EMU8086 & Simple Program

1. প্রশ্ন: EMU8086 কী?
   উত্তর: EMU8086 হলো 8086 microprocessor-এর একটি emulator।
2. প্রশ্ন: EMU8086 কেন ব্যবহার করা হয়?
   উত্তর: Assembly language program লিখে run ও debug করার জন্য।
3. প্রশ্ন: `.model small` মানে কী?
   উত্তর: Code ও data আলাদা segment-এ থাকবে।
4. প্রশ্ন: `.data` segment-এর কাজ কী?
   উত্তর: Variable ও data declare করা হয়।
5. প্রশ্ন: `.code` segment-এর কাজ কী?
   উত্তর: Program instruction লেখা হয়।
6. প্রশ্ন: `print` macro কী করে?
   উত্তর: Screen-এ text output দেখায়।
7. প্রশ্ন: `INT 21H` কী?
   উত্তর: DOS interrupt যা system service দেয়।
8. প্রশ্ন: `AH = 4Ch` কেন ব্যবহার করা হয়?
   উত্তর: Program terminate করার জন্য।
9. প্রশ্ন: Program execution কোথা থেকে শুরু হয়?
   উত্তর: `main proc` থেকে।
10. প্রশ্ন: EMU8086-এ program run করার ধাপ কী?
    উত্তর: Write → Compile → Run।

Input / Output Operation

1. প্রশ্ন: Keyboard input এর জন্য কোন interrupt ব্যবহার হয়?
   উত্তর: `INT 21H`।
2. প্রশ্ন: `AH = 01h` কী কাজ করে?
   উত্তর: Keyboard থেকে একটি character input নেয়।
3. প্রশ্ন: Output দেখানোর জন্য কোন instruction ব্যবহার হয়?
   উত্তর: `AH = 02h`।

4. প্রশ্ন: Output character কোন register-এ থাকে?
   উত্তর: DL register-এ।
5. প্রশ্ন: ASCII code কী?
   উত্তর: Character-এর numeric representation।
6. প্রশ্ন: Input নেওয়ার পর কেন 48 বিয়োগ করা হয়?
   উত্তর: ASCII থেকে numeric value পাওয়ার জন্য।
7. প্রশ্ন: Output দেখানোর আগে কেন 48 যোগ করা হয়?
   উত্তর: Numeric value কে ASCII বানানোর জন্য।
8. প্রশ্ন: Input–Output কোন ধরনের operation?
   উত্তর: Runtime operation।
9. প্রশ্ন: INT 21H ছাড়া I/O করা যায়?
   উত্তর: DOS environment-এ যায় না।
10. প্রশ্ন: I/O operation কেন দরকার?
    উত্তর: User এর সাথে program interaction করার জন্য।

Arithmetic Operation

1. প্রশ্ন: ADD instruction কী করে?
   উত্তর: দুইটা value যোগ করে।
2. প্রশ্ন: SUB instruction কী করে?
   উত্তর: বিয়োগ করে।
3. প্রশ্ন: MUL instruction result কোথায় রাখে?
   উত্তর: AX register-এ।
4. প্রশ্ন: DIV instruction quotient কোথায় রাখে?
   উত্তর: AL register-এ।
5. প্রশ্ন: DIV করার আগে AH কেন clear করা হয়?
   উত্তর: কারণ dividend AX register-এ থাকে।
6. প্রশ্ন: Arithmetic operation কোন level-এর?
   উত্তর: CPU level operation।
7. প্রশ্ন: AL আর AX-এর পার্থক্য কী?
   উত্তর: AL 8-bit, AX 16-bit।
8. প্রশ্ন: Overflow flag কখন set হয়?
   উত্তর: Result limit ছাড়ালে।
9. প্রশ্ন: Single digit ব্যবহার সহজ কেন?
   উত্তর: ASCII conversion সহজ হয়।

10. প্রশ্ন: Arithmetic instruction কেন গুরুত্বপূর্ণ?
    উত্তর: সব computation-এর base।

