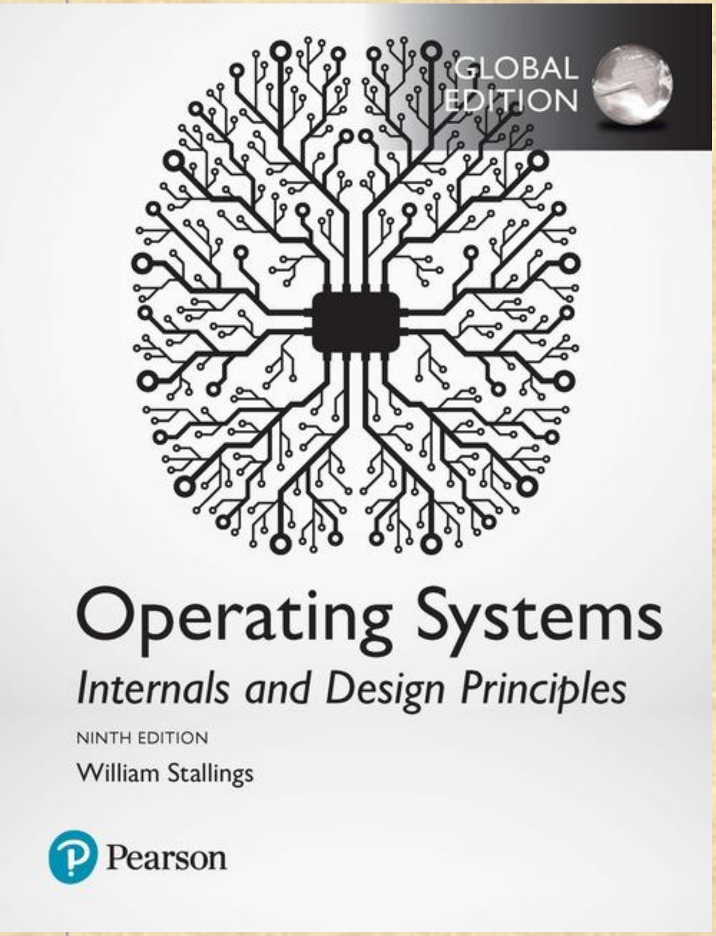


Chapter 1 – Part 1 Computer System Hardware

Ninth Edition, Global Edition

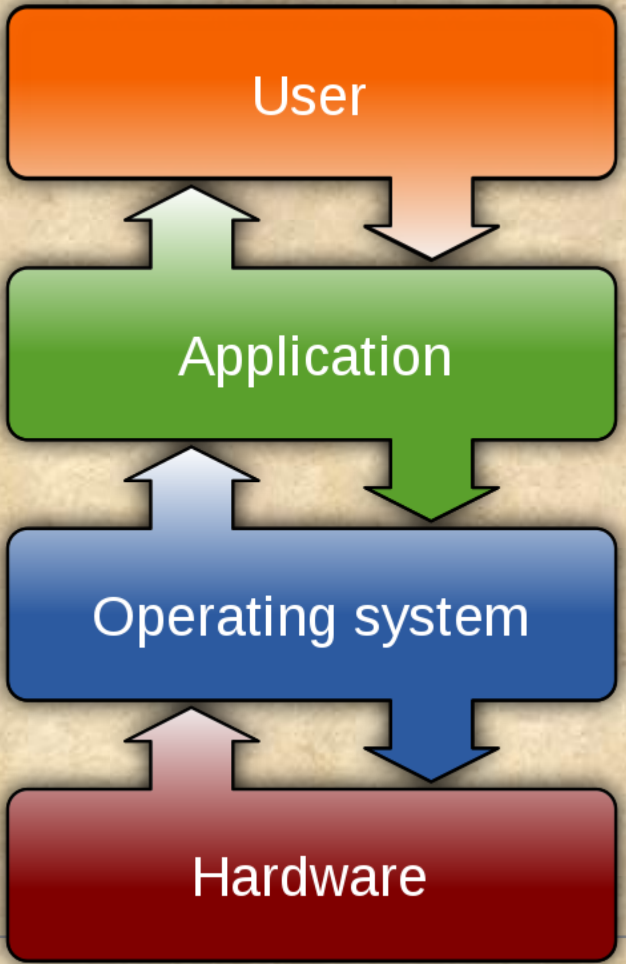
By William Stallings

Operating System

- Manages the **hardware components** of the computer:

Processor, registers, main memory, disk memory and I/O devices

- Provides services to user applications e.g. scheduling



Basic Hardware Components

- Processor
- I/O Modules
- Main Memory
- System Bus

Processor

Central Processing Unit (CPU)

Controls the operation of the computer

Loads and executes instructions

Main Memory

- Stores data and programs (sequence of instructions)
- Volatile
  - Contents of the memory is lost when the computer is shut down
- Referred to as **real memory** or **primary memory**

I/O Modules

Move data between the computer and its external environment

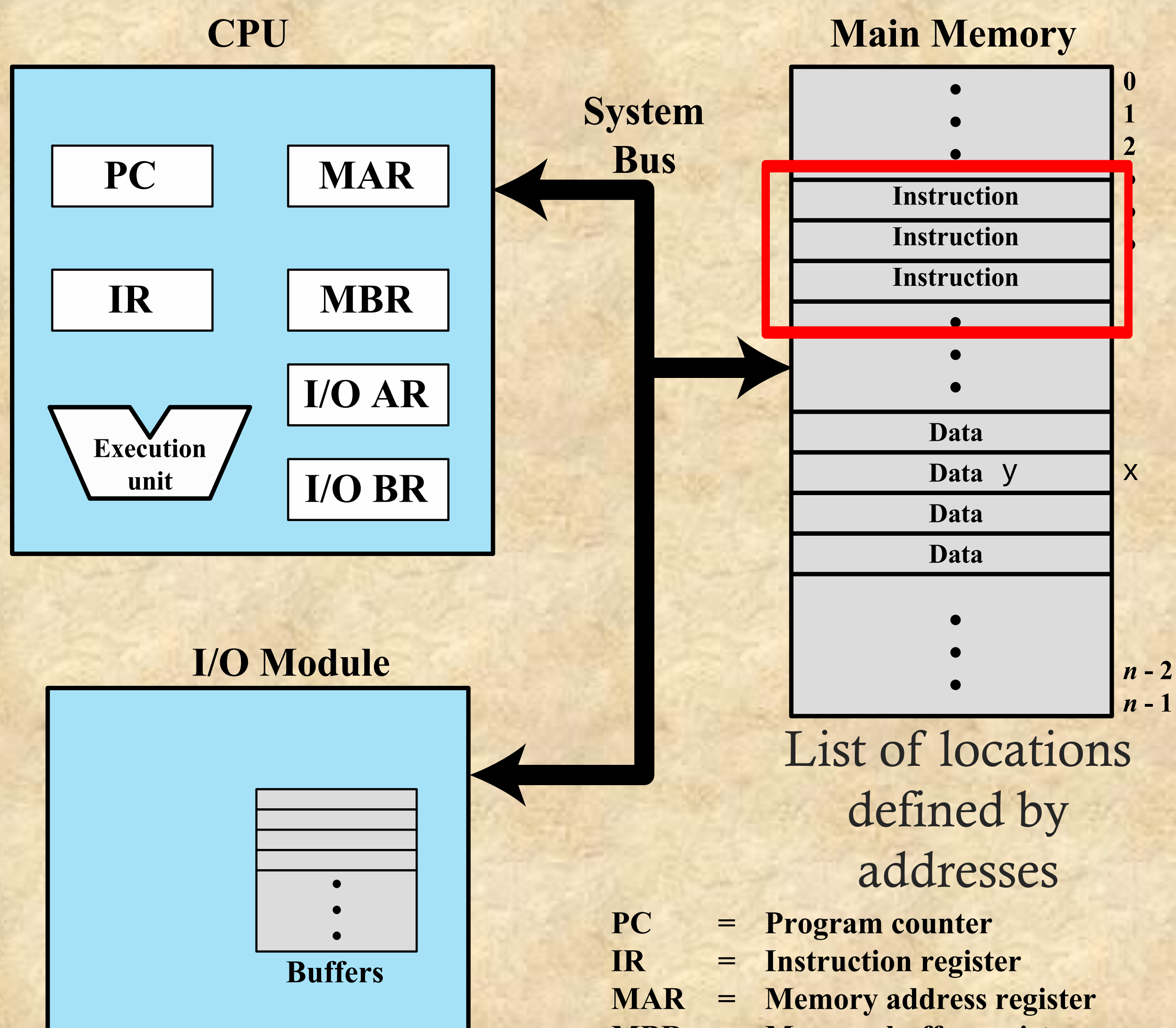
- Secondary memory devices (e.g. hard disks)
- Communications equipment (e.g. Keyboard, Printer)
- Terminals

