1. Input and Output of Pass 1 & Pass 2 of Assembler

Pass 1:

- **Input:** Assembly language program (source code)
- **Output:** Symbol Table, Literal Table, Intermediate Code, Location Counter values

Pass 2:

- **Input:** Intermediate Code, Symbol Table, Literal Table
- **Output:** Final Machine Code (Object Code)

Input and Output of Pass 1 & Pass 2 of Macro Processor

Pass 1:

- **Input:** Source program with macros
- **Output:** Macro Name Table (MNT), Macro Definition Table (MDT), Argument List Array (ALA)

Pass 2:

- **Input:** MNT, MDT, ALA, Source program
- **Output:** Expanded Source Code (All macros expanded)

2. Preemptive and Non-Preemptive Scheduling

- **Preemptive:** CPU can be taken away from a running process (e.g., Round Robin, SRTF).
- **Non-Preemptive:** Once a process starts, it runs until completion (e.g., FCFS, SJF Non-Preemptive).

3. Turnaround Time, Response Time, Waiting Time

- **Turnaround Time = Completion Time – Arrival Time**
  (Def : Turnaround time start from process submission to completion of process)
- **Response Time = First Response Time – Arrival Time**
  (Def : It is time from the Submission of Request until the first response is produced)
- **Waiting Time = Turnaround Time – Burst Time**
  (Def : The average period of time a process spends waiting. Process is normally in the ready queue in the ready queue in waiting time)

4. Best Algorithm (FCFS, RR, Priority, SJF)

- **SJF (Shortest Job First)** is best for **minimum average waiting and turnaround time**,
  but **RR (Round Robin)** is better for **time-sharing systems** (fairness).

5. Best Algorithm (LRU, Optimal, FIFO)

- **Optimal** is best (gives the least page faults),
  but it's **theoretical** (requires future knowledge).
- **LRU (Least Recently Used)** is next best in practice.
- **FIFO (First In First Out)** has the poorest performance (may cause Belady's anomaly).

6. What is Macro and Assembler?

- **Macro:** A sequence of instructions given a name; expands automatically when called.
- **Assembler:** A program that converts assembly language code into machine code.

7. Why Banker's Algorithm is Required?

It is used for **deadlock avoidance** — ensures that a system never enters an unsafe state by checking if resource allocation will lead to deadlock or not.

8. What is Deadlock? Four Necessary Conditions

- **Deadlock:** A state where processes wait indefinitely for resources held by each other.

Conditions:

1. Mutual Exclusion
2. Hold and Wait
3. No Preemption
4. Circular Wait

Condition	Meaning	Explanation
1. Mutual Exclusion	Resource is non-shareable	Only one process can use a resource at a time.
2. Hold and Wait	Process holds one resource and waits for another	Process is holding some resource and waiting for more.
3. No Preemption	Resource cannot be forcibly taken	Resource can only be released voluntarily by the process holding it.
4. Circular Wait	Circular chain of processes waiting for each other	Example: P1 → P2 → P3 → ... → P1

11. Difference Between Pass 1 and Pass 2 of Assembler

Feature	Pass 1	Pass 2
Objective	Build Intermediate Code ,Symbol & Literal Tables	Generate Machine Code
Input	(Source Code) Assembly language program	Intermediate Code, Symbol Table, Literal Table
Output	Intermediate Code, Symbol Table, Literal Table	Final Machine Code (Object Code)

12. Drawback of Pass 1 of Assembler

- It cannot generate the final machine code directly since symbol addresses are not yet resolved completely.

13. Difference Between Compiler and Interpreter

Parameter	Compiler	Interpreter
Program Scanning	Scans the entire program at once.	Scans and executes code **line by line**.
Error Detection	Shows **all errors at once** after full scanning.	Shows **errors one at a time**, as each line is interpreted.
Object Code	Converts source code into **object code** (machine code).	**Does not generate** object code.
Execution Time	**Faster execution**, since code is pre-compiled.	**Slower execution**, as each line is interpreted during runtime.
Source Code Need	**Not needed** after compilation.	**Needed** every time the program runs.
Languages Used	Used in languages like **C, C++, C#**.	Used in languages like **Python, Ruby, Perl, MATLAB**.