Branching (If–Else)

1. প্রশ্ন: Branching কী?
   উত্তর: Condition অনুযায়ী decision নেওয়া।
2. প্রশ্ন: CMP instruction কী করে?
   উত্তর: দুইটি value compare করে।
3. প্রশ্ন: JE instruction কী?
   উত্তর: Equal হলে jump করে।
4. প্রশ্ন: JNE instruction কী?
   উত্তর: Not equal হলে jump করে।
5. প্রশ্ন: JG instruction কী?
   উত্তর: Greater হলে jump করে।
6. প্রশ্ন: JL instruction কী?
   উত্তর: Less হলে jump করে।
7. প্রশ্ন: Conditional jump কী?
   উত্তর: Condition true হলে jump করে।
8. প্রশ্ন: Unconditional jump কী?
   উত্তর: Condition ছাড়াই jump করে।
9. প্রশ্ন: Flag register-এর কাজ কী?
   উত্তর: Comparison result ধরে রাখা।
10. প্রশ্ন: If–else কেন দরকার?
    উত্তর: Decision making-এর জন্য।

Control Structure / Jump Table

1. প্রশ্ন: Control structure কী?
   উত্তর: Program flow control করার পদ্ধতি।
2. প্রশ্ন: Jump table কী?
   উত্তর: Multiple condition-এর জন্য jump list।

3. প্রশ্ন: Jump table কেন ব্যবহার করা হয়?
   উত্তর: Multi-way decision দ্রুত করার জন্য।
4. প্রশ্ন: Switch-case assembly-তে কীভাবে হয়?
   উত্তর: Jump table দিয়ে।
5. প্রশ্ন: Menu-driven program কী?
   উত্তর: User choice অনুযায়ী program চালানো।
6. প্রশ্ন: JMP instruction কী?
   উত্তর: Direct jump করে।
7. প্রশ্ন: Jump table কোন register ব্যবহার করে?
   উত্তর: Index register।
8. প্রশ্ন: Control structure ছাড়া কী সমস্যা?
   উত্তর: Program flexible হয় না।
9. প্রশ্ন: Multi-way branching কী?
   উত্তর: একাধিক condition handle করা।
10. প্রশ্ন: Control structure কোথায় বেশি লাগে?
    উত্তর: Complex program-এ।

Loop

1. প্রশ্ন: Loop কী?
   উত্তর: একই কাজ বারবার করা।
2. প্রশ্ন: LOOP instruction কী করে?
   উত্তর: CX decrement করে jump করে।
3. প্রশ্ন: CX register-এর কাজ কী?
   উত্তর: Loop counter রাখা।
4. প্রশ্ন: Loop কেন দরকার?
   উত্তর: Repetitive কাজের জন্য।
5. প্রশ্ন: Infinite loop কী?
   উত্তর: শেষ না হওয়া loop।
6. প্রশ্ন: Loop আর JMP-এর পার্থক্য কী?
   উত্তর: Loop counter দেখে jump করে।
7. প্রশ্ন: Loop-এ INC কেন লাগে?
   উত্তর: Value update করার জন্য।
8. প্রশ্ন: Loop ছাড়া repeat সম্ভব?
   উত্তর: কঠিন।

9. প্রশ্ন: Loop কোথায় বেশি ব্যবহার হয়?
   উত্তর: Series print, array processing।
10. প্রশ্ন: Loop instruction efficient কেন?
   উত্তর: Automatic counter control করে।