System Bus

- Communication line among processors, main memory, and I/O modules

Interplay Between Basic Hardware Components

Registers for exchanging data with memory and I/O modules



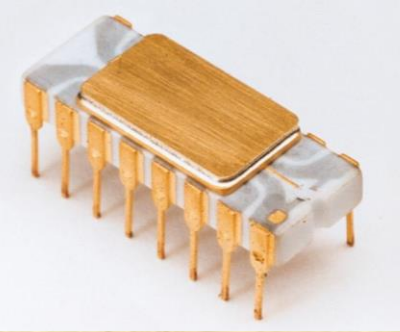
E.g. keyboard module buffers keyboard input

List of locations defined by addresses

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

Evolution of Hardware: Microprocessor

- First processor on a **single** chip
- Invention that brought up desktop and handheld computers
- Fastest general-purpose processor
- Today also multiprocessor technology:
  - **Multiple** processors (cores) on a single chip



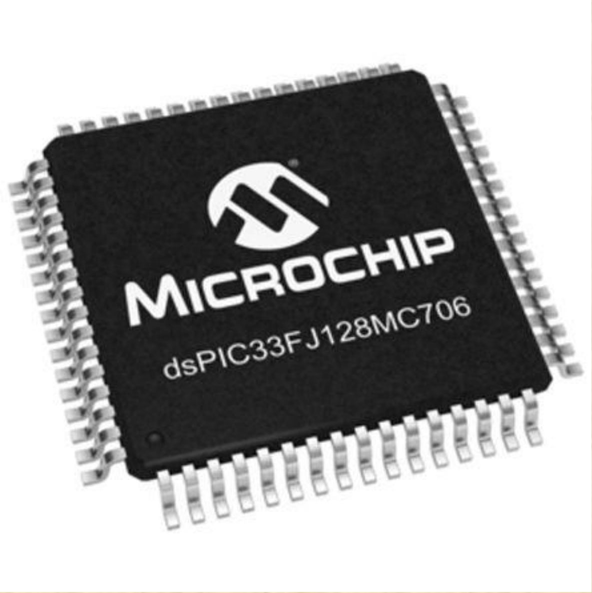
Evolution of Hardware: Graphical Processing Units (GPU)

- Initially introduced as a special-purpose processor for graphical computations
- Efficient computation on arrays of data

- Today also used for general numerical processing
  - Physics simulations
  - Computations on large spreadsheets

Evolution of Hardware: Digital Signal Processors (DSP)

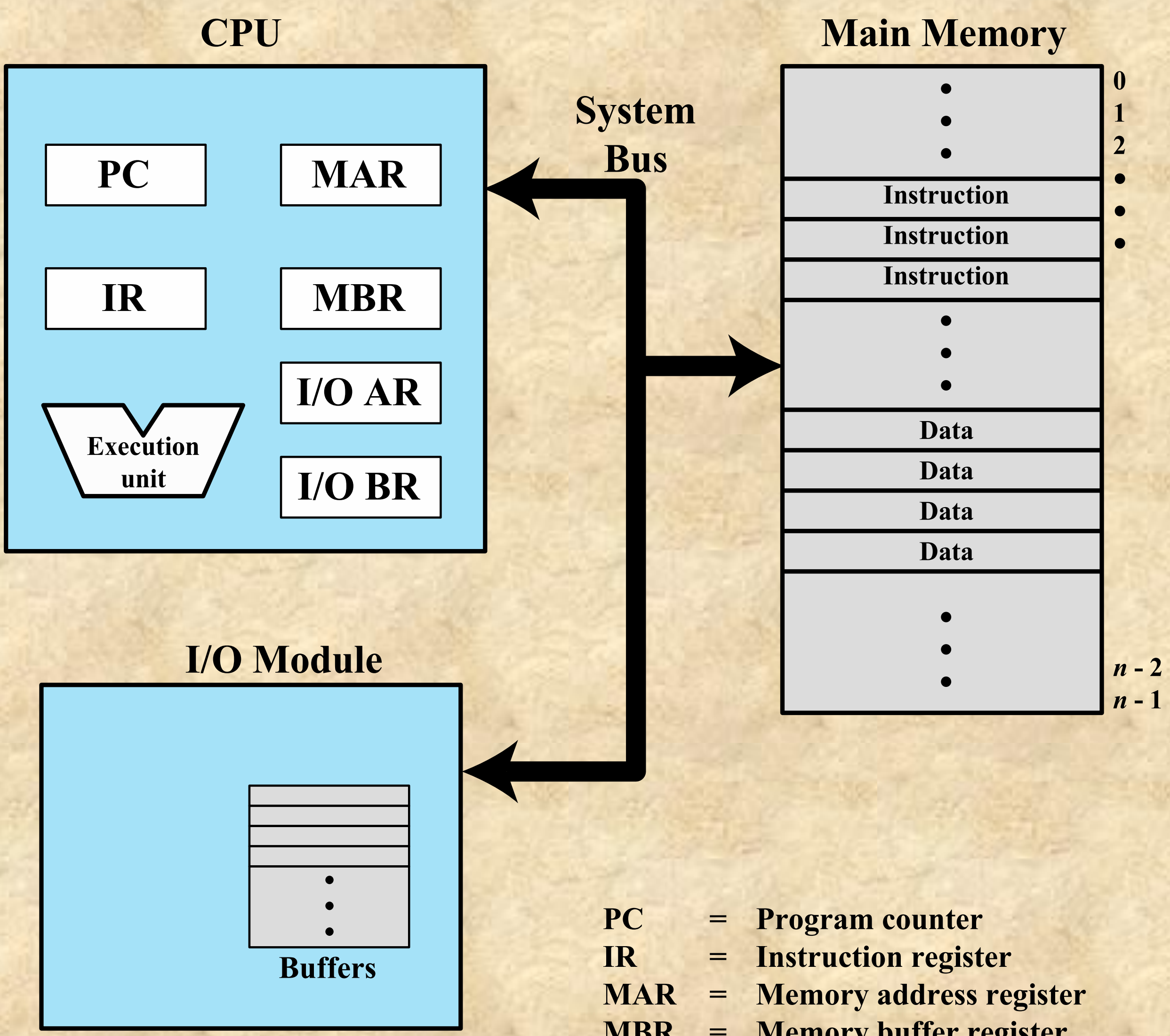
- Special-purpose processors for streaming signals such as audio or video
- Encoding/decoding speech and video (codecs)
- Provide support for encryption and security



