

Assemblers



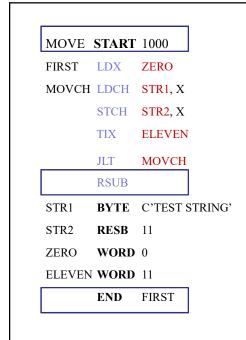
- Features that are fundamental and should be found in any Assembler
- Features closely related to the machine architecture
- Features that are commonly found in implementations of this type of software and are relatively machine-independent

IIT ROORKEE

- Fundamental functions that any assembler must perform
 - Translating mnemonic operation codes to their machine language equivalents
 - Assigning machine addresses to symbolic labels used by programmers
- Features of the assembler that depend heavily on the source language it translates and the machine language it produces
 - Different machine instruction formats
 - Different addressing modes
- Features that have no direct relation to machine architecture
 - More than one program blocks, control sections etc., use of literals

Assembler directives

Specify name and starting address for the program
Indicates the end of the program and (optionally) specify the first executable instruction in the program
Generate character or hex constant, occupying as many bytes as needed to represent the constant
generate one word integer constant
reserve the indicated number of bytes for a data area
reserve the indicated number of words for a data area



The translation of the source program to object code requires

- 0. Process assembler directives
- Convert mnemonic operation codes to their machine language equivalents
- Convert symbolic operands to their equivalent machine addresses

MOVE START 1000 FIRST LDX **ZERO** MOVCH LDCH STR1, X STCH STR2, X TIX **ELEVEN** JLT MOVCH **RSUB** STR1 BYTE C'TEST STRING' STR2 RESB 11 WORD 0 ZERO ELEVEN WORD 11 END **FIRST**

The translation of the source program to object code requires

- 3. Build the machine instructions in the proper format
- 4. Convert the data constants specified in the source program into their internal machine representations
- Write the object program and the assembly listing

- Can we do all of the above by sequential processing of the source program?
- forward references
- Therefore most assemblers make two passes over the source program.
 - Pass 1: scan the source program for label definitions and assign addresses
 - Pass 2: perform actual translation and generate object code

SIC Assembler V1.2						
1000	test	start	1000			
1000	first	lda	five			
1003		sta	alpha			
1006		ldch	charz			
1009		stch	c1			
:		:				
100C	alpha	resw	1			
100F	five	word	5			
1012	charz	byte	c'Z'			
1013	с1	resb	1. In dec.			
1014		end	first			

SIC Assembler V1.2							
1009 : 100C 100F	00100F 0C100C 501012 541013 : 000005 5A	test first alpha five charz c1	start Ida sta Idch stch : resw word byte resb end	1000 five alpha charz c1 1 5 c'Z' 1 first			

Assemblers



- Translates the source program to object code.
- This requires
 - 0. Process assembler directives
 - 1. Convert mnemonic **operation codes** to their machine language equivalents
 - 2. Convert **symbolic operands** to their equivalent machine addresses
 - 3. Build the machine instructions in the proper format
 - 4. Convert the data constants specified in the source program into their internal machine representations
 - 5. Write the object program and the assembly listing

IIT ROORKEE

SIC Assembler (Pass 1 – Define Symbols)

Input: ?

- · Assign addresses to all statements in the program
- Assign values to LABELs
- Save the values (addresses) assigned to all labels for use in Pass 2
- Perform some processing of assembler directives
- Record errors
- · Generate intermediate file

Output: ?

Data Structures

- Operation Code Table (OPTAB):
 - mnemonic operation code and its machine language equivalent.
 - information about instruction format and length.
 - Usually organized as a HASH TABLE, with mnemonic operation code as the key.
 - OPTAB is a static table
- Symbol table (SYMTAB):
 - name and value (address) for each label together with flags to indicate error conditions
 - (May also contain) information about the instruction labeled
 - Frequent insertions
 - rare deletions
 - Usually organized as a HASH TABLE
- Location Counter (LOCCTR)

SIC Assembler (Pass 2 – Generate Object Program)

- · Assemble instructions
 - Process OPCODE
 - Process LABELS
- Perform processing of assembler directives not done during pass 1
 - Generate data values defined by BYTE, WORD etc.
- Write the object program and the assembly listing.
- Report errors