William Stallings
Computer Organization and Architecture
8th Edition

Chapter 2

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

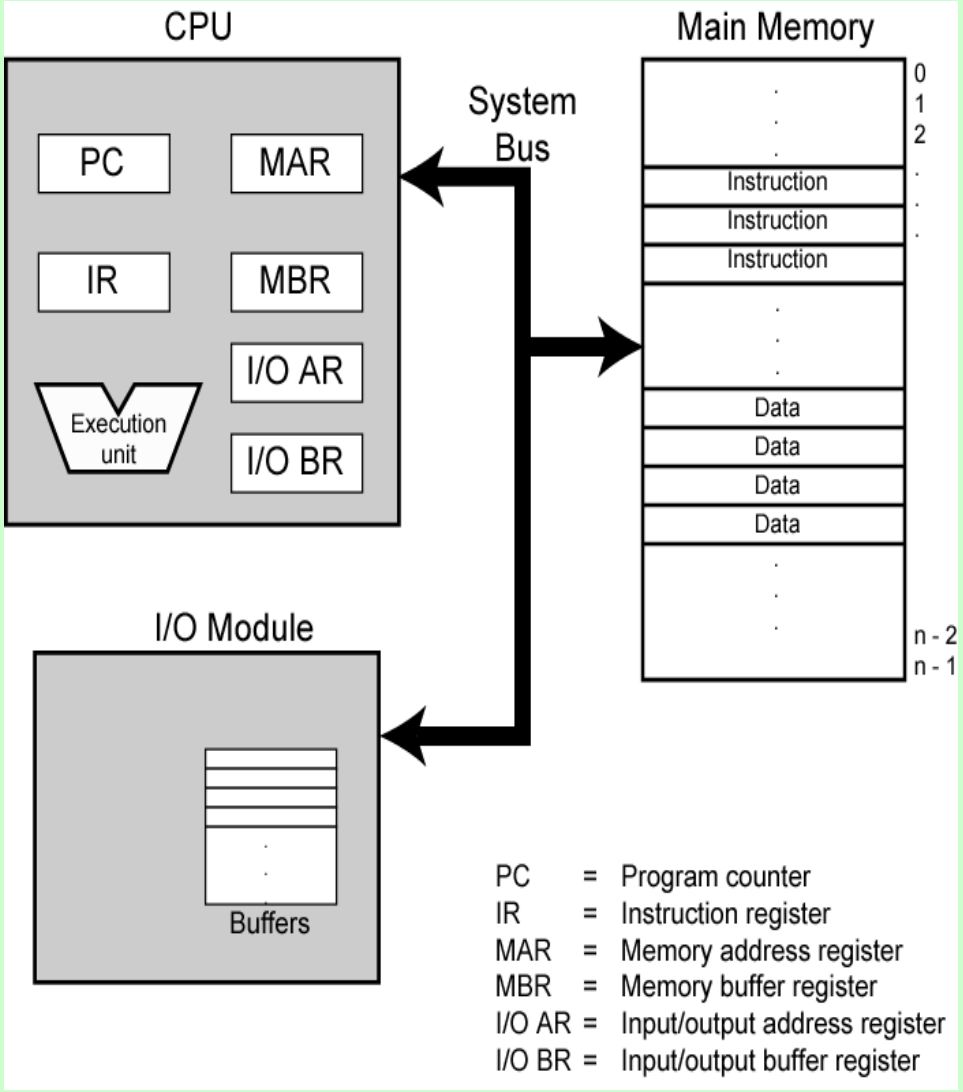
- The Control Unit and the Arithmetic and Logic Unit constitute the Central Processing Unit
- Data and instructions need to get into the system and results out
  - Input/output
- Temporary storage of code and results is needed
  - Main memory

Computer Components: Top Level View

The CPU exchanges data with memory. It uses of 2 internal (CPU) registers: a MAR, which specifies the address in memory for the next read or write, and a MBR, which contains the data to be written into memory or receives the data read from memory.

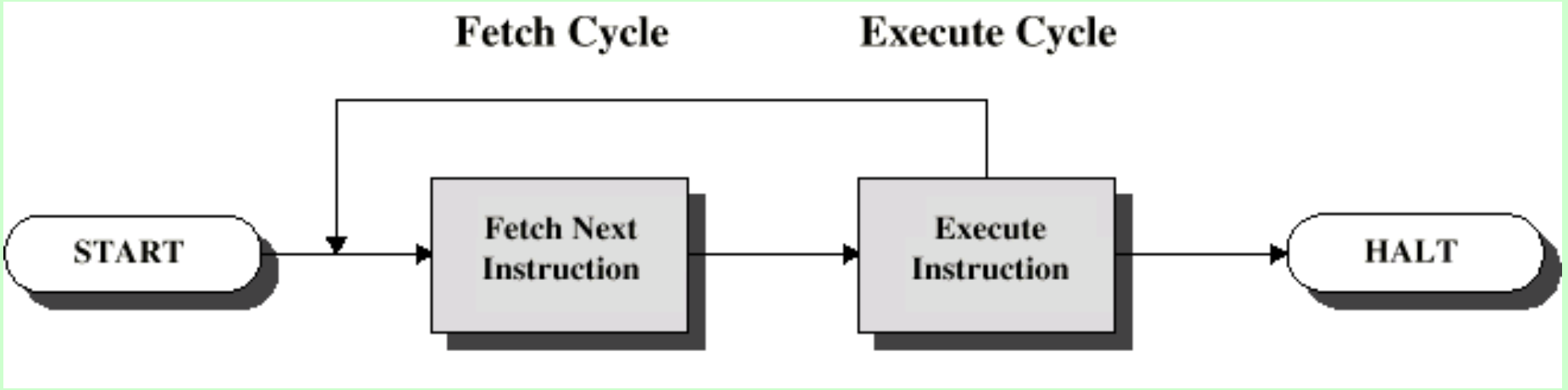
an I/OAR specifies a particular I/O device. An I/OBR register is used for the exchange of data between an I/O module and the CPU.

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.