Evolution of Hardware: System on a Chip (SoC)

- CPUs, DSPs, GPUs, I/O modules and main memory are on **single** chip
- Fast communication between components

Basic Hardware Components – Instruction Execution



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

Instruction Execution

- A program consists of a set of instructions stored in memory

Two step execution

Processor reads (fetches) instructions from memory

Processor executes each instruction

2-Stage Instruction Cycle

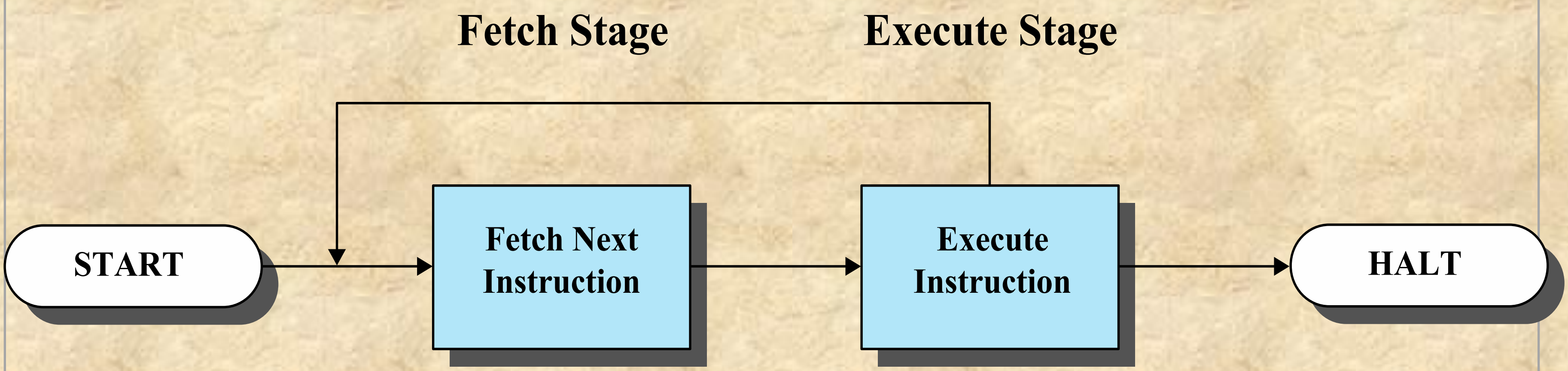


Figure 1.2 Basic Instruction Cycle

Instruction Fetch

- Program counter register: 0100
- The processor fetches an instruction, represented as a bit word, from memory:
0001 1101 1100 0000 0100
- The program counter (PC) holds the address of the next instruction to be fetched
  - PC incremented after each fetch e.g. 5, 6, 7, ...
  - Typically sequential execution

Instruction Register (IR)

Fetched instruction is loaded into Instruction Register (IR)

- Processor interprets the instruction and performs required action:
  - Processor-memory transfer
  - Processor-I/O transfer
  - Operation on data
  - Control

