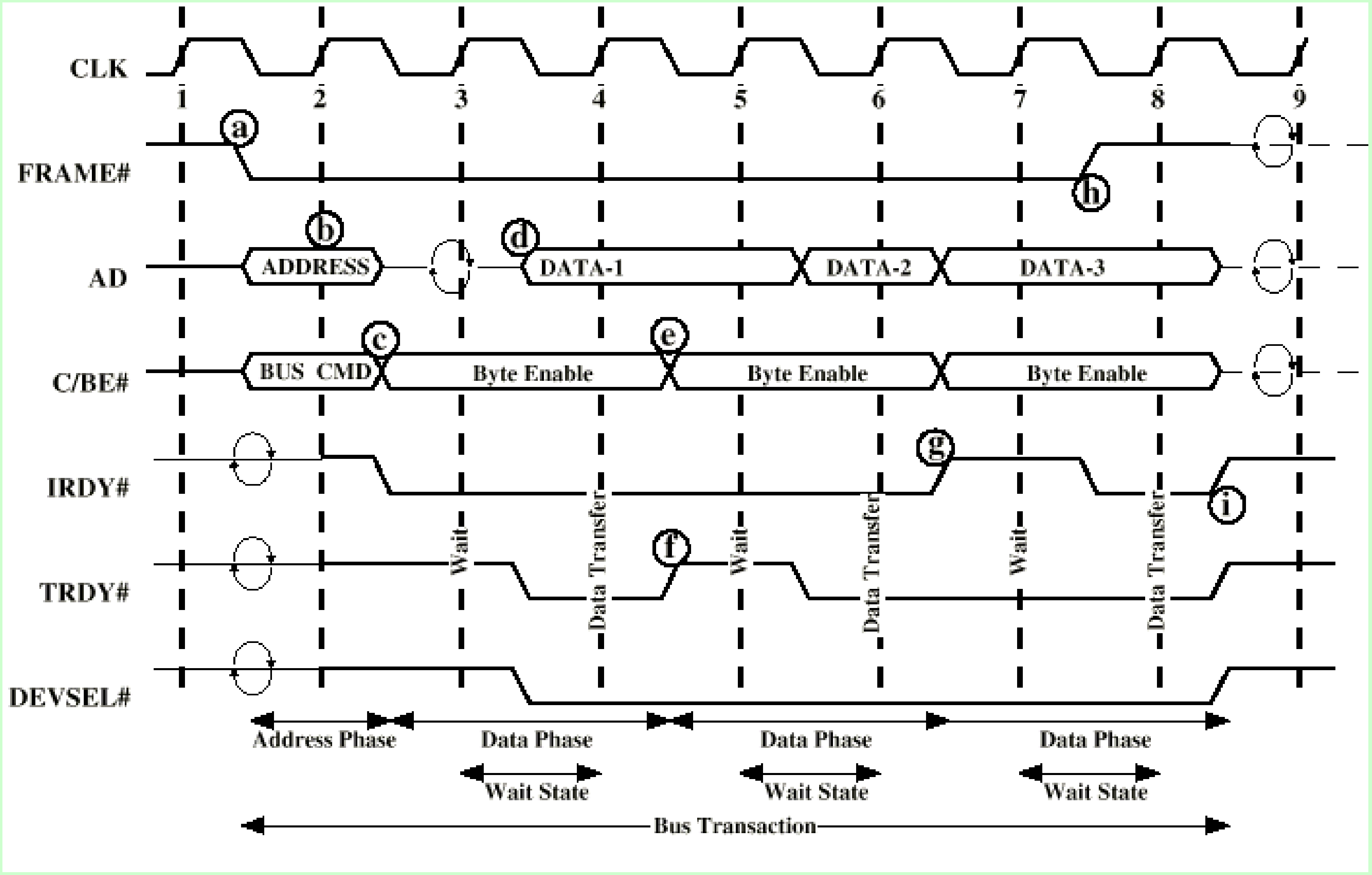
PCI Bus Lines (Optional)