Instruction Cycle

- Two steps:
  - Fetch
  - Execute



Fetch Cycle

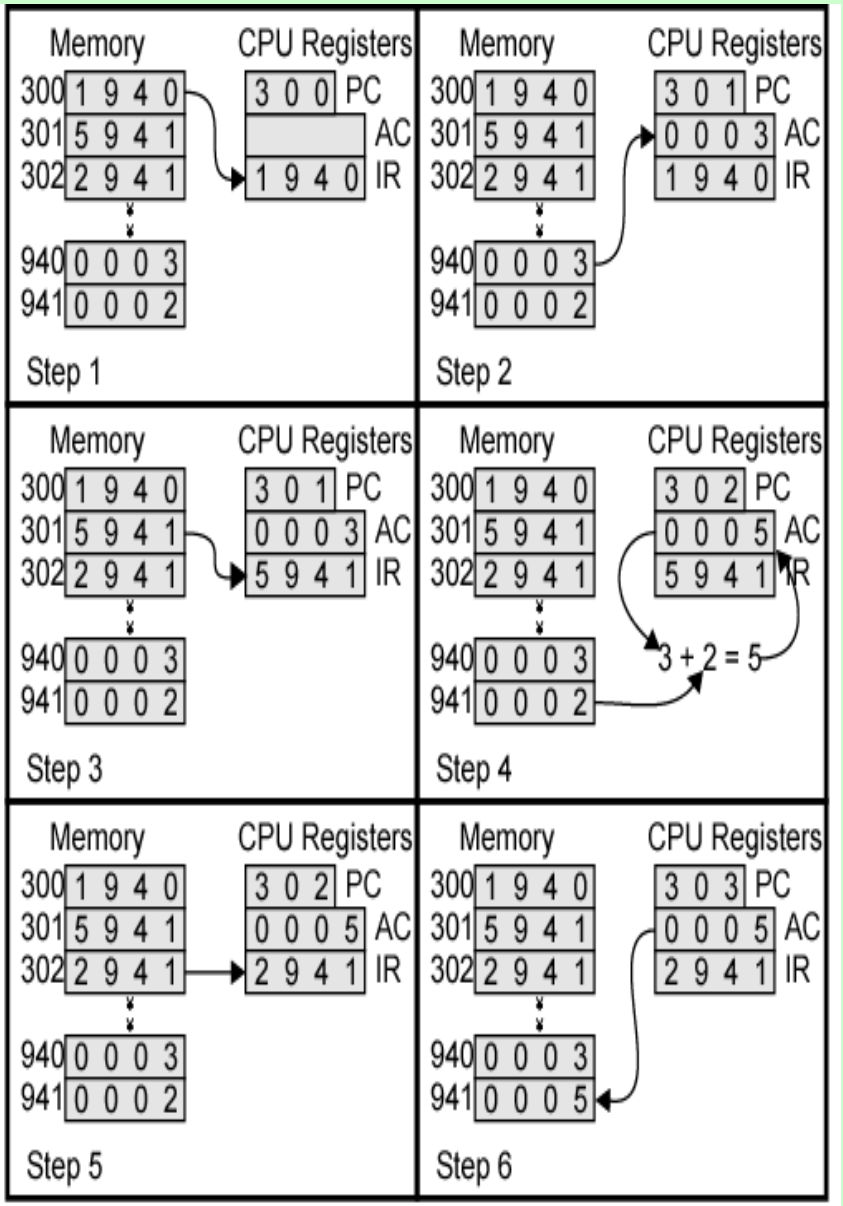
- Program Counter (PC) holds address of next instruction to fetch (to be executed)
- Processor fetches instruction from memory location pointed to by PC
- Increment PC
  - Unless told otherwise
- Instruction loaded into Instruction Register (IR)
- Processor interprets instruction and performs required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

Execute Cycle

- Processor-memory(data movt instructions
  - —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
- Combination of above

Example of Program Execution

1. The PC contains 300, the address of the first instruction. This instruction (1940 in hex) is loaded into the instruction register IR and the PC is incremented. This process involves the use of MAR and MBR.
2. The first 4 bits (first hex digit) in the IR indicate that the AC is to be loaded. The remaining 12 bits (three hex 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.



Instruction Format

- Opcode field

—Part of the code that describes the operation

- Operand field(s)

—Part of the code that describe the data to operate on

- Mode field

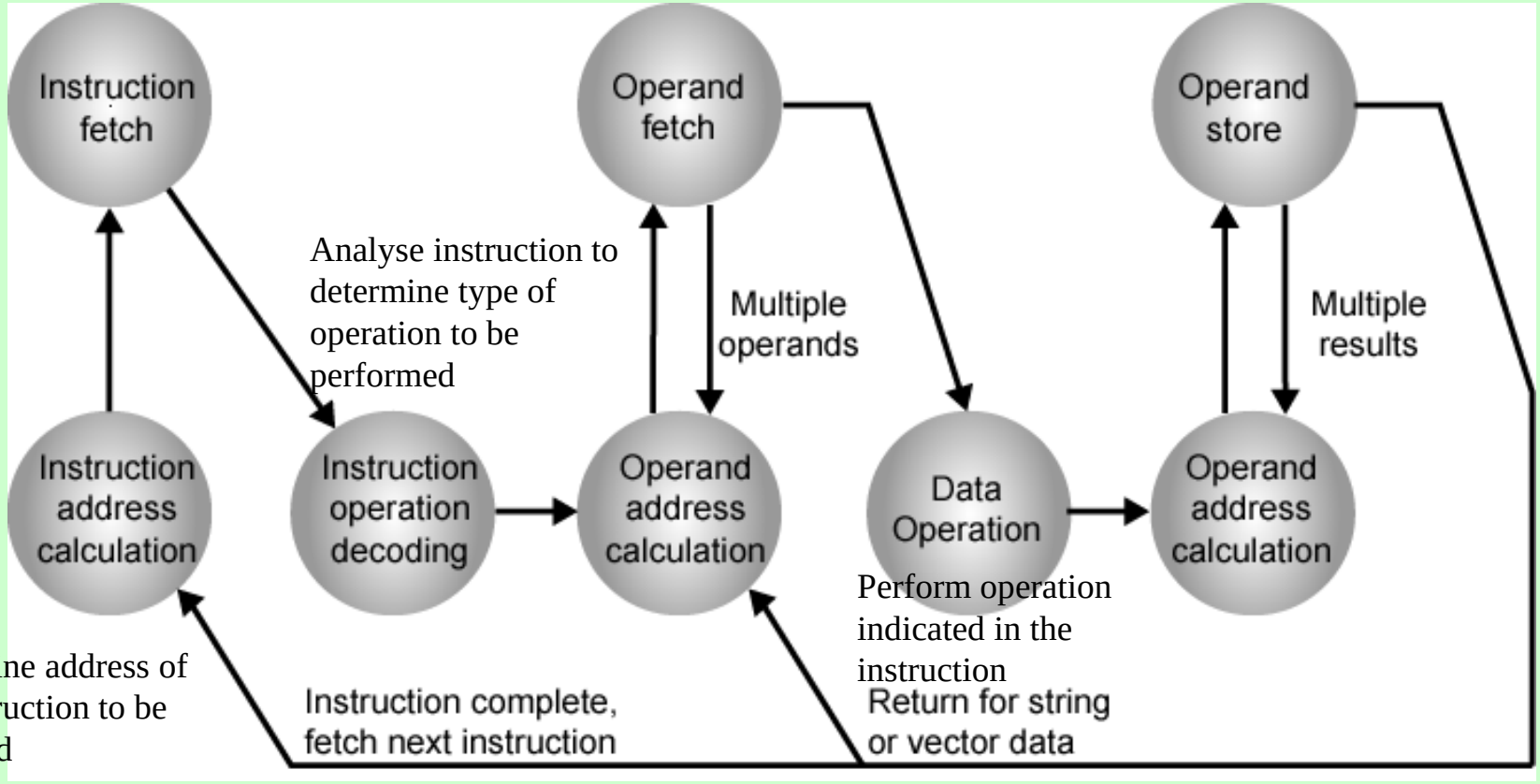
—The way opcode understands operand field