(a) **Instruction format**

(b) **Data format**

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

(d) **Partial list of opcodes**

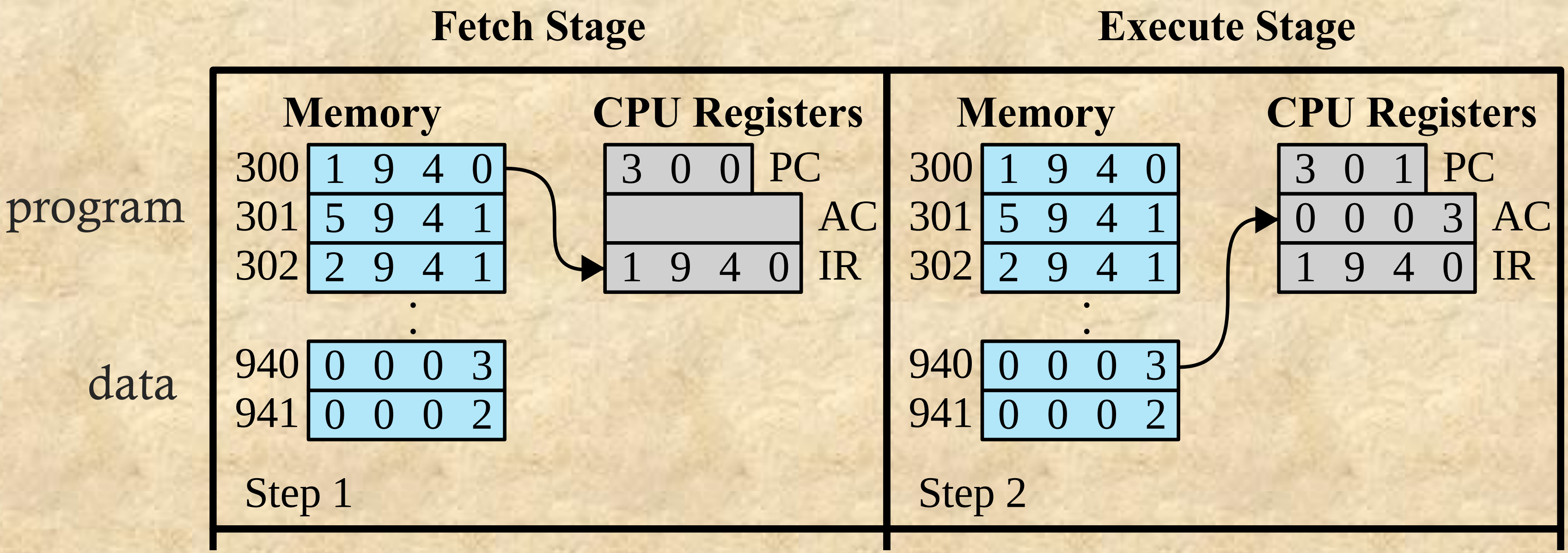
AC is accumulator register of CPU

Example for 16-bit Instruction

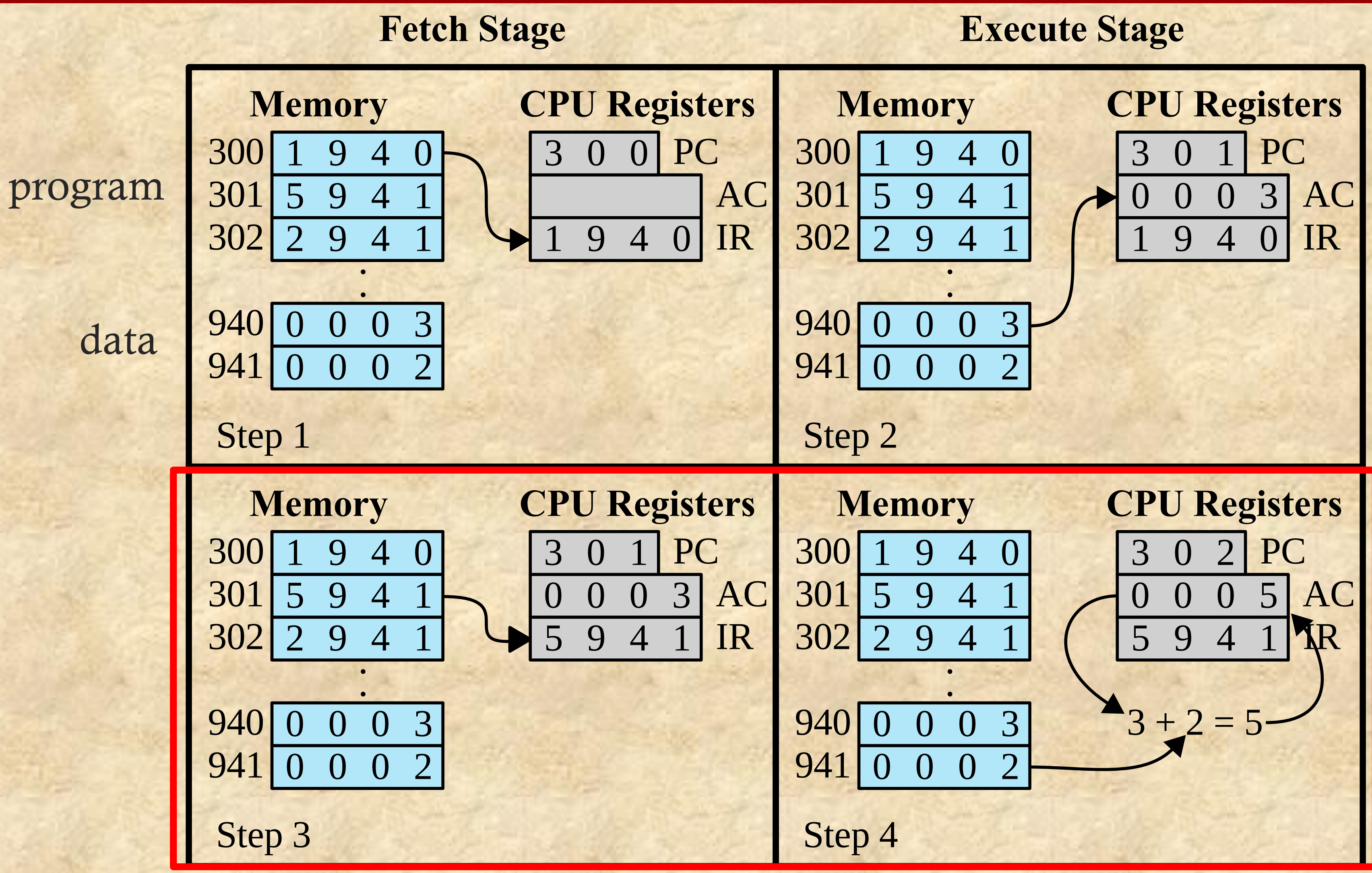
Binary: 0001 1001 0100 0000

Hexadecimal: 1 9 4 0

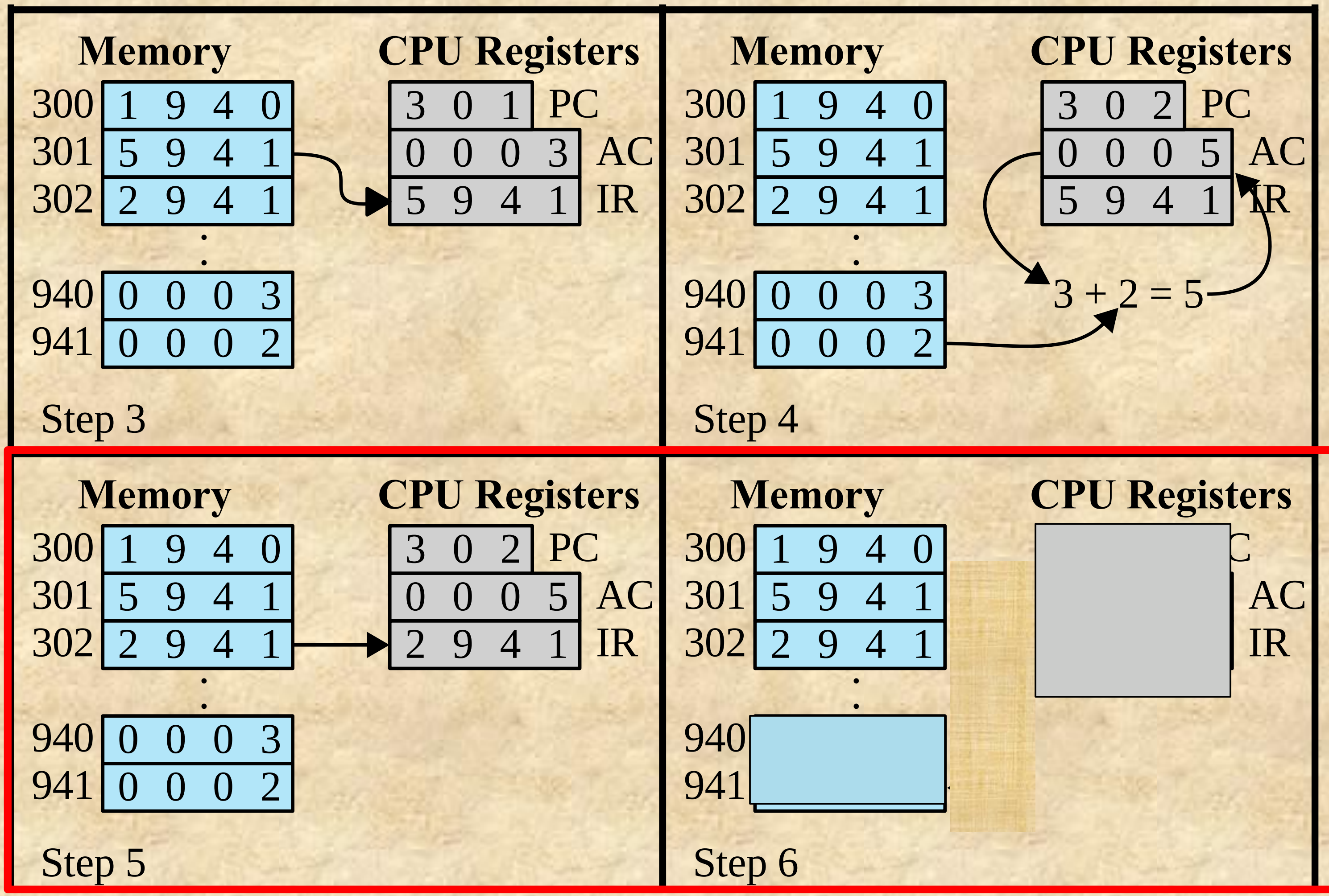
Meaning: LOAD from memory address 940



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

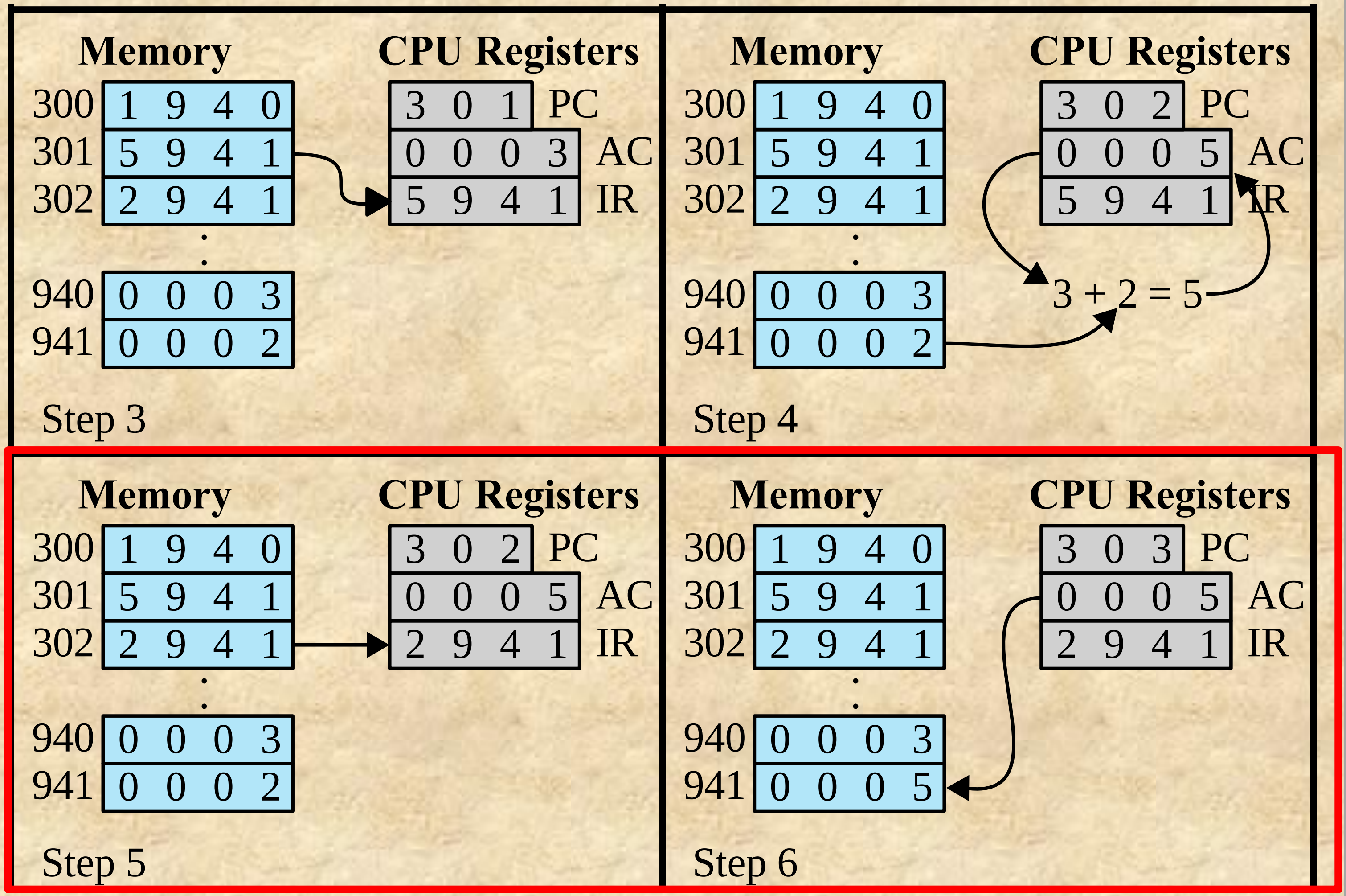


0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory



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

Content of memory and registers after Step 6?



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

Interrupts

- Mechanism by which modules (I/O) can interrupt the normal program execution of the processor
- Improves utilization of the processor
  - I/O devices are much slower than the processor
  - Without interrupts processor must pause while a device performs I/O commands
  - With interrupts processor can execute other instructions while a device performs I/O commands
