Prepared By Abhiyan Dev Rimal
https://github.com/abhirimal/be-software-notes

Assembler

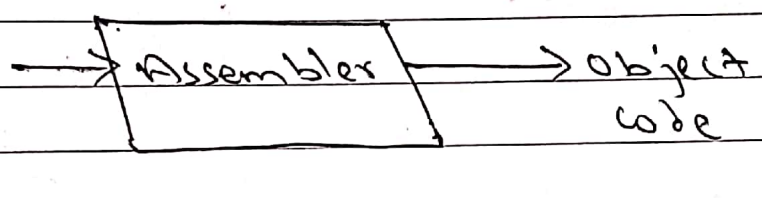
Chapter = 2 (2 or 3 Q)

1) Assembler

→ (it translates mnemonic instruction into machine code.)

→ Assembler is a system software which is used to convert an assembly language program into its equivalent object code.

→ The design of an assembler depends upon machine architecture.



2) Fundamental functions of Assembler

→ convert assembly language into its equivalent object code.

→ assigning machine address to symbolic labels used by programmers.

3) Assembler directives

i) START → specify name and start address of program.

ii) END → indicate end of program.

iii) BYTE → generate character or hexadecimal constant, occupying as many bytes as needed to represent the constant

iv) WORD → generate one-word integer constant and store constant in memory.

Scanned with CamScanner

v) RESB → to reserve specified byte of memory
vi) RESW → to reserve specified word of memory but do not store constant in memory.

14) What is one-pass and multi pass assembler? which one do you prefer while designing an assembler? Justify your answer.
or
write advantage and disadvantage/drawbacks of one pass assembler over two pass assembler?

Ans: Most assembler are two pass assemblers that process source program into two passes.
→ Assembler design deals with
a) One pass Assembler
b) Multi pass Assembler

a) One pass assembler
→ A single/one pass assembler scans the program only once and creates a equivalent object code.
i.e it performs translation in one pass only.
→ The main problem in one pass assembler is forward referencing.
→ forward referencing is handled by back patching.
→ Back patching is handled by TII (Table of Incomplete Instruction).
→ It is faster than two pass assembler.

Scanned with CamScanner

→ There are two types of one pass assembler

1) load and go assembler

→ generates object code in memory for immediate execution.

→ no object program is written out, no loader is needed.

2) the other

→ produces the usual kind of object program for later execution.

1 How forward references are handled in one pass assembler?

→ Two types of forward reference :-

i) Data items

ii) labels on instruction

→ Solution

i) → Eliminating forward reference to data items can be easily done by asking the programmer to define variables before using them.

ii) → However, eliminating forward reference to instruction cannot be easily done.

→ there is no good solution

→ it can be avoided by putting the restriction.

→ Data structure used in one pass assembler are Symbol table, literal table, T I I (Table of Incomplete Instructions)

Scanned with CamScanner

/Multipass

b) Two pass assembler

-> It performs the translation in two pass or more

-> Multipass assembler means more than one pass is used by assembler.

-> There is a generation of intermediate file.

-> It solves the problem of forward reference by devoting one pass to exclusively resolve all forward references and then generate object code with no hassles in the next pass.

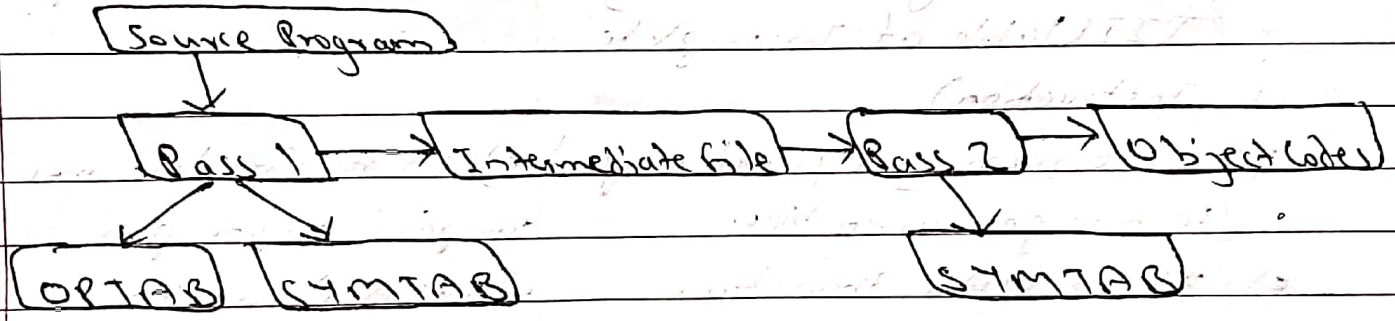


fig:- Two pass assembler

function of two pass assembler

i) Pass 1

-> Assign address to all statements in program

-> Save addresses assigned to all labels for use in Pass 2.