- Interrupt lines
  - Not shared
- Cache support
- 64-bit Bus Extension
  - Additional 32 lines
  - Time multiplexed
  - 2 lines to enable devices to agree to use 64-bit transfer
- JTAG/Boundary Scan
  - For testing procedures

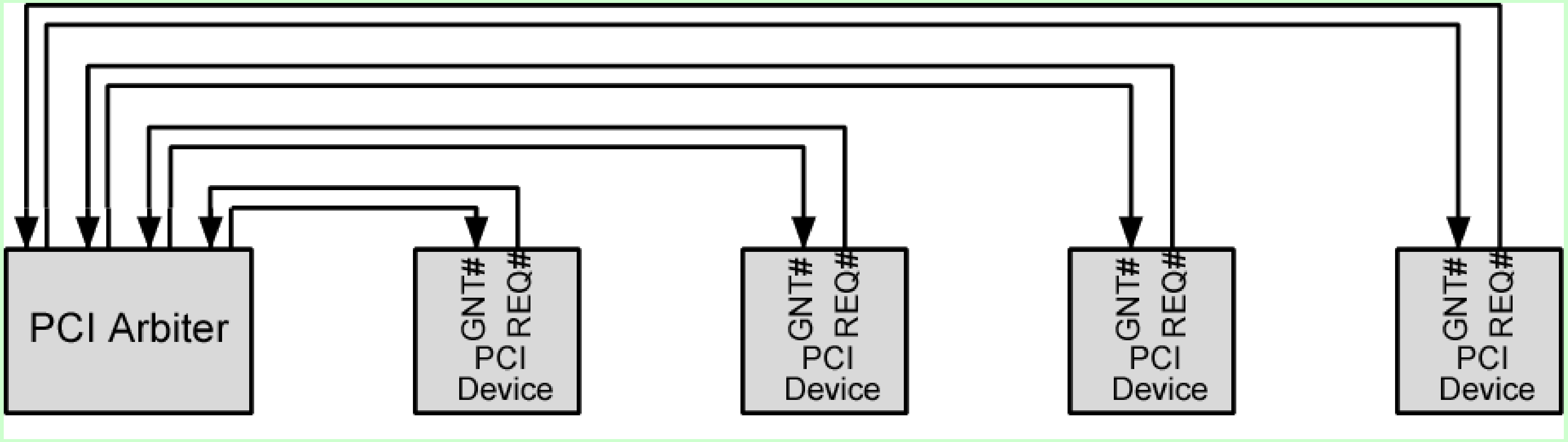
PCI Commands

- Transaction between initiator (master) and target
- Master claims bus
- Determine type of transaction
  - e.g. I/O read/write
- Address phase
- One or more data phases