  - I/O module will send an interrupt when the command is finished and the processor is needed by the module

Classes of Interrupts

I/O	Generated by an I/O module, to signal that it needs the processor
Program	Exceptions for arithmetic overflow, division by zero, etc.
Timer	Generated by a timer. Allows the OS to perform certain functions on a regular basis, e.g. automatic backup
Hardware	Generated by a failure, such as power failure

Example: Program Execution without Interrupts

User program:

Code segments 1 and 2 that do not involve I/O

WRITE calls (pause user program and execute I/O program on processor)

I/O program:

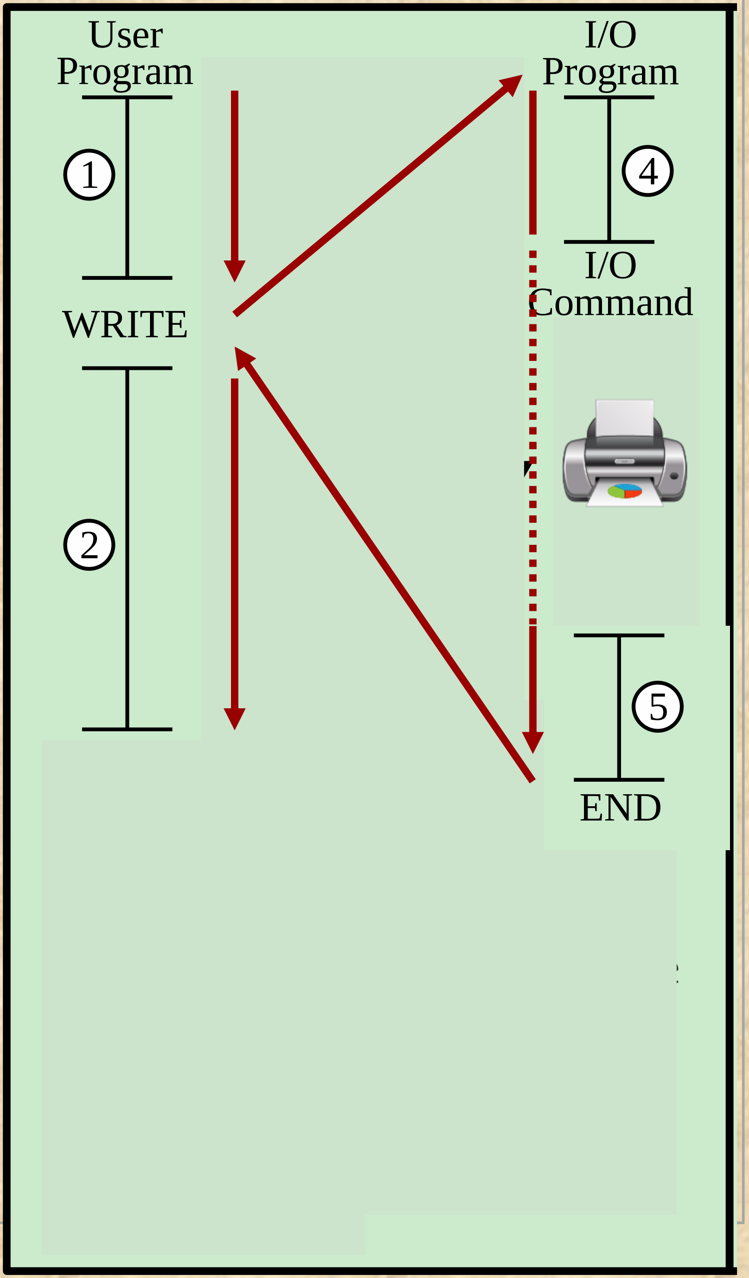
Segment 4 copies data to write buffer

I/O command prints on printer (processor not involved)

Segment 5 notifies about success or failure of WRITE

Without interrupts, user program must wait until I/O program ends

Processor idles during I/O command



Execution with interrupts:

After Segment 4, processor continues execution of user program (Segment 2)