Types of Errors	Detects **syntactic and semantic errors**.	Detects **syntactic errors only**.
Size & Efficiency	**Larger in size** but more **efficient**.	**Smaller in size**, but less **efficient and slower**.

14. Parts of Assembly Language

1. **Operation Code (Opcode)** – Specifies the operation (e.g., ADD, MOV).
2. **Operand** – Specifies data or memory location.
3. **Label** – Used for identification of instructions.
4. **Comments** – Documentation.

15. What is Process? Different States of Process

Process: A program in execution.

States:

1. New
2. Ready
3. Running
4. Waiting (Blocked)
5. Terminated

A **Process** is a **program in execution**.

When a program is loaded into memory and starts running, it becomes a **process**.

- It has **instructions**, **data**, **memory**, and **CPU time** allocated.
- It is a **dynamic** entity (keeps changing state as it executes).

State	Meaning / Explanation
New	The process is just created and is waiting to be admitted into the ready queue.
Ready	The process is loaded in memory and **waiting for CPU** to start execution.

Running	The process is currently **executing on the CPU**.
Waiting / Blocked	The process is **waiting for some event** (e.g., I/O operation, resource) to complete. CPU is not being used.
Terminated / Exit	The process has **finished execution** and is removed from the system.

16. What is Paging?

A memory management technique where physical memory is divided into fixed-size blocks called **frames**, and logical memory into **pages**. It eliminates external fragmentation.

17. What is Segmentation?

Memory management technique dividing processes into variable-sized **segments** based on logical divisions like functions, arrays, etc.

18. What are Pages and Frames?

- **Page:** Fixed-size block of logical memory (in virtual memory).
- **Frame:** Fixed-size block of physical memory (RAM).
  Each page maps to a frame.

19. Why is CPU Scheduling Algorithm Needed?

To decide which process gets the CPU next — improves CPU utilization, throughput, and response time, and ensures fairness.

20. Types of Schedulers

1. **Long-Term Scheduler (Job Scheduler)**
2. **Medium-Term Scheduler (Swapper)**
3. **Short-Term Scheduler (CPU Scheduler)**

21. Explain Long-Term, Medium-Term, and Short-Term Scheduler

- **Long-Term:** Controls degree of multiprogramming by deciding which processes enter ready queue.
- **Medium-Term:** Swaps processes in/out of memory (handles suspended states).
- **Short-Term:** Selects one process from ready queue to run next on CPU.

Types of Real-Time Applications of HCI

- **Gaming Systems** – Real-time interaction between player and system.
- **Air Traffic Control Systems** – Fast response, high reliability.
- **Robotics / Industrial Automation** – Live human input controlling machines.
- **Virtual Reality / AR Applications** – Real-time motion tracking.
- **Medical Systems** – Surgeon interacts with real-time monitoring devices.

Types of Memory (in HCI context)

We design for three types of human memory:

- **Sensory Memory:** Very brief (less than a second) buffer for sights and sounds.
- **Short-Term (Working) Memory:** Holds a small amount of information (see Miller's Law) for immediate use, like remembering a verification code.
- **Long-Term Memory:** Vast, permanent storage for knowledge, skills, and experiences. We design for recognition (seeing) over recall (remembering).

Principles of Good Design and Bad Interface

Principles of Good Design	Description	Characteristics of Bad Interface	Description
Clarity	Interface is easy to understand and navigate.	**Poor Usability**	Users find it difficult to perform simple tasks.
Consistency	Similar elements behave the same across the system.	**Clutter**	Too many elements make the interface confusing.
Feedback	System responds clearly to user actions (e.g., messages, highlights).	**Inconsistency**	Elements behave differently, forcing users to re-learn.
Error Prevention & Handling	Interface prevents mistakes and gives simple instructions to fix errors.	**Lack of Feedback**	Users are unsure if their actions were registered.