-> Perform some processing of assembler directives.

ii) Pass 2

-> Assemble instructions

-> generate data values defined by BYTE, WORDS.

-> Perform processing of assembler directives not done in Pass 1.

-> write the object program.

Scanned with CamScanner

Q) Difference between

Single One Pass Assembler	Multi Two Pass Assembler
1) It performs translation in one pass only.	1) It performs translation in two pass.
2) Intermediate code are not generated	2) Intermediate code are generated
3) forward referencing is handled by back patching.	3) After pass one all symbols and literals are getting address.
4) Back patching is handled by TII (Table of Incomplete Instruction)	4) No need of back patching.
5) More memory required compared to two pass assembler.	5) It requires less memory compared to one pass
6) It is faster than two pass assembler	6) It is slower than one pass assembler.
7) Data structure used are SYMTAB, TIS, etc.	7) Data structure used are SYMTAB, LOPTAB, etc.

Drawback of single pass assembler over two pass

→ Two pass is efficient than single pass assembler.

→ Two pass assembler solves the problem of forward referencing easily.

Scanned with CamScanner

III 5) Explain load and go assembler.

Ans: Load and go assembler is a type of one pass assembler.

-> It reads program only once and translate it into object code.

-> It generates object code in memory for immediate execution.

-> No object program is created.

-> No loader is needed.

-> It is useful in a system with frequent program development and testing.

-> As program are ~~re-run~~ re-assembled nearly every time they run, efficiency of assembly process is an important consideration.

-> It avoids overhead of writing object program out and reading back in.

Scanned with CamScanner

LOCCTR, OPTAB, SYMTAB

16) Discuss the roles of data structure OPTAB, SYMTAB & LOCCTR in machine dependent Pass 1 Algorithm

→ The machine dependent pass 1 algorithm ~~defines~~ does
- → assign address to all statements in program.
- → save addresses assigned to all labels for use in Pass 2.
- → perform some processing of assembler directives.

→ The data structures used in Assembler are:-

i) OPTAB (Operation code Table)

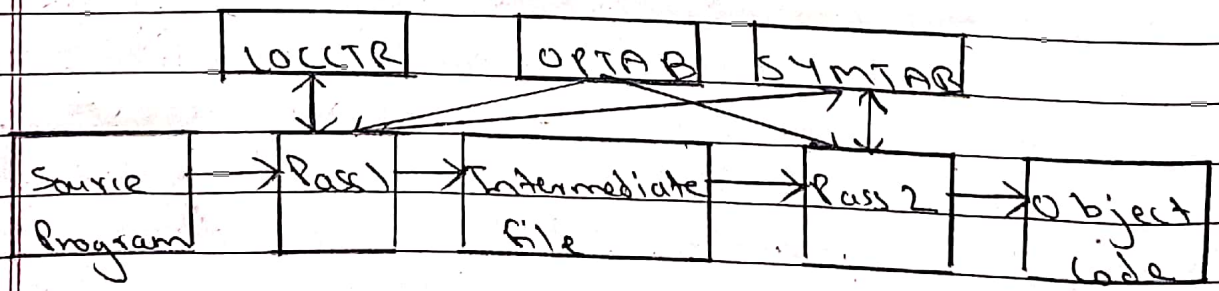
→ used to lookup mnemonic operation code and translate them into their machine equivalent.

ii) Symbol Table (SYMTAB)

→ it stores name and values assigned to table.

iii) Location Counter (LOCCTR)

→ variable used to help in the assignment of addresses.



Scanned with CamScanner

i) LOCCTR (Location counter)

-> variable used to help in assignment of addresses.

-> it is initialized to the beginning address specified in START statement.

-> LOCCTR ← LOCCTR + (instruction length)

-> The current value of LOCCTR gives the address to the label encountered.

ii) OPTAB (Operation code Table)

-> contains mnemonics operation and its machine language equivalent.