Instruction Cycle State Diagram

Read instruction from its memory loaction into the processor

Fetch operand from memory or read it from I/O

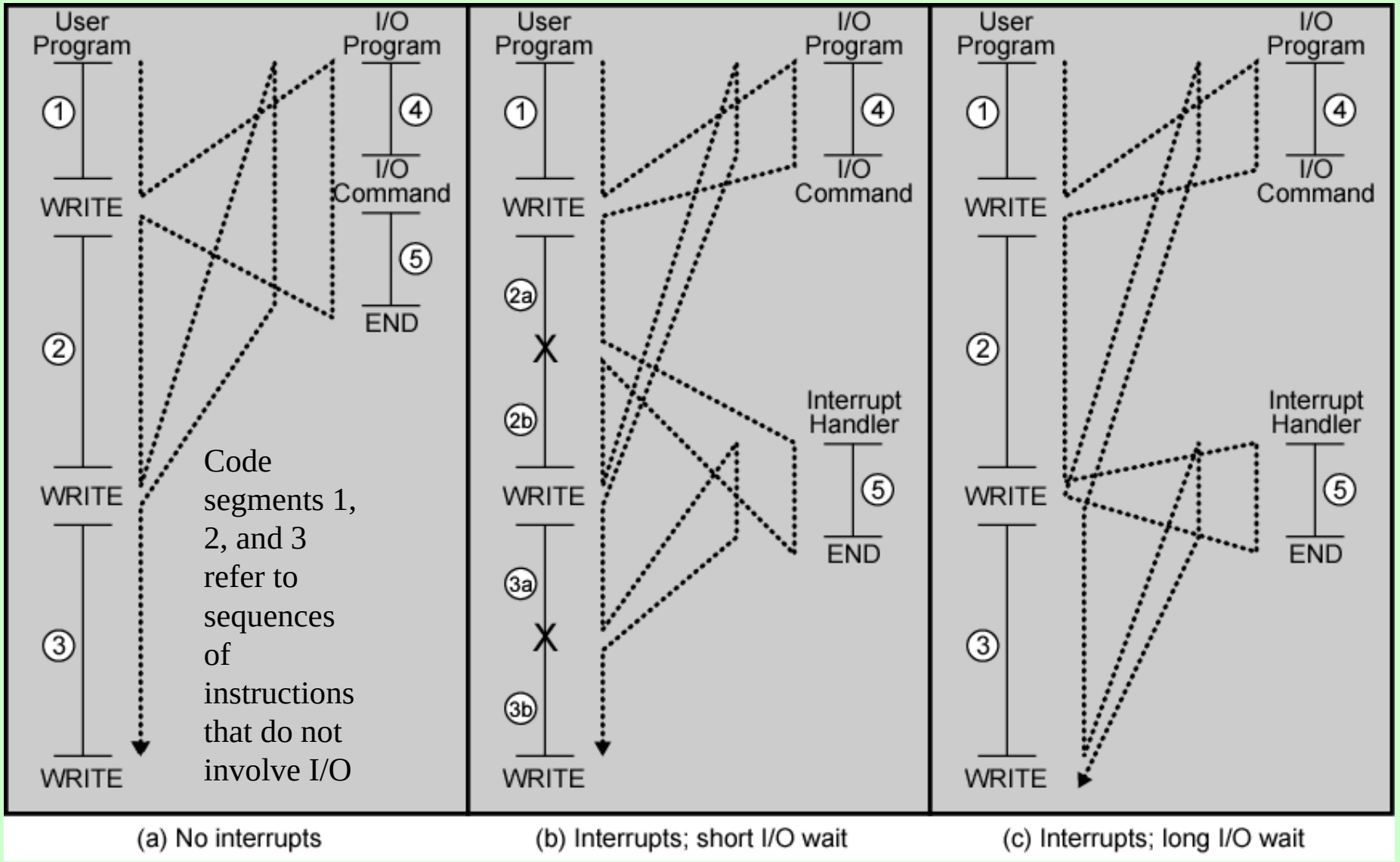
Write result into memory or read it from I/O



