M IT, MUZAFFARPUR

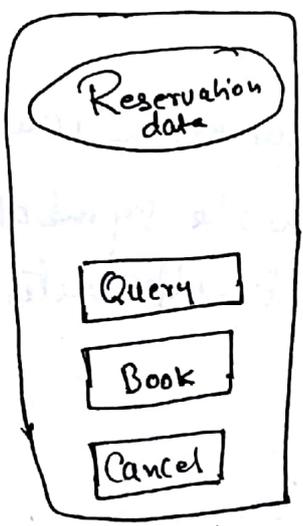Semestr :- 5th   DepH :- IT
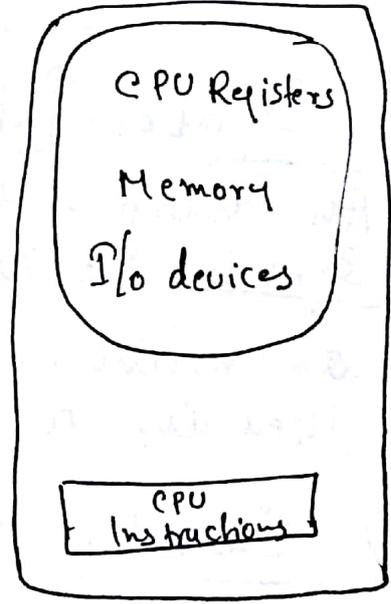
SYSTEM PROGRAMMING (OS IXOS)

Q.1 Define the following terms :-

1) **System Software** :- The System Software is a Collection of programs that bridge the gap between the level at which users wish to interact with the computer and the level at which the computer is capable of operating.

2) **Semantic Gap** :-

Application domain

Execution domain

Semantic Gap.

Scanned by CamScanner

3) Specification Gap.