PCI Read Timing Diagram



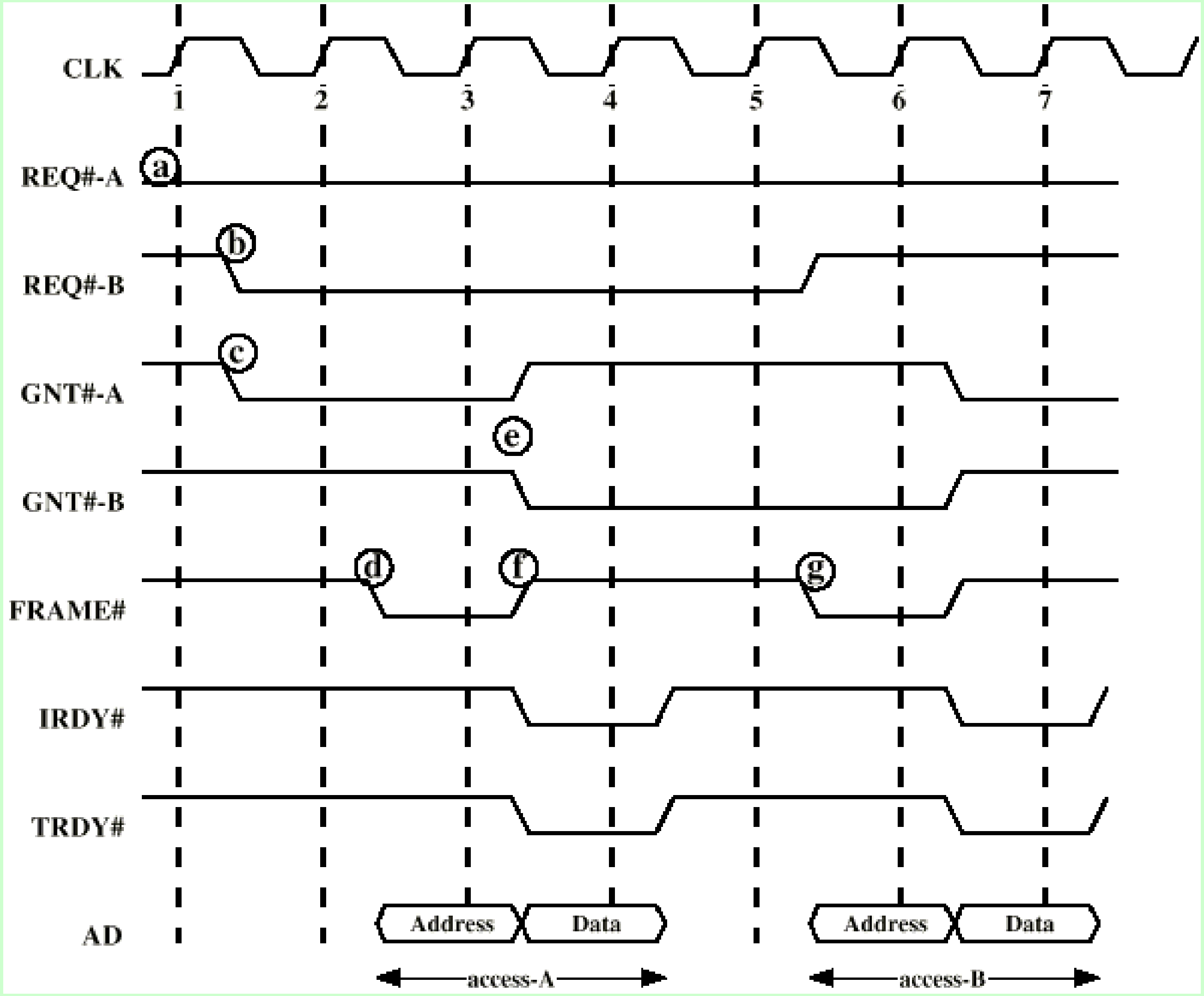
PCI Bus Arbiter



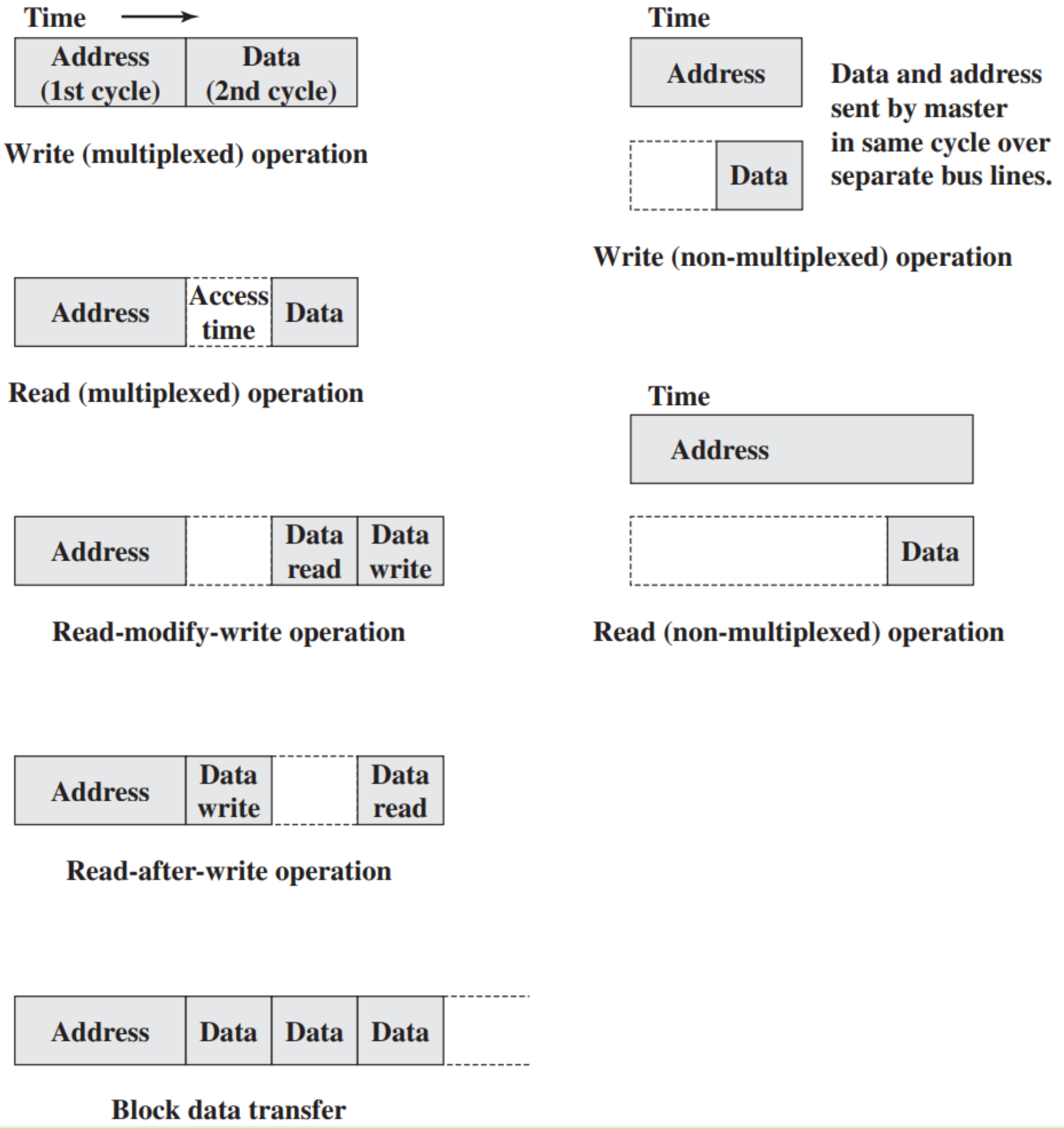
Arbiter: Penengah

In all but the simplest systems, more than one module may need control of the bus. For example, an I/O module may need to read or write directly to memory, without sending the data to the processor. Because only one unit at a time can successfully transmit over the bus, some method of arbitration is needed. In a centralized scheme, a single hardware device, referred to as a bus controller or arbiter, is responsible for allocating time on the bus.