Interrupts

- 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



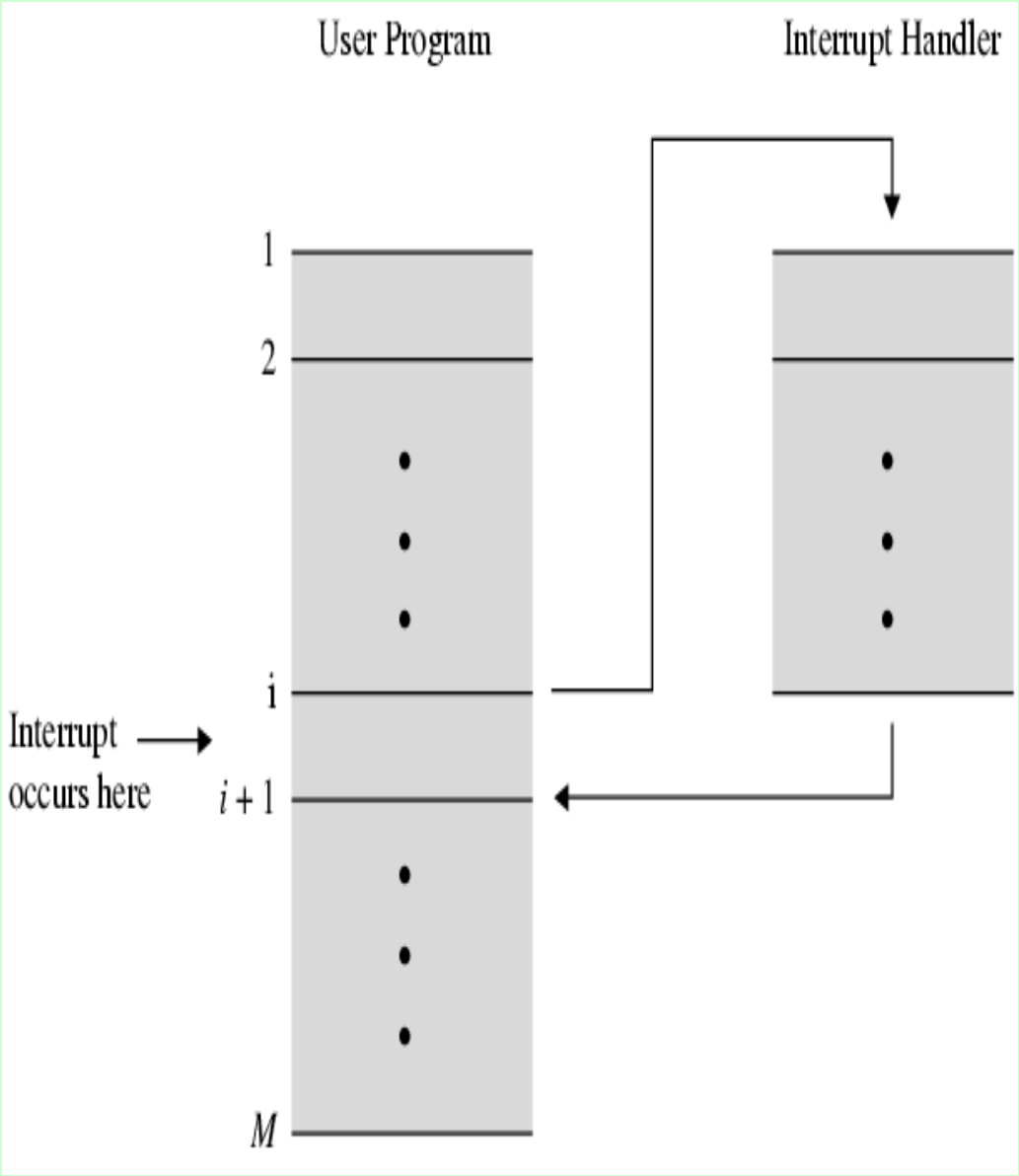
Interrupt Cycle

- 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

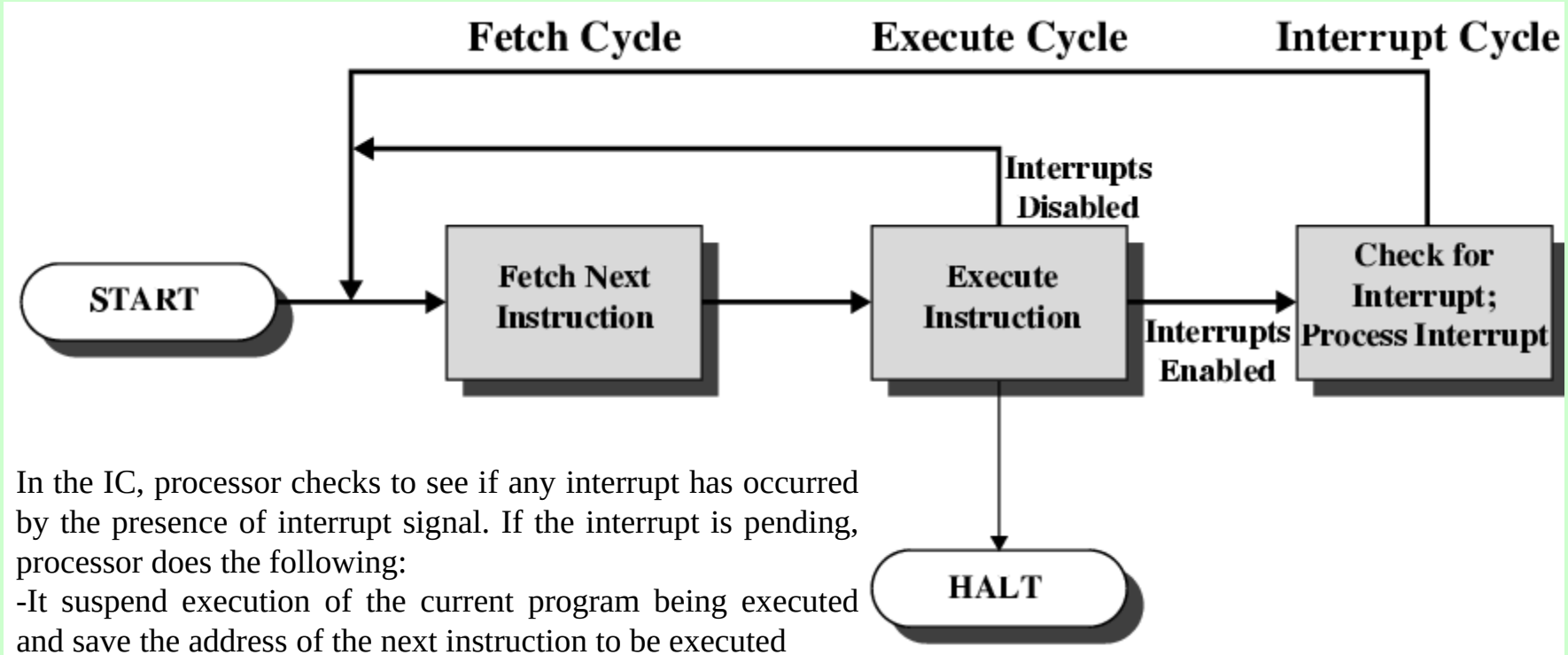
From the point of view of the user program, an interrupt is just that: an interruption of the normal sequence of execution. When the interrupt processing is completed, execution resumes.