Miller’s Law

Miller's Law (proposed by George Miller in 1956) states that the average person can only hold about **7 ± 2 items** (or "chunks") of information in their working memory at one time.

Application in UX Design:
Designers use this law to organize content into smaller, digestible groups (**chunking**) to reduce cognitive load.

- **Example 1: Phone Numbers** are typically formatted as (XXX) XXX-XXXX instead of a single long string of digits.
- **Example 2: Navigation Menus** often limit the number of primary navigation items to between 5 and 9.
- **Example 3: Credit Card Numbers** are broken into four groups of four digits on the physical card and in online forms.

Interaction Styles (different with advantages and disadvantages)

Style	Description	Advantage	Disadvantage	Real-Time Example
Command Line (CLI)	User types text commands.	Fast, precise, powerful for experts.	High learning curve; hard to learn.	A developer using Git or the Terminal.
Menu-driven	User selects options from a list.	Easy to learn, reduces errors.	Can be slow or "click-heavy."	An ATM machine or a restaurant kiosk.
Direct Manipulation	User clicks, drags, and moves	Intuitive, visual, engaging.	Hard to automate complex tasks.	Dragging files on a desktop; designing in Figma.

	visual objects.			
Natural Language	User speaks or types in human language.	Very natural and accessible.	Can be ambiguous; high error rate.	Using Siri, Alexa, or Google Assistant.

Ergonomics for UX/UI

Ergonomics in UX/UI is about designing for human comfort, efficiency, and safety.

- **Physical Ergonomics:** Deals with things like font size (legibility) and button size/placement (Fitts's Law). Are tap targets on a phone big enough for a thumb?
- **Cognitive Ergonomics:** Deals with mental load. Is the workflow logical? Is the interface consistent? Does it minimize the need to remember information?

How the 8 Golden Rules are Applicable to Figma Designing

Shneiderman's Eight Golden Rules of Interface Design are foundational usability principles. When designing in Figma, these rules guide the creation of user-friendly interfaces:

No.	Golden Rule	Explanation	Example

1.	**Strive for Consistency**	Use consistent layout, colors, commands, and actions across the interface.	MS Word uses same "Save" icon and shortcut (Ctrl + S) across versions.
2.	**Enable Frequent Users to Use Shortcuts**	Provide keyboard shortcuts and command abbreviations to speed up use.	In Gmail, pressing C opens a new compose window instantly.
3.	**Offer Informative Feedback**	Give immediate and clear feedback for every user action.	"Form submitted successfully" message after submitting an online form.
4.	**Design Dialogs to Yield Closure**	Group actions into logical steps with start, progress, and completion message.	Online payments show **"Transaction Successful – Ref ID"** after payment.
5.	**Offer Simple Error Handling**	Prevent errors and provide clear, helpful error messages and recovery options.	"Enter a valid email address" message when user types it incorrectly.
6.	**Permit Easy Reversal of Actions**	Allow users to undo/redo actions to reduce mistakes and stress.	Ctrl + Z to undo changes in Photoshop or MS Word.
7.	**Support Internal Locus of Control**	Users should feel in control of the system, not forced by it.	Clicking "Play" on a music app plays song instantly, as expected.
8.	**Reduce Short-Term Memory Load**	Don't make users remember too much; guide them with visible steps.	Ticket booking shows progress bar: Step 1 → Step 2 → Step 3.

Intranet vs. Internet

- **Intranet:** A private network internal to an organization, using Internet protocols.
- **Internet:** A global, public network of computer networks.

Intranet	Internet
Private network	Public network
Used inside an organization	Used globally
Secure & limited access	Open access

GUI vs. Web Application Types

- **GUI (Graphical User Interface) / Desktop Application:**
    - *Characteristics:* Installed locally on a computer, usually faster performance, high degree of control over the interface, platform-specific.