Array Operation

1. প্রশ্ন: Array কী?
   উত্তর: একই ধরনের data-এর collection।
2. প্রশ্ন: Assembly-তে array কোথায় থাকে?
   উত্তর: Memory-তে।
3. প্রশ্ন: SI register কেন ব্যবহার হয়?
   উত্তর: Array index access করার জন্য।
4. প্রশ্ন: DI register কেন লাগে?
   উত্তর: Destination index-এর জন্য।
5. প্রশ্ন: Indexed addressing কী?
   উত্তর: Index দিয়ে memory access।
6. প্রশ্ন: Array loop দিয়ে কেন traverse করা হয়?
   উত্তর: Sequential access-এর জন্য।
7. প্রশ্ন: Array element compare করা যায়?
   উত্তর: হ্যাঁ, CMP দিয়ে।
8. প্রশ্ন: Array size কিভাবে ঠিক হয়?
   উত্তর: Declaration অনুযায়ী।
9. প্রশ্ন: Array sequential কেন?
   উত্তর: Memory continuous থাকে।
10. প্রশ্ন: Array কেন দরকার?
   উত্তর: Multiple data manage করার জন্য।

Palindrome / String

1. প্রশ্ন: Palindrome কী?
   উত্তর: উল্টো করলেও একই string।

2. প্রশ্ন: String কী?
   উত্তর: Character-এর sequence।
3. প্রশ্ন: $ কেন ব্যবহার করা হয়?
   উত্তর: String end marker হিসেবে।
4. প্রশ্ন: String length কিভাবে বের করা হয়?
   উত্তর: $ পর্যন্ত count করে।
5. প্রশ্ন: SI register কেন লাগে?
   উত্তর: Start pointer হিসেবে।
6. প্রশ্ন: DI register কেন লাগে?
   উত্তর: End pointer হিসেবে।
7. প্রশ্ন: Palindrome-এ কয়টা compare লাগে?
   উত্তর: Length-এর অর্ধেক।
8. প্রশ্ন: Middle character কেন skip হয়?
   উত্তর: Compare দরকার নেই।
9. প্রশ্ন: String operation কেন কঠিন?
   উত্তর: Memory based হওয়ায়।
10. প্রশ্ন: Palindrome কোথায় ব্যবহার হয়?
    উত্তর: Text validation-এ।

Procedure

1. প্রশ্ন: Procedure কী?
   উত্তর: Reusable code block।
2. প্রশ্ন: CALL instruction কী করে?
   উত্তর: Procedure call করে।
3. প্রশ্ন: RET instruction কী করে?
   উত্তর: Control ফেরত দেয়।
4. প্রশ্ন: Procedure কেন দরকার?
   উত্তর: Code reuse-এর জন্য।
5. প্রশ্ন: Procedure কোথায় থাকে?
   উত্তর: Code segment-এ।
6. প্রশ্ন: Stack-এর ভূমিকা কী?
   উত্তর: Return address রাখা।
7. প্রশ্ন: Procedure modular কেন?
   উত্তর: Program ভাগ করা যায়।

8. প্রশ্ন: Debug সহজ কেন?
   উত্তর: আলাদা code block।
9. প্রশ্ন: Procedure ছাড়া বড় program সম্ভব?
   উত্তর: কঠিন।
10. প্রশ্ন: Procedure বাস্তবে কোথায় লাগে?
    উত্তর: Large application-এ।

CALL & RET

1. প্রশ্ন: Multiple procedure কেন ব্যবহার করা হয়?
   উত্তর: Program organized করার জন্য।
2. প্রশ্ন: CALL কীভাবে কাজ করে?
   উত্তর: Return address stack-এ রাখে।
3. প্রশ্ন: RET কীভাবে কাজ করে?
   উত্তর: Stack থেকে address নিয়ে ফেরত দেয়।
4. প্রশ্ন: Main procedure কী?
   উত্তর: Program starting point।
5. প্রশ্ন: Nested procedure কী?
   উত্তর: Procedure-এর ভিতরে procedure call।
6. প্রশ্ন: CALL stack কী?
   উত্তর: Return address সংরক্ষণ।
7. প্রশ্ন: RET না থাকলে কী হবে?
   উত্তর: Program crash করবে।
8. প্রশ্ন: Procedure execution order কেমন?
   উত্তর: CALL অনুযায়ী।
9. প্রশ্ন: Procedure কেন efficient?
   উত্তর: Code reuse হয়।
10. প্রশ্ন: CALL–RET কোথায় বেশি ব্যবহার হয়?
    উত্তর: Structured programming-এ।