Thus, the user program does not have to contain any special code to accommodate interrupts; the processor and the operating system are responsible for suspending the user program and then resuming it at the same point.



Instruction Cycle with Interrupts

To accommodate interrupts, an *interrupt cycle* is added to the instruction cycle



In the IC, processor checks to see if any interrupt has occurred by the presence of interrupt signal. If the interrupt is pending, processor does the following:

-It suspend execution of the current program being executed and save the address of the next instruction to be executed

-It sets the program counter to the start of address of an interrupt handler routine

Interrupt handler is a part of operating system

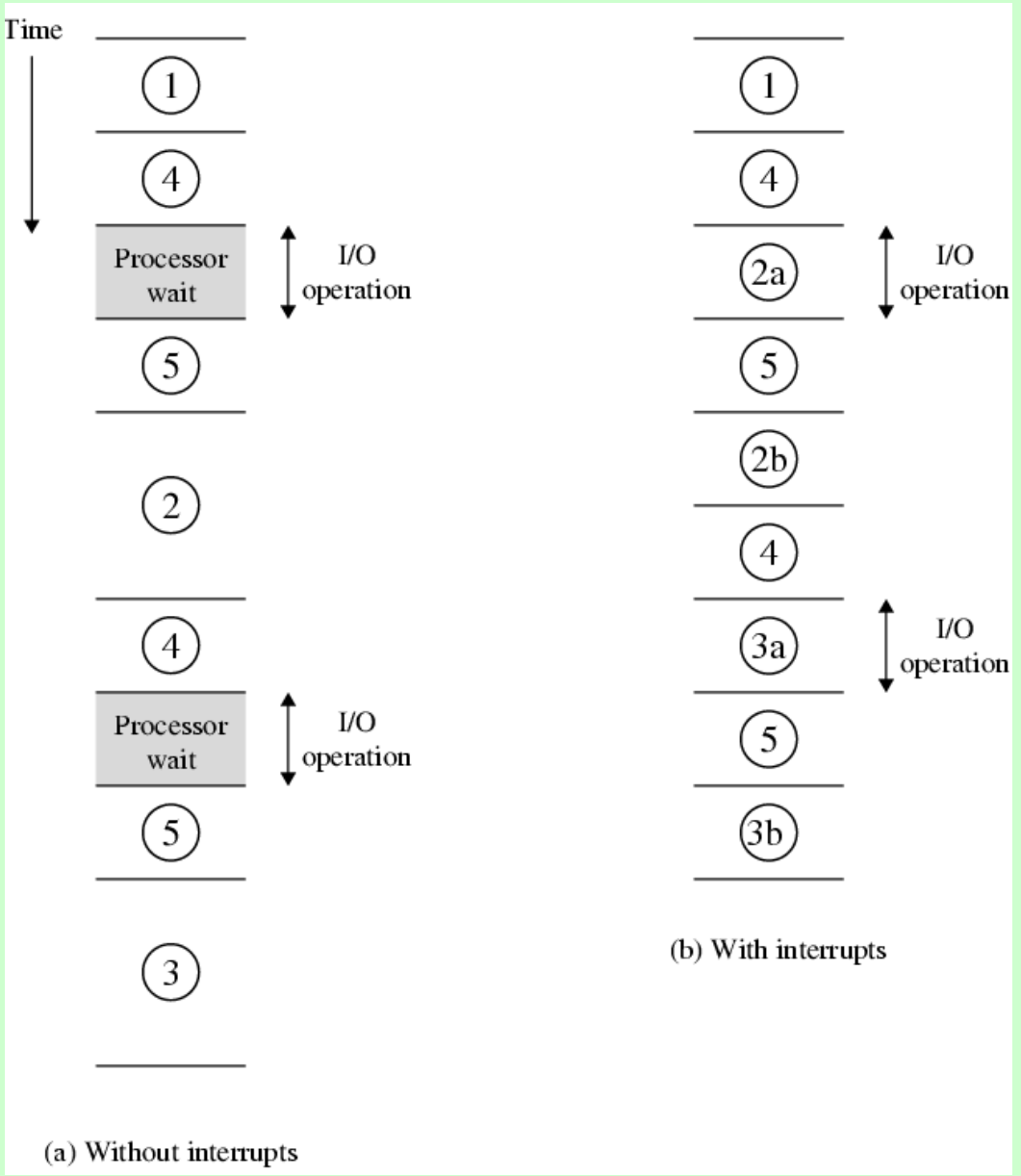
Extra instruction must be executed

Program Timing Short I/O Wait

Assume that the time required for the I/O operation is relatively short: less than the time to complete the execution of instructions between write operations in the user program.

The more typical case, especially for a slow device such as a printer, is that the I/O operation will take much more time than executing a sequence of user instructions.

The result is that the user program is hung up at that point.