Reservation data

Query

Book

Cancel

Application domain

Data structures

functions

Programming Language domain
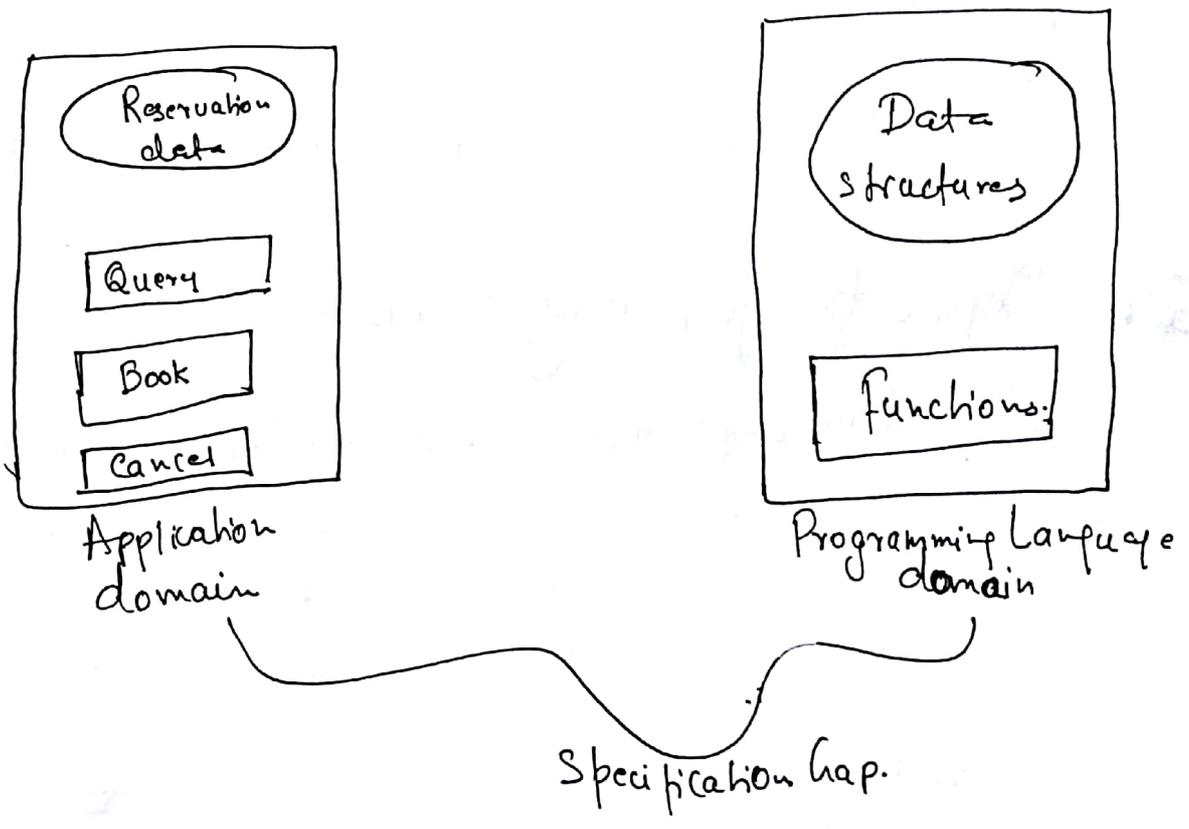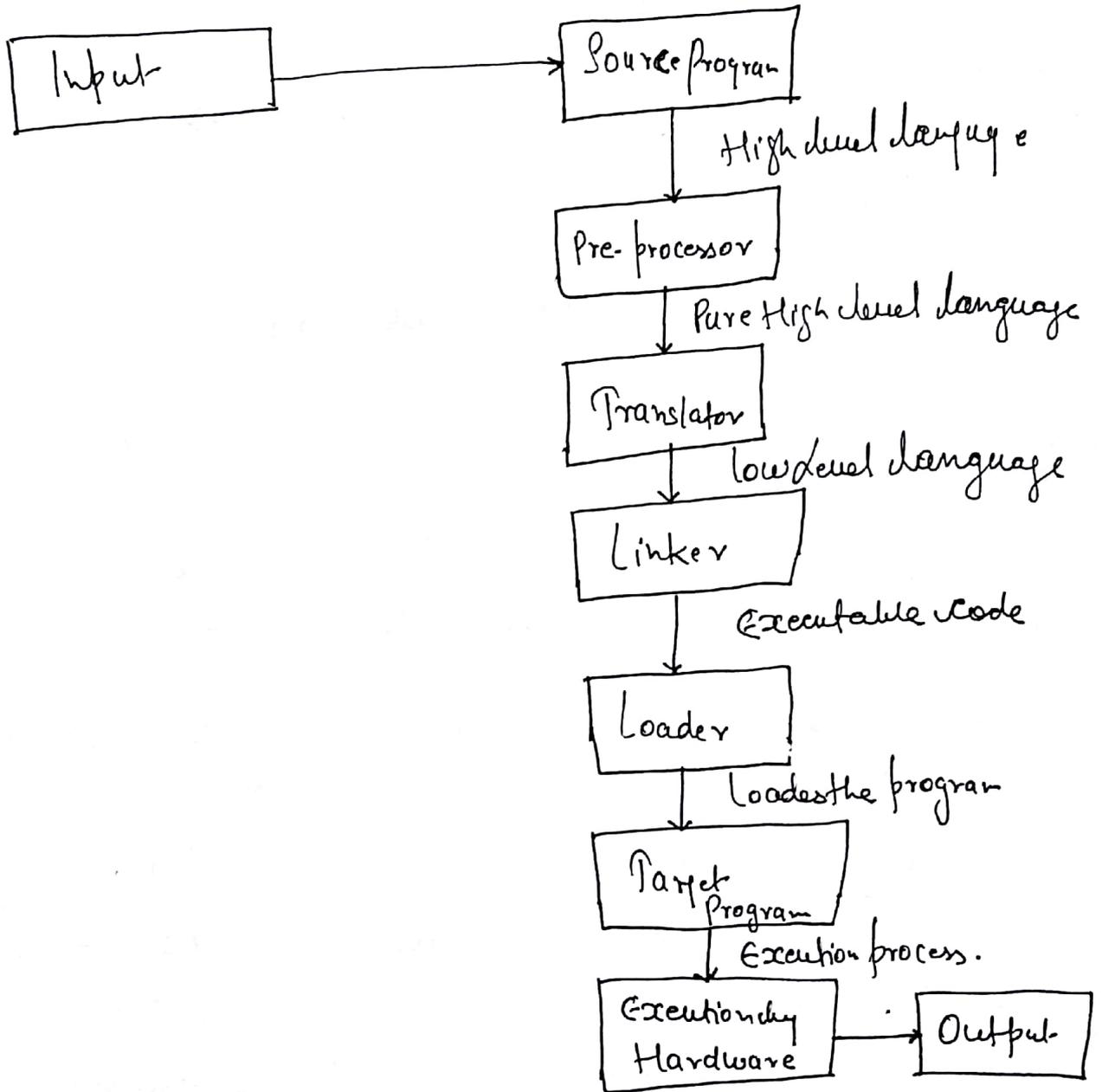
Specification Gap.