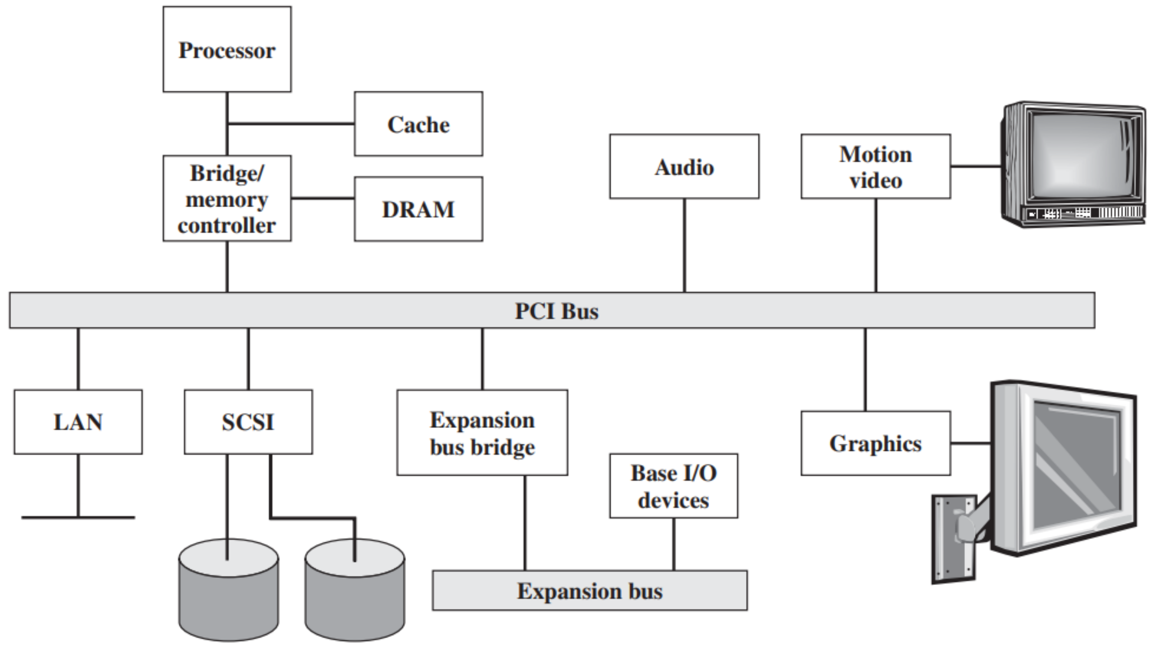
PCI Bus Arbitration



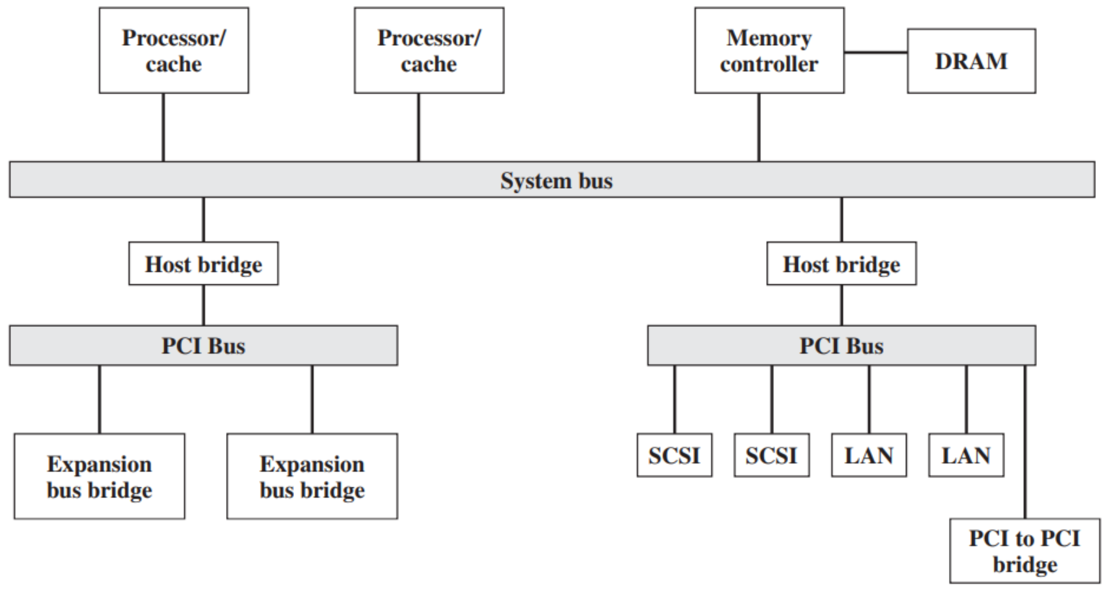
Bus Data Transfer Type


Time
Address (1st cycle)
Data (2nd cycle)
Write (multiplexed) operation
Address
Access time
Data
Read (multiplexed) operation
Address
Data read
Data write
Read-modify-write operation
Address
Data write
Data read
Read-after-write operation
Address
Data
Data
Data
Block data transfer
Time
Address
Data
Data and address sent by master in same cycle over separate bus lines.
Write (non-multiplexed) operation
Time
Address
Data
Read (non-multiplexed) operation