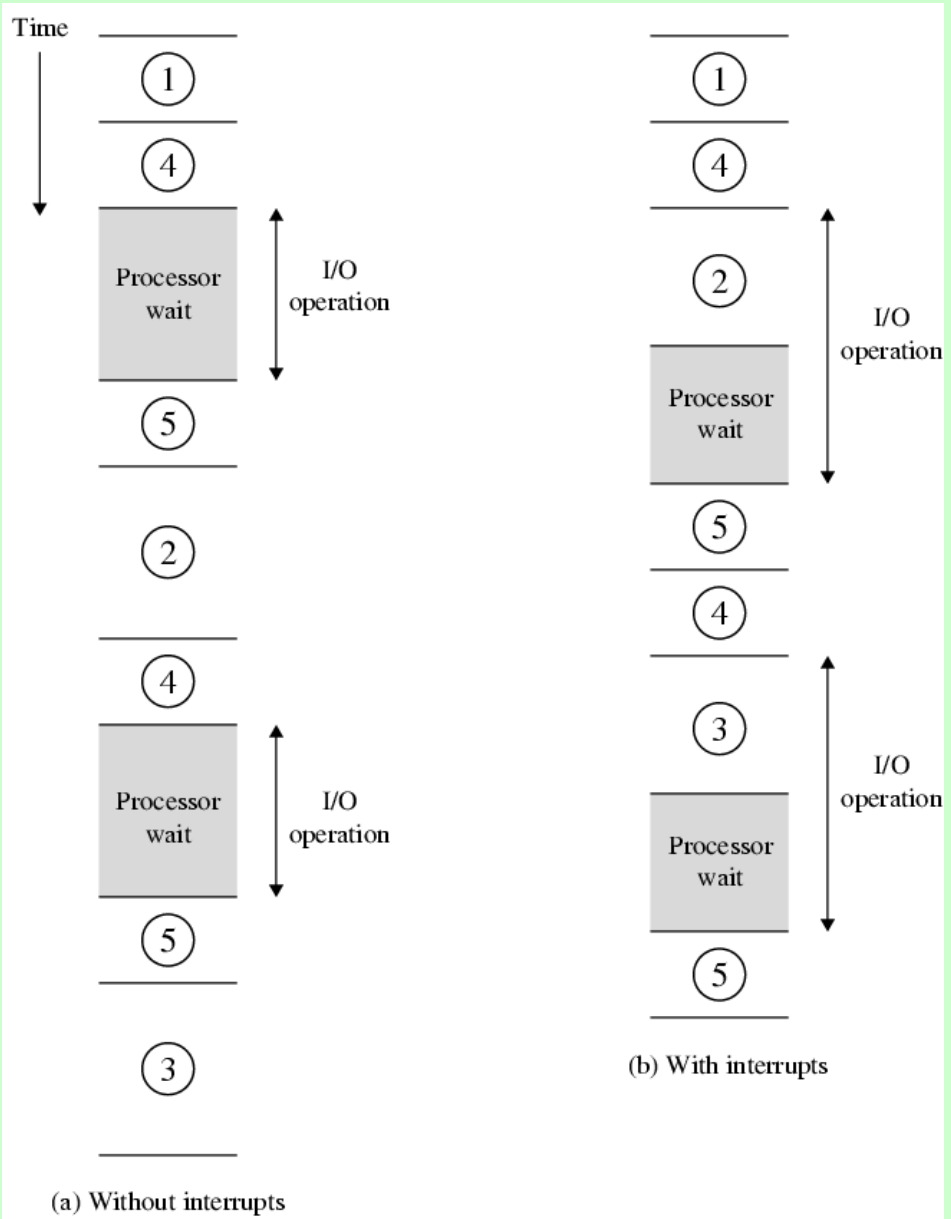


Program Timing Long I/O Wait

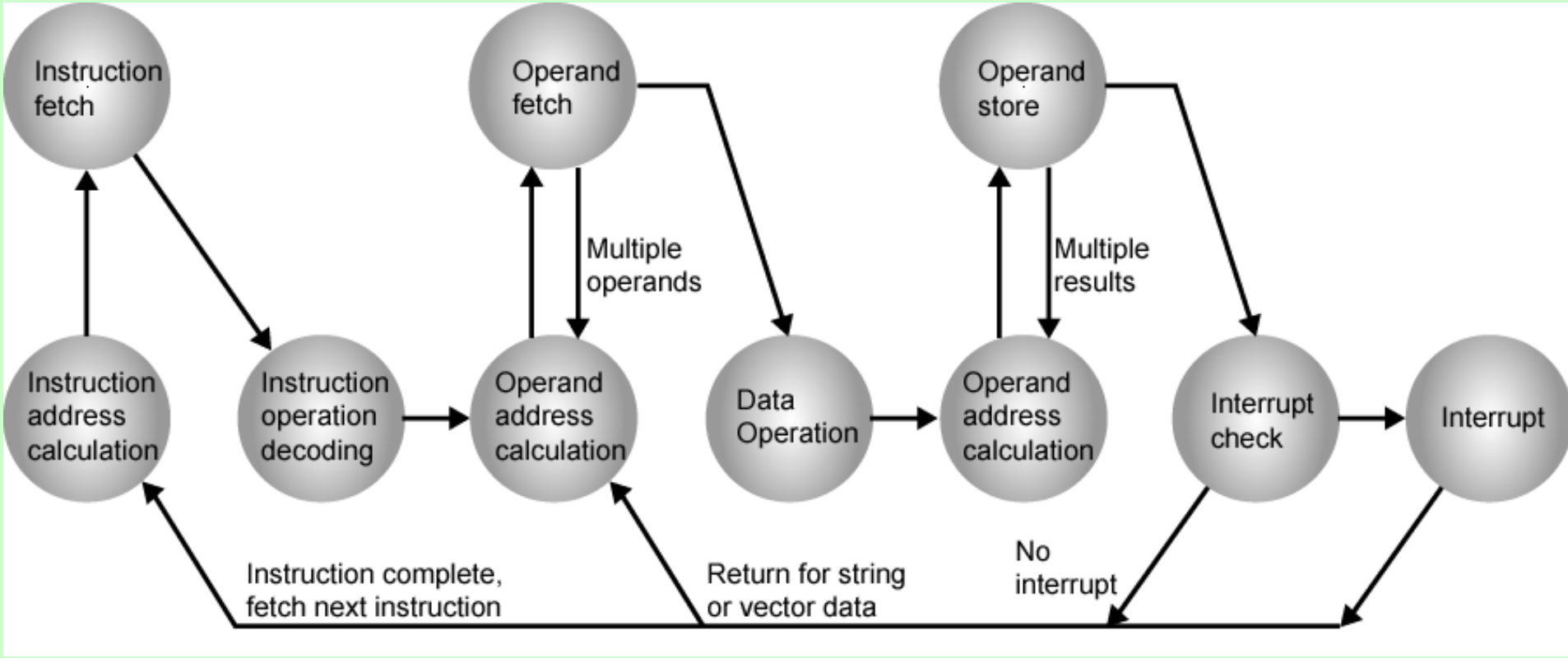
When the preceding I/O operation is completed, this new WRITE call may be processed, and a new I/O operation may be started.