Now user program and I/O command run in parallel

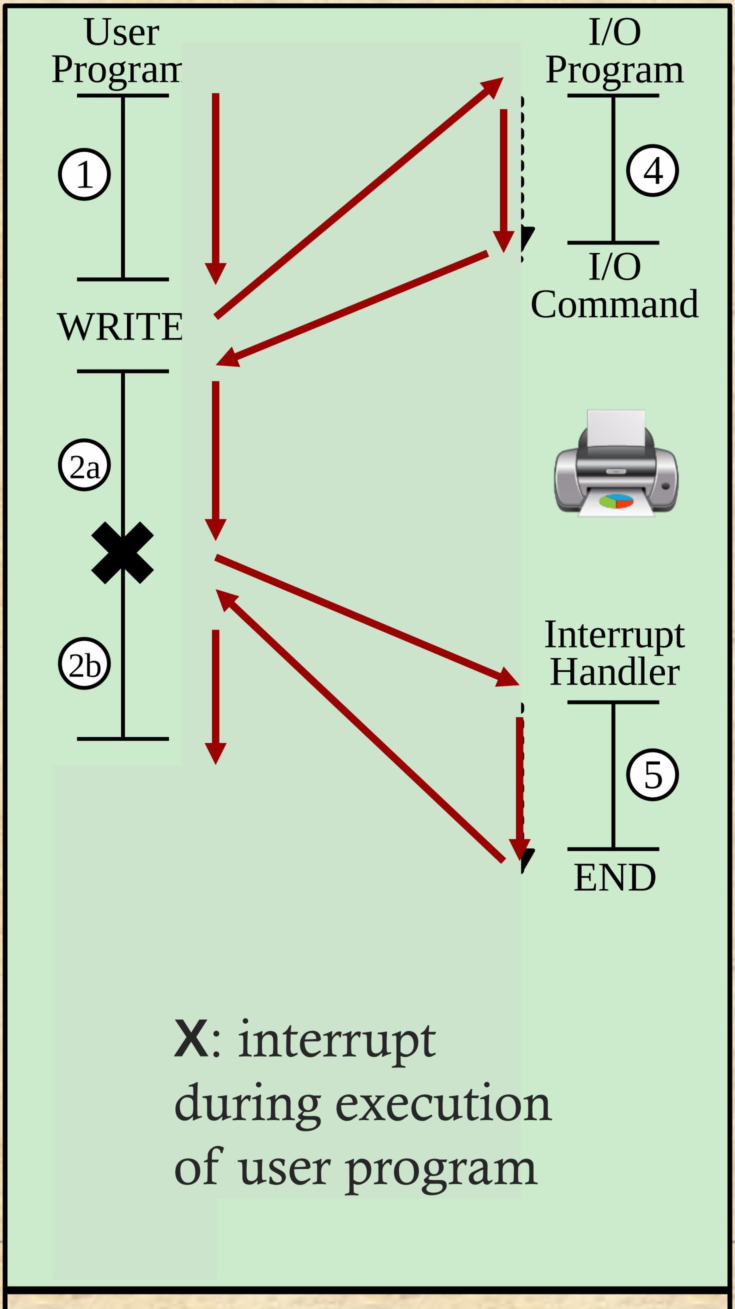
When I/O command finishes, I/O module sends interrupt request to processor

Processor suspends user program, services Segment 5 of I/O program (interrupt handler routine), and resumes user program