Bus Tipikal



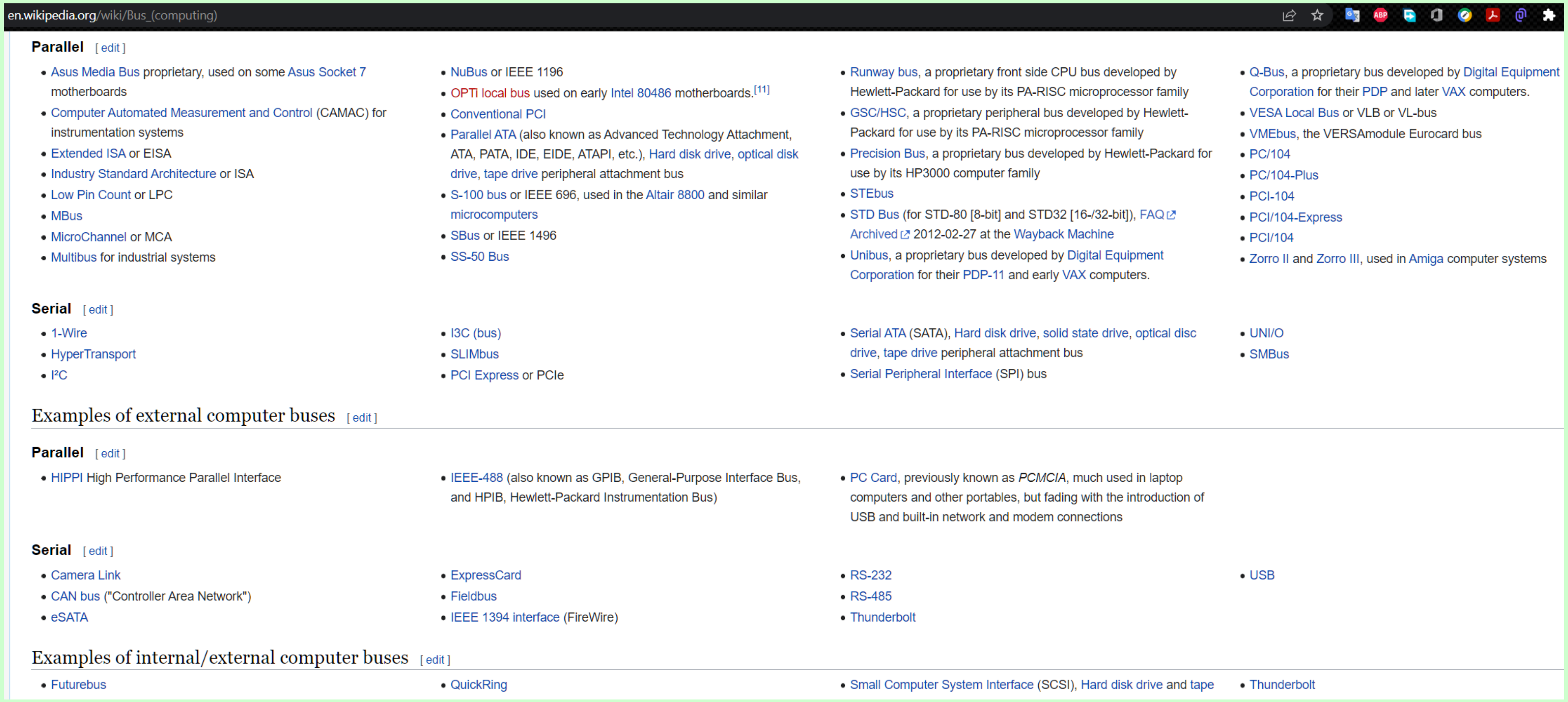
(a) Typical desktop system



(b) Typical server system

Foreground Reading

- https://en.wikipedia.org/wiki/Bus_(computing)
- https://en.wikipedia.org/wiki/USB-C
- https://en.wikipedia.org/wiki/Peripheral_Component_Interconnect
- https://en.wikipedia.org/wiki/Accelerated_Graphics_Port


en.wikipedia.org/wiki/Bus_(computing)

Parallel [edit]

- Asus Media Bus proprietary, used on some Asus Socket 7 motherboards
- Computer Automated Measurement and Control (CAMAC) for instrumentation systems
- Extended ISA or EISA
- Industry Standard Architecture or ISA
- Low Pin Count or LPC
- MBus
- MicroChannel or MCA
- Multibus for industrial systems
- NuBus or IEEE 1196
- OPTi local bus used on early Intel 80486 motherboards.[11]
- Conventional PCI
- Parallel ATA (also known as Advanced Technology Attachment, ATA, PATA, IDE, EIDE, ATAPI, etc.), Hard disk drive, optical disk drive, tape drive peripheral attachment bus
- S-100 bus or IEEE 696, used in the Altair 8800 and similar microcomputers
- SBus or IEEE 1496
- SS-50 Bus
- Runway bus, a proprietary front side CPU bus developed by Hewlett-Packard for use by its PA-RISC microprocessor family
- GSC/HSC, a proprietary peripheral bus developed by Hewlett-Packard for use by its PA-RISC microprocessor family
- Precision Bus, a proprietary bus developed by Hewlett-Packard for use by its HP3000 computer family
- STEbus