The left panel shows the timing for this situation with and without the use of interrupts.

We can see that there is still a gain in efficiency because part of the time during which the I/O operation is underway overlaps with the execution of user instructions.



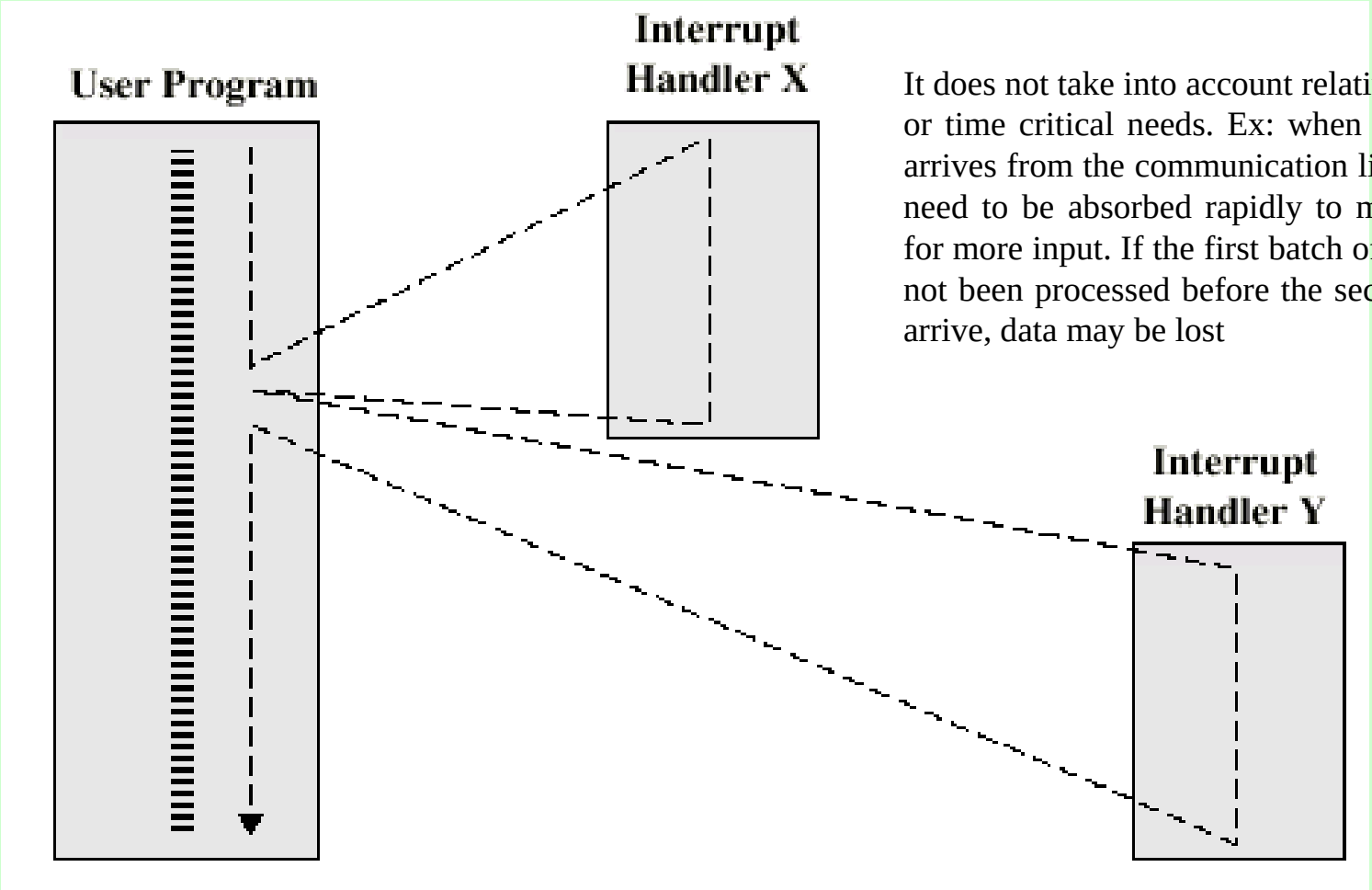
Instruction Cycle (with Interrupts) - State Diagram