4) Symbol Tables

An identifier used in the source program is called a symbol. A language processor uses the symbol table to maintain information about attributes of symbols used in a source program.
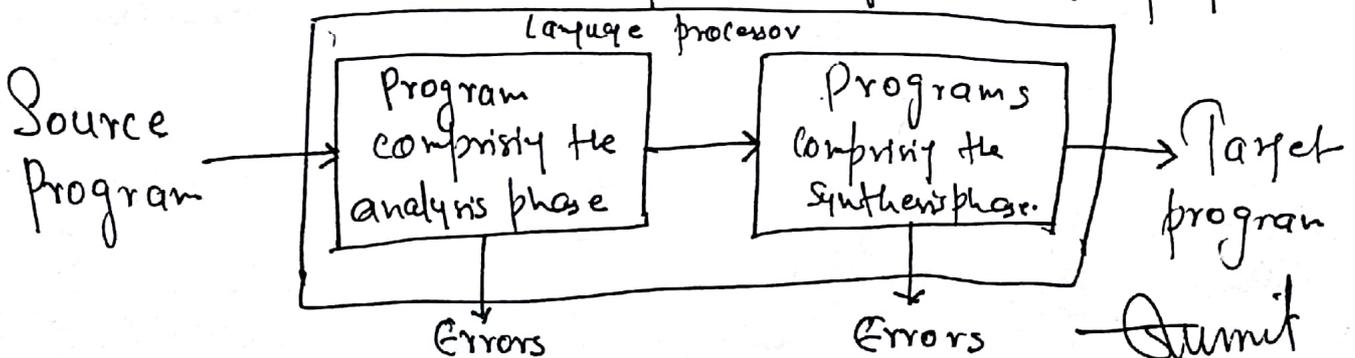
5) Execution Gap

Execution Gap.

Data Structures

functions

Programming Language domain

CPU registers
Memory
I/o devices

CPU instructions

Execution domain

Q2 (i).

```
┌──────────┐                    ┌──────────────┐
│  Input   │ ─────────────────▶ │Source Program│
└──────────┘                    └──────────────┘
                                        │ High level language
                                        ▼
                                ┌──────────────┐
                                │ Pre-processor│
                                └──────────────┘
                                        │ Pure High level language
                                        ▼
                                ┌──────────────┐
                                │  Translator  │
                                └──────────────┘
                                        │ low level language
                                        ▼
                                ┌──────────────┐
                                │    Linker    │
                                └──────────────┘
                                        │ Executable code
                                        ▼
                                ┌──────────────┐
                                │    Loader    │
                                └──────────────┘
                                        │ Loades the program
                                        ▼
                                ┌──────────────┐
                                │    Target    │
                                │   Program    │
                                └──────────────┘
                                        │ Execution process.
                                        ▼
                                ┌──────────────┐      ┌────────┐
                                │ Execution by │ ───▶ │ Output │
                                │   Hardware   │      └────────┘
                                └──────────────┘
```

ii) Language processing : Analysis of the source program + Synthesis of the target program.

```
                    ┌──────────── Language processor ────────────┐
                    │ ┌───────────────┐      ┌───────────────┐    │
  Source            │ │  Program      │      │  Programs     │    │        Target
  Program  ───────────▶│ comprising the│ ───▶ │ comprising the│ ─────────▶ program
                    │ │ analysis phase│      │ Synthesis phase│   │
                    │ └───────┬───────┘      └───────┬───────┘    │
                    └─────────┼──────────────────────┼───────────┘
                              ▼                       ▼
                           Errors                  Errors        Sumit
```

(iii)

a) ORIGIN

ORIGIN < address specification>.

b) EQU

<symbol> EQU <address specification>.

c) LTORG.