- STD Bus (for STD-80 [8-bit] and STD32 [16-/32-bit]), FAQ Archived 2012-02-27 at the Wayback Machine
- Unibus, a proprietary bus developed by Digital Equipment Corporation for their PDP-11 and early VAX computers.
- Q-Bus, a proprietary bus developed by Digital Equipment Corporation for their PDP and later VAX computers.
- VESA Local Bus or VLB or VL-bus
- VMEbus, the VERSAmodule Eurocard bus
- PC/104
- PC/104-Plus
- PCI-104
- PCI/104-Express
- PCI/104
- Zorro II and Zorro III, used in Amiga computer systems

Serial [edit]

- 1-Wire
- HyperTransport
- I²C
- I3C (bus)
- SLIMbus
- PCI Express or PCIe
- Serial ATA (SATA), Hard disk drive, solid state drive, optical disc drive, tape drive peripheral attachment bus
- Serial Peripheral Interface (SPI) bus
- UNI/O
- SMBus

Examples of external computer buses [edit]

Parallel [edit]

- HIPPI High Performance Parallel Interface
- IEEE-488 (also known as GPIB, General-Purpose Interface Bus, and HPIB, Hewlett-Packard Instrumentation Bus)
- PC Card, previously known as *PCMCIA*, much used in laptop computers and other portables, but fading with the introduction of USB and built-in network and modem connections

Serial [edit]

- Camera Link
- CAN bus ("Controller Area Network")
- eSATA
- ExpressCard
- Fieldbus
- IEEE 1394 interface (FireWire)
- RS-232
- RS-485
- Thunderbolt
- USB

Examples of internal/external computer buses [edit]

- Futurebus
- QuickRing
- Small Computer System Interface (SCSI), Hard disk drive and tape
- Thunderbolt