Multiple Interrupts

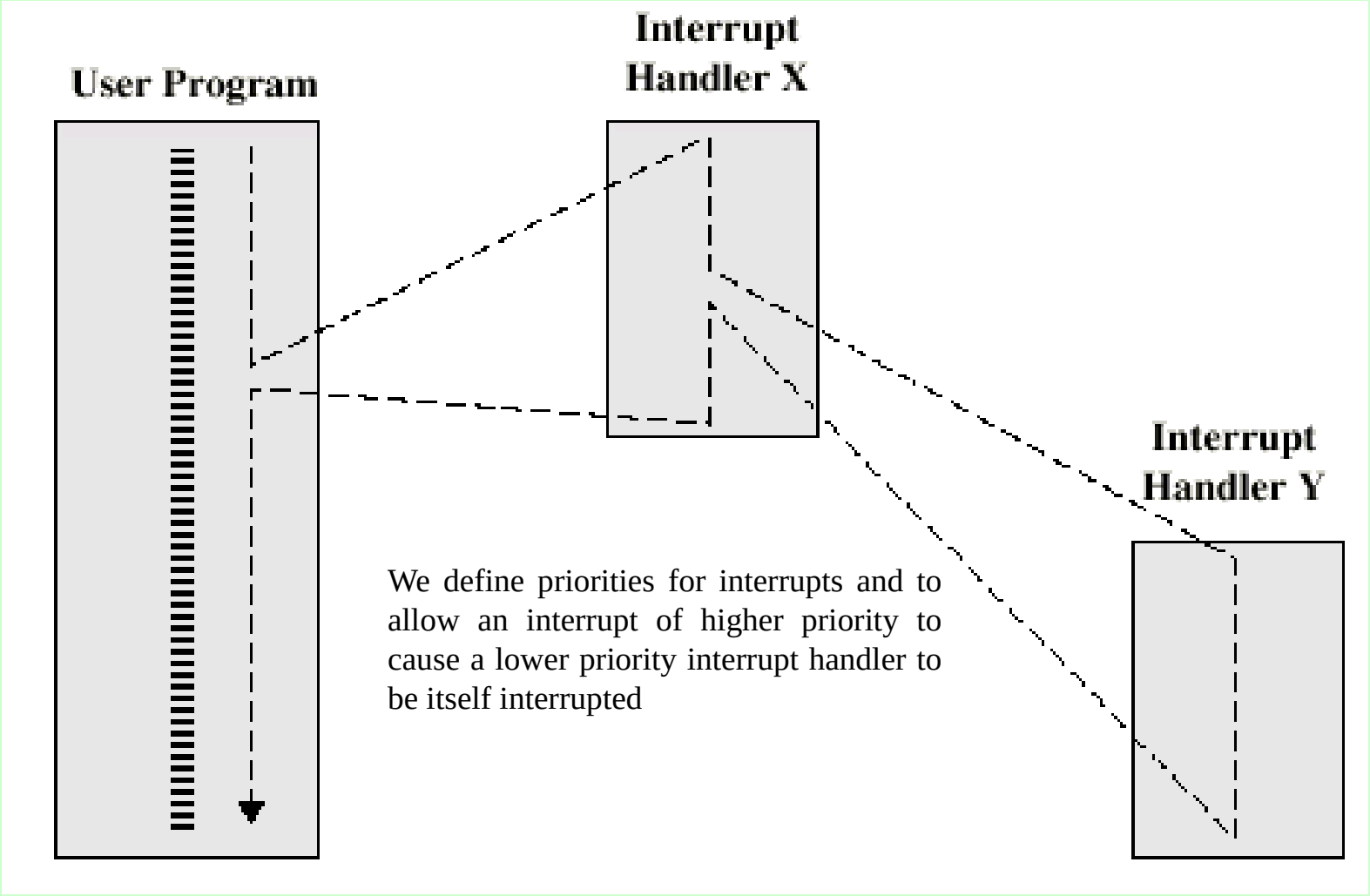
- Disable interrupts
  - Processor will ignore further interrupts whilst processing one interrupt
  - Interrupts remain pending and are checked after first interrupt has been processed
  - Interrupts handled in sequence as they occur
- Define priorities
  - Low priority interrupts can be interrupted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

Multiple Interrupts - Sequential

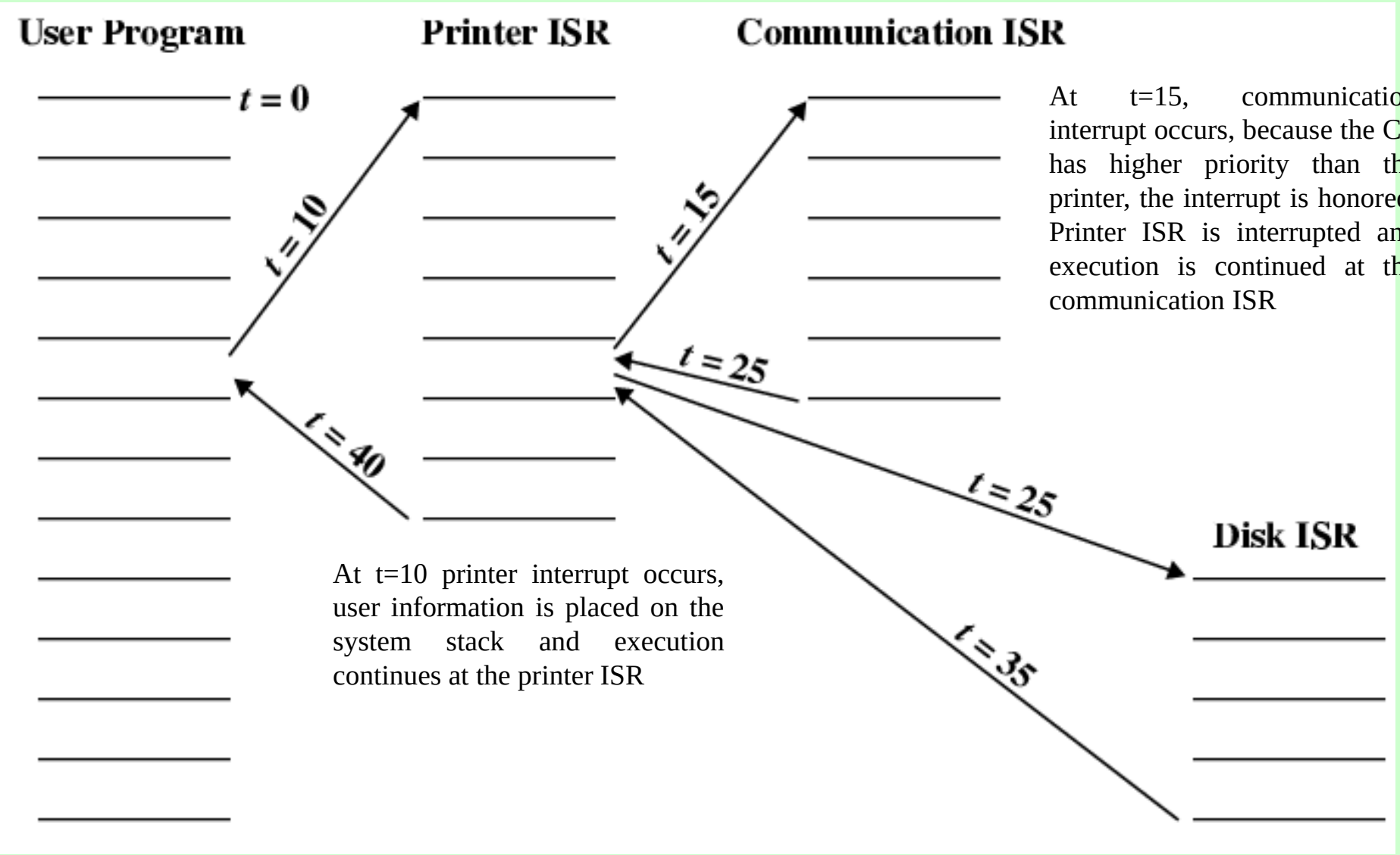


It does not take into account relative priority or time critical needs. Ex: when input data arrives from the communication line, it may need to be absorbed rapidly to make room for more input. If the first batch of input has not been processed before the second batch arrive, data may be lost

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
  - Input/Output
  - CPU