-> Mnemonic operation codes <=> Machine code

-> also contains information about instruction format and length.

-> In Pass 1 -> OPTAB is used to lookup and validate operation code in source program.

-> In Pass 2 -> used to translate operation code to machine language equivalent.

-> Organised as a hash table.

iii) SYMTAB (Symbol Table)

-> include name and value (address) for each label in the source program.

-> include flags to indicate error condition.

-> In Pass 1 -> labels are entered into SYMTAB, along with assigned address from LOCCTR.

-> In Pass 2 -> symbols used as operand are lookup in SYMTAB to get address.

Scanned with CamScanner

direct addressing = absolute addressing

Example:-

* Symbol Table (SYMTAB)

Line	Symbol
STRPY	1000
FIRST	1000
ZERO	101E

* OPTAB Example

-> Mnemonic and opcode table given in question.

III 7) What is the advantage of relative addressing mode over absolute addressing mode?

Relative addressing mode	Absolute addressing mode
1) Address is calculated by adding the contents of the CPU register with the address part of the instruction	1) The address for fetching the operand is already provided in the address part of the instruction. It is not calculated.
2) Relative mode has three types: PC Relative, Index Register Relative, Base Register Relative Addressing modes.	2) It has no types. (Absolute addressing mode is also known as direct addressing mode.)
3) It results in shorter address fields in the instruction format.	3) It results in longer address field as it requires more bits to represent memory address.
4) It is used to achieve program relocation during run time.	4) It is used to access static data.

Scanned with CamScanner

5) It is used to represent array addressing records.	5) It is used to implement variables.
6) It is used to handle recursive procedures.	6) It cannot handle recursive procedures.

11 8) What is literal? Explain its handling during Pass 1 and Pass 2

Ans. The programmer writes the value of a constant operand as a part of the instruction that use it.

-> This avoids having to define constant elsewhere in program and make label for it.

-> Such an operand is called literal.

-> Handling of literal during Pass 1 and Pass 2.

Pass 1:

-> Build LITTAB with literal name, operand value and length, leaving the address unassigned.

-> When LTORG or END statement is encountered, assign an address to each literal not yet assigned an address.

Pass 2:

-> Search LITTAB for each literal operand.

-> Generate data values using BYTE or WORD.

-> Generate modification record for literals that represent an address in program.

Scanned with CamScanner

classmate
Date
Page

1111 Q) Write about program blocks and control section

Ans: The source program logically contained subroutines, data areas, etc but assembles handle them as a single entity resulting in a single block of object code.

→ Some assembler provides flexible handling by allowing the generated machine instruction and data to appear in object program in different order. They are:-

i) Program block

→ segments of code that are rearranged with the single object program unit.

ii) Control sections

→ segment of code that are translated into independent object program units.

(Explain) ~~above~~:

i) Program block

→ Program block allow the generated machine instructions and data to appear in the object program in a different order.

→ It refers to the segment of code that are rearranged with the single object program unit.

→ Assembler directive 'USE' is used.

Scanned with CamScanner

-> The assembler directive 'USE' indicates which portion of source program belongs to various blocks.
-> if no 'USE' statements are included, the entire program belongs to single block.
-> The 'USE' ~~signal~~ statement signals the beginning of block.
-> A program block may contain several separate segments ~~to~~ of the source program.
-> the assembler will rearrange these segments together to gather pieces of each block.

Advantages
-> No need of program ~~redability~~ record.
-> improve program redability.
-> Simply, LTORG statements is used to ensure literals are placed a head of large data areas.

ii) Control Section and Program linking
-> Control section can be loaded and relocated independently of other.
-> Segment of code that are translated into independent object program unit.
-> A control section is a part of the program that maintains its identity after assembly.
-> The programmer can assemble, load and manipulate each of these control section separately.

Scanned with CamScanner

-> Because of this, there should be some means for linking control sections together.
For example: instructions in one control section may refer to the data or instructions of other control sections.

-> Since control sections are independently loaded and relocated, the assembler is unable to process these references in the usual way.

-> Such references between different control sections are called external references.

-> The assembler generates the information about each of the external references that will allow the loader to perform the required linking.

-> When a program is written using multiple control sections, the beginning of each of the control section is indicated by an assembler directive.

-> Assembler directive 'CSECT' is used.

10) How does program blocks and control section make flexible in handling of the source and object program.

Explain program block & control section.

Scanned with CamScanner