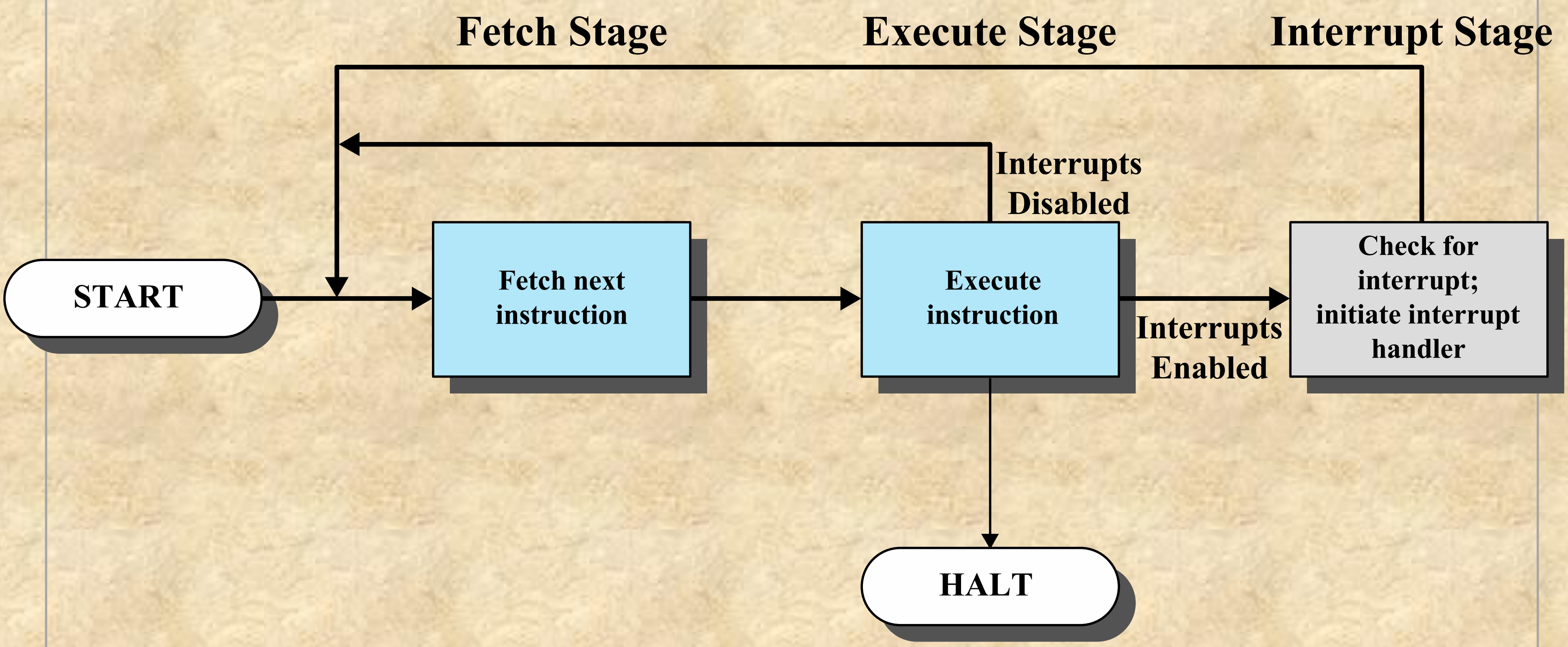
Interrupt can occur at any time

State of the user program needs to be saved for an interrupt

Better utilization of the processor with interrupts



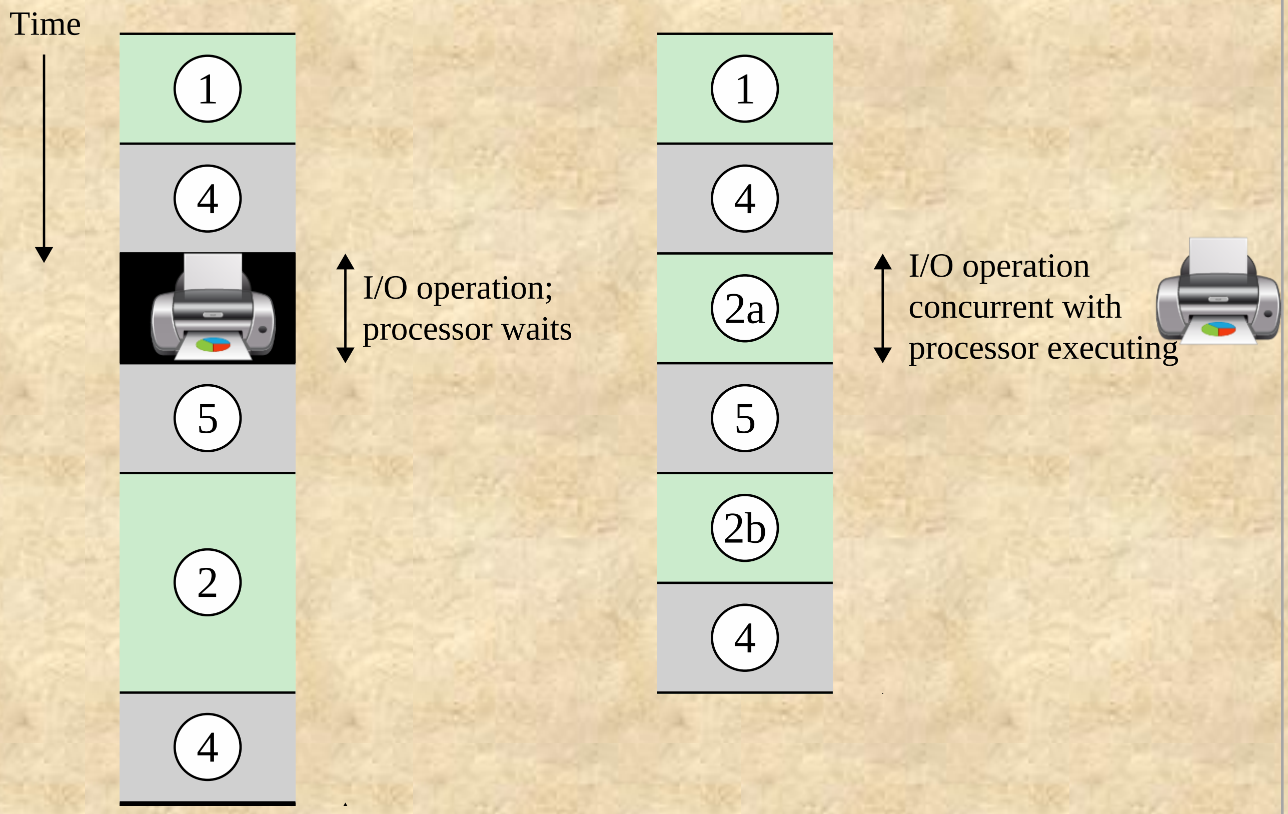
3 Stage Instruction Cycle with Interrupts



Timing diagrams for program execution without and with interrupts

Less idle time with interrupts

Interrupts enable better utilization of the processor, and thus, better performance



Memory Content

Interrupt handling:

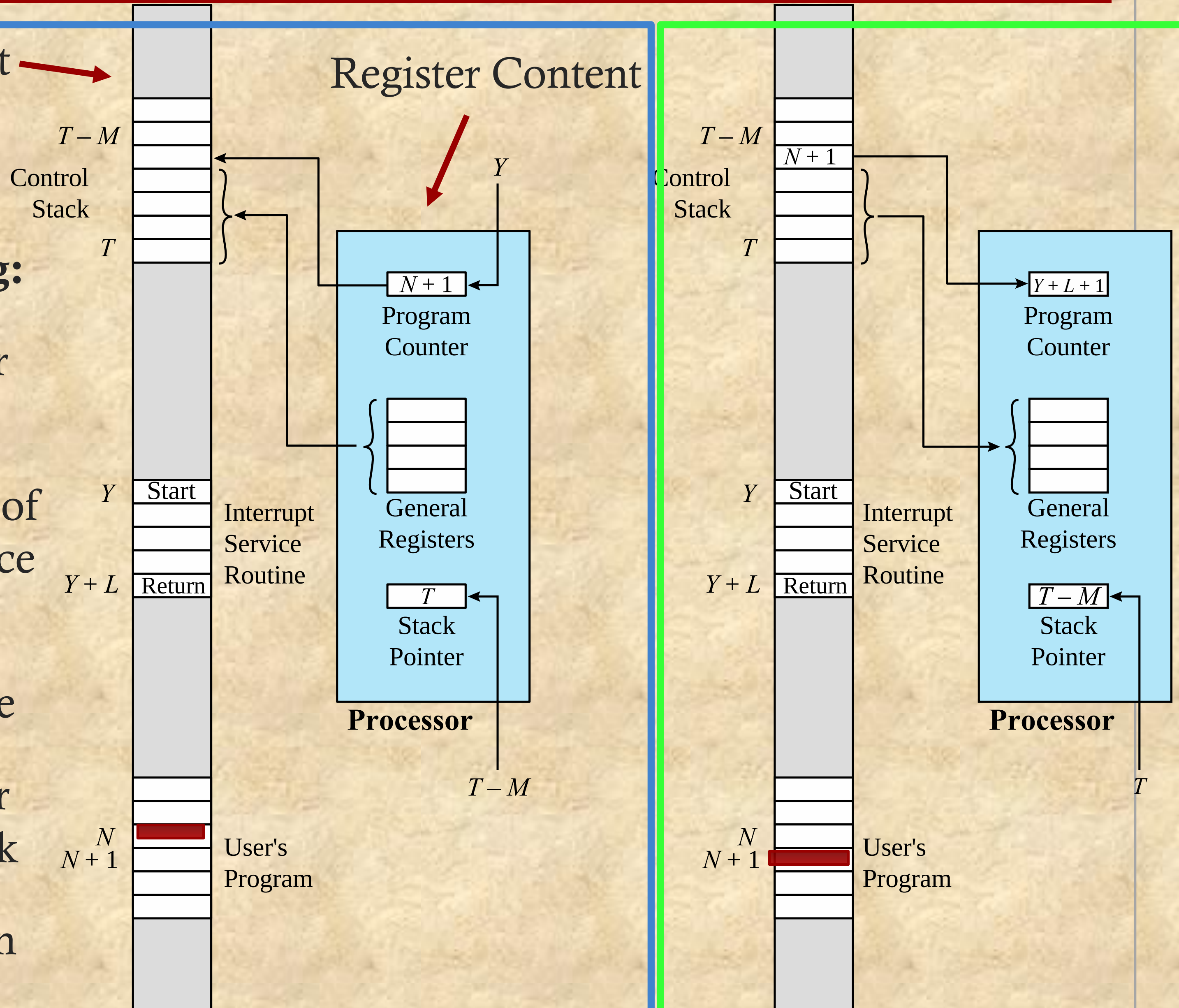
Store status of user program on stack

Set PC to the start of the Interrupt Service Routine

Execute the routine

Load status of user program from stack

Continue execution of user program



Multiple Interrupts

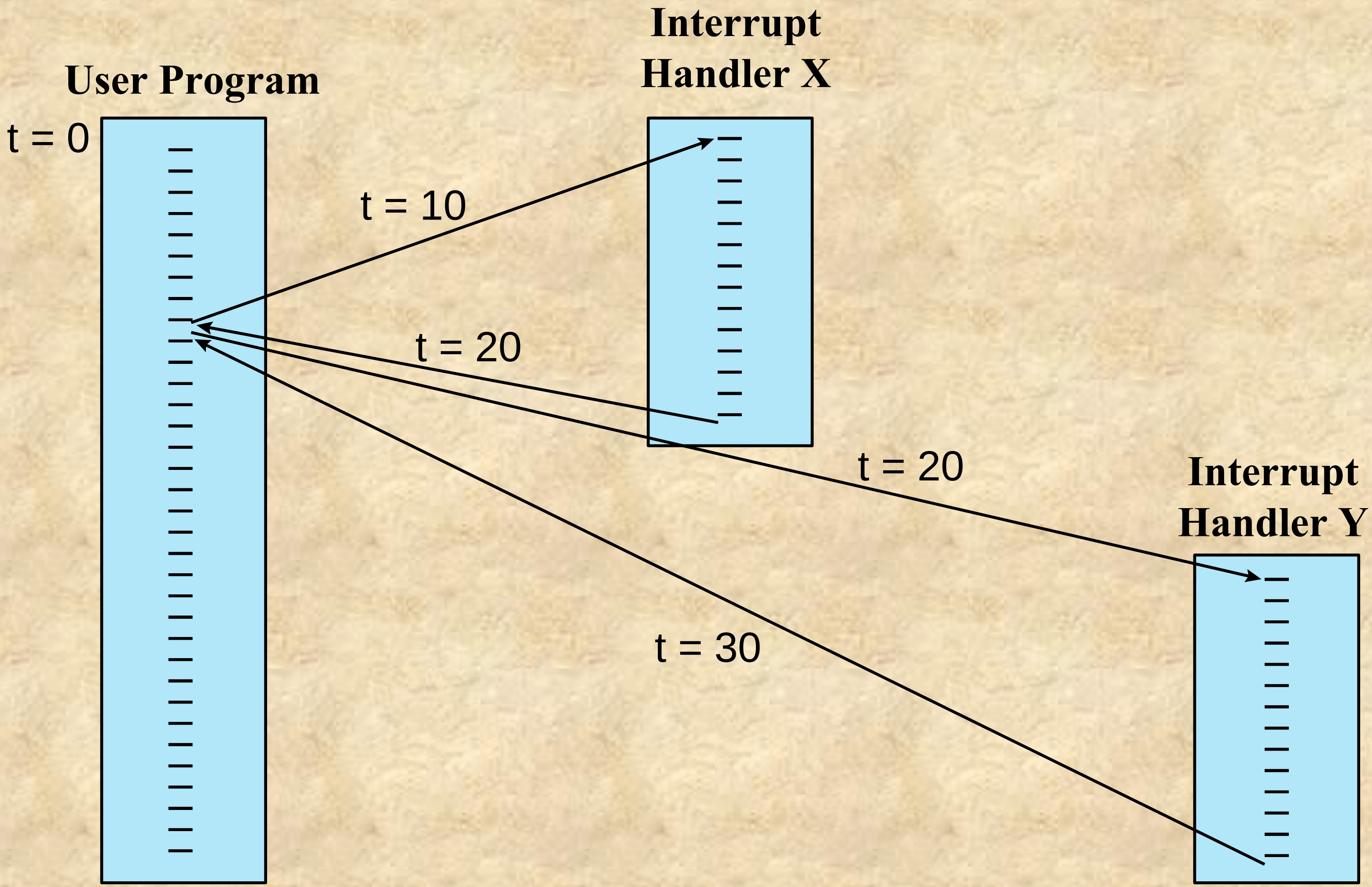
An interrupt may occur while another interrupt is being processed

- e.g. receiving data from keyboard and printing results at the same time

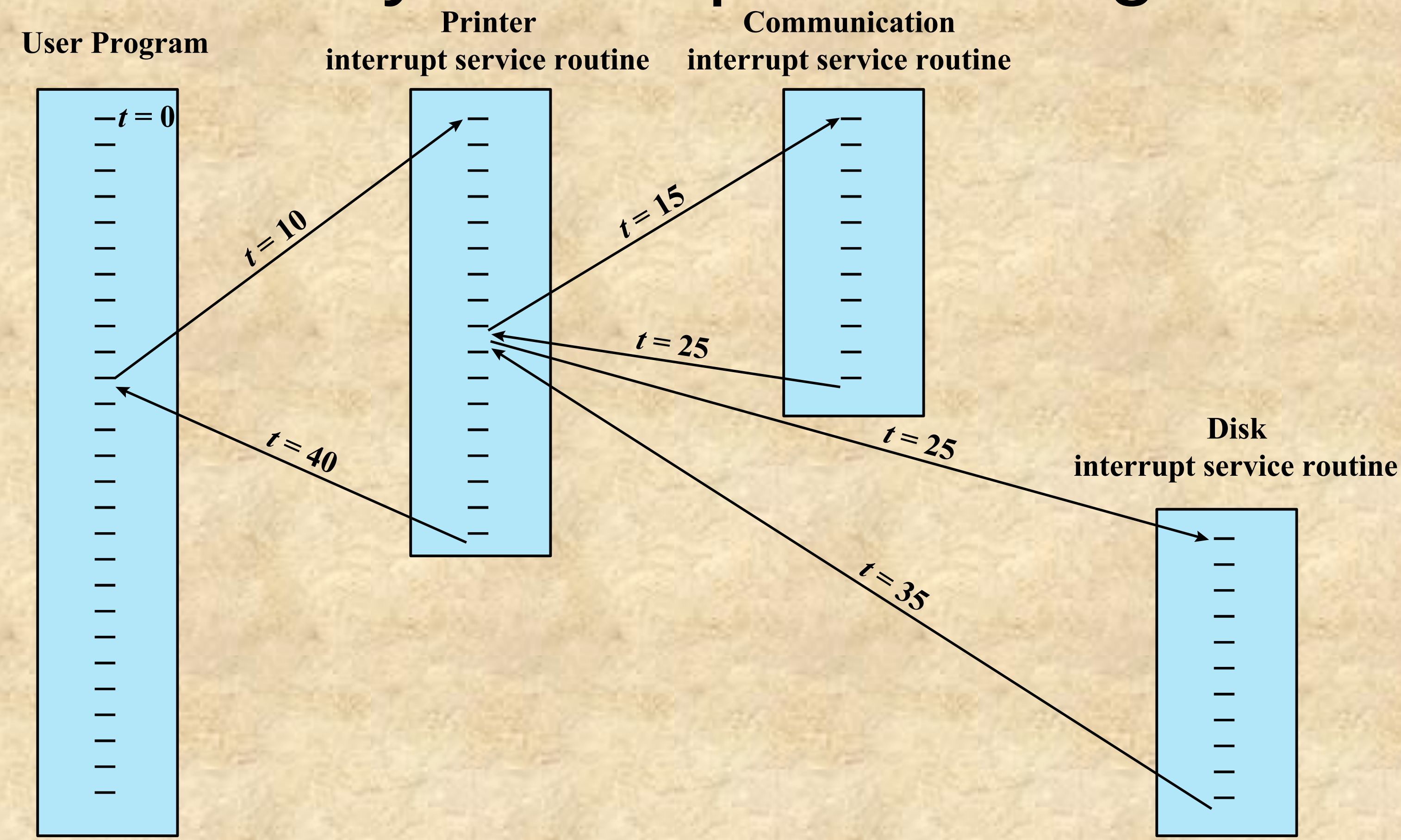
Two approaches:

- Sequential interrupt handling
- Use a priority scheme

Sequential Interrupt Handling



Priority Interrupt Handling



Priority order on devices: Printer < Hard Disk < Keyboard

Allows to incorporate time-critical needs