Q3   Differentiate between the following

1) a) Binders helps in the association of an attribute of a program entity with a value.
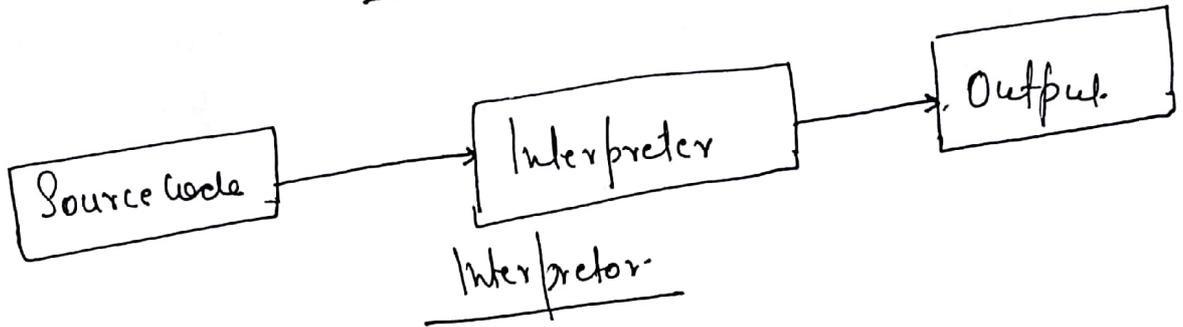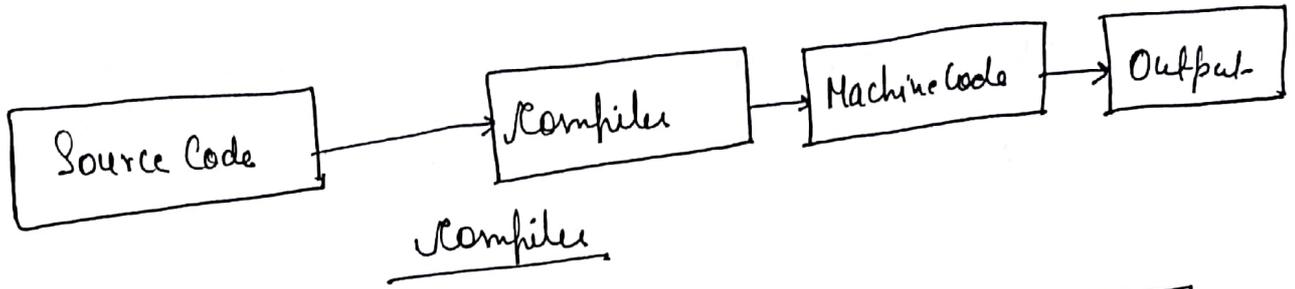
b) Overlays:- An overlay is a part of a program that has the same load origin as some other part(s) of the program. A program containing overlays is called an overlay structured program. It consists of the following:-

  • A permanently resident part, called the root.
  • A set of overlays that would be loaded in memory when needed.

2) **Compiler and Interpreter.**

```
┌──────────────┐        ┌──────────┐      ┌──────────────┐   ┌─────────┐
│ Source Code  │───────▶│ Compiler │─────▶│ Machine Code │──▶│ Output  │
└──────────────┘        └──────────┘      └──────────────┘   └─────────┘
```

Compiler

```
                                                            ┌─────────┐
                                                     ──────▶│ Output  │
                        ┌──────────────┐                    └─────────┘
┌──────────────┐        │ Interpreter  │────────
│ Source Code  │───────▶│              │
└──────────────┘        └──────────────┘
```

Interpretor

3) **Top-down parsing and Bottom-up Parsing.**

a) When the parser starts constructing the parse tree from the start symbol and then tries to transform the the start symbol to the input, it is called top-down parsing.

b) Bottom up parsing starts with the input symbols and tries to construct the parse tree up to the start symbol.

4) **Syntax and Semantics**

a) Syntax refers to the structure/form of the code that a specific programming language specifies.

b) Semantics deals with the meaning assigned to the symbols characters and words.

Q4 (i) a) **User-centric View:-** includes system programs that provide user convenience by permitting a user to specify her computational needs in a manner of her own choice.

b) **System-centric View:-** includes system programs that provide effectiveness in the Simultaneous Servicing of many users computational needs by interleaving Servicing of many programs & ensuring non-interference.
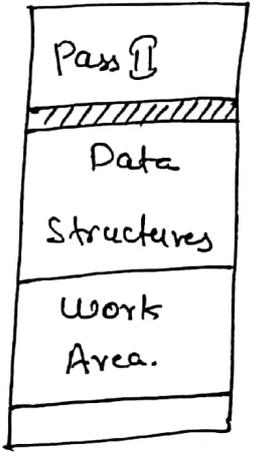
ii) <u>Different types of Intermediate Code Representation</u>
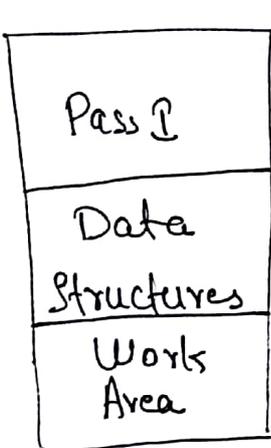
<u>Variant I</u>
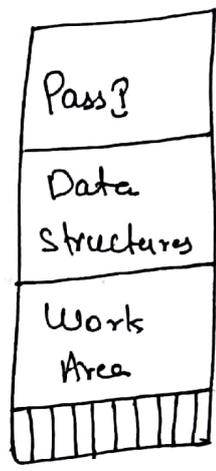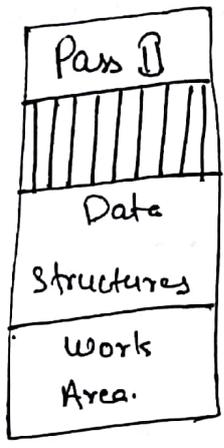
<u>Variant II</u>