- **Web Application:**
    - *Characteristics:* Runs within a web browser, accessible from any internet-connected device, updates are deployed centrally (no user installation), dependent on network connectivity.

GUI Application	Web Application
Installed software	Runs in browser
No internet needed usually	Needs internet
Example: MS Word	Example: YouTube

Types of UX/UI Evaluation

1. Heuristic Evaluation:
Experts check the design using basic usability rules to find problems.

2. User Testing:
Real people use the design while we watch to see what confuses them.

3. A/B Testing:
We show two different versions to users and see which one they like or use more.

4. Cognitive Walkthrough:
We imagine we are the user and step-by-step try to complete tasks to see if it is easy.

5. Surveys & Interviews:
We ask users questions to understand their opinions and experiences.

GOMS Models

GOMS (Goals, Operators, Methods, Selection Rules) is a family of human information processing models used to predict and analyze expert user performance. It breaks down tasks into:

- **Goals:** What the user wants to achieve (e.g., "delete file").
- **Operators:** Basic perceptual, cognitive, or motor actions (e.g., "move mouse," "click button").
- **Methods:** Sequences of operators to achieve a goal (e.g., "drag file to trash" vs. "right-click and select delete").
- **Selection Rules:** Rules to choose between alternative methods for a goal.

In Short:

- **Heuristic Evaluation** → *Is the design good?* (Based on rules)
- **Cognitive Walkthrough** → *Can a new user easily do the task?* (Based on steps)

Heuristic Evaluation	Cognitive Walkthrough
Based on usability **rules/heuristics**.	Based on **step-by-step tasks** performed by a user.
Done mostly by **experts**.	Evaluators **act like users** and check each step.
Focuses on **overall interface quality**.	Focuses on **how easily a user can learn and complete tasks**.
Quick evaluation.	**Detailed** and takes more time.
Finds **general usability problems**.	Finds **problems users may face while doing tasks**.

Paradigms for Interaction

Key Paradigms (In Short)

- **Command-Line (CLI):** The user **tells** the computer what to do by typing text commands.
    - *Example:* MS-DOS, Linux Terminal.
- **WIMP (Windows, Icons, Menus, Pointer):** The classic "desktop" model. The user **shows** the computer what to do by pointing and clicking.
    - *Example:* Windows 10, macOS.
- **Direct Manipulation:** The feeling of "physically" moving digital objects (like dragging files or resizing a window) with immediate feedback.
- **Touch & Mobile:** Direct manipulation using **fingers and gestures** (tap, swipe, pinch) on a screen.
    - *Example:* iOS, Android.
- **Ubiquitous Computing (Ubicomp):** "Invisible" computing. Technology is embedded in the environment and acts automatically based on context. The computer **senses** what to do.
    - *Example:* Smart lights, fitness trackers.
- **Augmented/Virtual Reality (AR/VR):** The interface moves from a 2D screen into 3D space.
    - **VR:** You are **inside** a completely digital world.
    - **AR:** Digital information is **overlaid** onto the real world.

Five-Stage Search Framework

This describes the process users go through when seeking information:

1. **Goal Formulation:** Defining what information is needed.
2. **Query Formulation:** Translating the goal into a query the system understands.
3. **Results Examination:** Reviewing the search results.
4. **Information Extraction:** Getting the needed information from the selected result.
5. **Refinement/Reiteration:** Modifying the query or process if the information is not found or is insufficient.

17. Dynamic Queries & Faceted Search

- **Dynamic Queries:** Results change **instantly** when sliders/filters are moved.
  Example: E-commerce price filter.
- **Faceted Search:** User refines results using multiple **independent filters**.
  Example: Flipkart filters — brand, RAM, price, ratings.

Designing for 10-Foot Interfaces Principles

10-foot interfaces are designed for viewing from a distance of about 10 feet (e.g., smart TVs, gaming consoles, set-top boxes). Principles include:

- Big elements and large text
- Simple navigation (remote control friendly)
- Minimal input typing
- High contrast colors
- Clear focus highlight state