

Q1. What is forward reference? How it is handled in 2-Pass assembler?

⇒ A forward reference occurs when a label or symbol is used before it is defined in the code

- For ex. If you have a jump instruction that refers to a label that appears later in the program, this is a forward reference.
- In 2-pass assembler, forward references are handled in following way:

• First Pass :-

1. Symbol table creation :- The assembler scans the entire source code to create a symbol table. This table maps labels and symbols to their respective memory addresses.
2. Location Counter :- It keeps track of the location counter, which helps in assigning addresses to instructions and data.
3. No code generation :- The assembler does not generate any machine code. It only records the addresses of all labels.

• Second Pass :-

1. Code Generation :- The assembler scans the source code and using the symbol table it generates the final machine code.
2. Address Resolution :- It looks up the symbol table to find address of the label and replaces placeholder with correct address.

Q2. What is ORIGIN statement?

⇒ This can be used to indirectly assign values to symbols.

- When this statement is encountered during assembly of a program, the assembler resets its location counter

- The ORG statement will thus affect the values of all the labels defined until next ORG.
- When ORG without an specific value is encountered, the previously saved location counter value is restored.
- Syntax of ORIGIN directive is
`ORIGIN <address spec>`
 where, <address spec> is an <operand spec> or <constant>
- This type of statement is useful when the target program does not consist of consecutive memory words.
- No forward reference is allowed.
- For ex. symbol : 6 bytes. FLAGS : 2 bytes
 value : 1 word LDA VALUE, X.

Q3. Explain EQU statement with Example.

⇒ Most assemblers provides an assembler directive that allows the programmer to define symbols and specify their values.

- Syntax is:
`symbol EQU value.`
- One common use of EQU is to establish symbolic names that can be used for improved readability in place of numeric values.
- Another common use of EQU is in defining mnemonic names for registers.
- For ex.
`A EQU 0`
`X EQU 1`
`L EQU 2.`
- These statements cause symbols A, X, L ... to be entered into symbol with their corresponding values 0, 1, 2, 3 ...

84. Explain variants of intermediate code?
 ⇒ Intermediate code consists of a set of IC units. Each of its consists of.

Address	Opcode	Operands
---------	--------	----------

- Variant 1 :-

① Single digit number is the code used for a register. This is the first operand.

② Memory operand is a second operand and represented in the following format :- (operand class, code).
 where, operand class = C (Constant), S = Symbol or L = literal.

③ For constant, there is internal representation of itself e.g. START 300 is (C, 300).

④ For symbol, it contains the original number of the operand's entry in SYMTAB or LITAB.

For ex. Symbol = xyz ; Literal = '25'
 (S, 17) ; (L, 35)

⑤ For forward reference,

MOVER AREG, A

A is entered in a symbol table at number n. It is represented as (S, n) in IC.

- Variant 2 :-

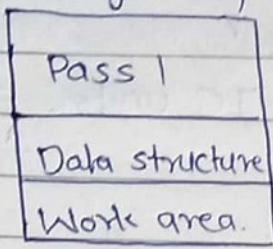
① It differs from variant 1 in operand fields of the source elements.

② Processing of the operands field is essential to support LC processing in declarative and assembler.

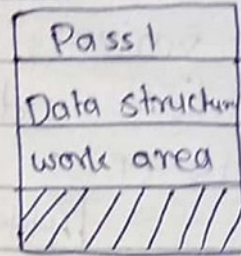
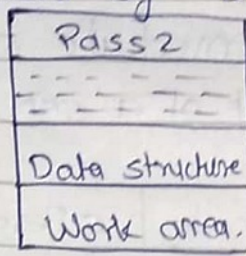
③ Operand field is processed only to identify literal reference for imperative statements.

④ In literal table, literals are entered and represented as (L, m) in IC.

- Memory Requirement by both variants.



Variant (1)



Variant (2)

Q5. Which data structures are used in Pass 1?

⇒ 1. OPTAB :- It consists of mnemonic opcode, class & mnemonic info field. The class field indicates the statement types, i.e. imperative, declaration or assembler.

- For ex. Mnemonic Class Mnemonic info.

MOVPR IS (04, 1)

DS DL R#7

START AD R#11

2. SYMTAB :- Symbol Table consists of the addresses and length of opcodes/symbols.

- For ex. Symbol Address length.

AGAIN 302 1

TERM 314 1

LAST 316 1

3. LITTAB :- Literal Table consists of literal and address fields.

- Processing of an assembly statement begins with the processing of its label field. If label contains a symbol, it is copied into a new entry.

- For ex. Literal Address

= '5'

= '1'

= '1'

SPPU-TE-COMP-CONTENT - KSKA Git

Q6. Which data structures are used in Pass 2?

⇒ 1. SYMTAB :- This table, populated during Pass 1, is used to resolve addresses of symbols during code generation.

2. LITTAB :- During Pass-2 the assembler uses this table to replace literal references with their corresponding addresses.

3. OPTAB :- During Pass-2, the assembler uses OPTAB to translate mnemonic opcode into their corresponding machine code instructions.

4. Location Counter :- The location counter keeps track of the current address in memory where the next instruction or data will be placed.

5. BASETAB :- This table is used in assemblers to keep track of base registers and their contents.

6. Modification Records :- These records are used to keep track of addresses that needs to be modified during the linking process. Essential for relocatable code.

Q7. Give example of LTORG statement?

⇒ - LTORG allows placing literals into a pool at some other location in the object program.

- Directive LTORG creates a literal pool that contains all of the literal operands used since beginning of program.

- For ex.

START 0

L 1, =F'5'

A 1, =F'10'

LTORG

L 2, =F'15'

A 2, =F'20'

LTORG

END

Date

--	--	--

- The First LTORG collects literals and place them into literal pool.
- The second LTORG collects literals and place them into another literal pool.