(iii) a) <u>START</u> :- The START directive instructs the assembler to place first word of the target program generated by it in the memory word having the address < Constant>
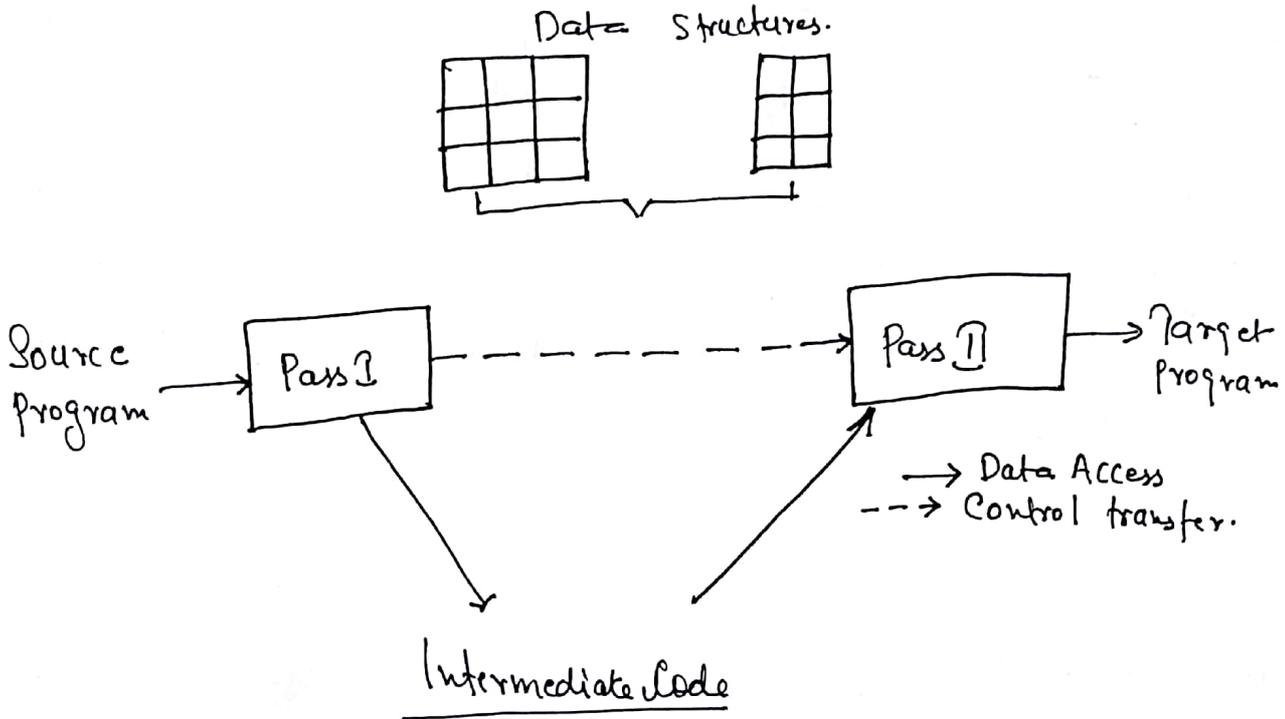
<u>Syntax</u> :-

START < constant >

b) <u>END</u> :- The END directive indicates the end of the source programm.

END [< operand specification >].

Q(5):-

(a)

Pass Structure of Assembler

Data Structures.

Source Program → Pass I - - - - - - - → Pass II → Target Program

→ Data Access
--→ Control transfer.

Intermediate Code

Flow chart for one-pass assembler.

b)

Initialize

Read

Label on card?

Yes → Store Label in Symbol table with LC value

No → Search Pseudo-op Table (PoT) — found → TYPE — GND → go to Pass 2

USING DROP

Not found

Search Machine-op Table (MOT)

TYPE → Determine length of data space reqd.

Get length of Instruction. → Process literals → Update LC