WEEK-14

3-8 Assembler.

• Mnemonic instructions, such as **LOAD 104**, are easy for humans to write and understand.

• They are impossible for computers to understand.

• *Assemblers* translate instructions that are comprehensible to humans into the machine language that is comprehensible to computers

– We note the distinction between an assembler and a compiler: In assembly language, there is a one-to-one correspondence between a mnemonic instruction and its machine code. With compilers, this is not usually the case.

• Assemblers create an *object program file* from mnemonic *source code* in two passes.

• During the first pass, the assembler assembles as much of the program is it can, while it builds a *symbol table* that contains memory references for all symbols in the program.

• During the second pass, the instructions are completed using the values from the symbol table.

• Consider our example program (top).

Note that we have included two directives **HEX** and **DEC** that specify the radix of the constants.

Address		Instruction	
	100	Load	X
	101	Add	Y
	102	Store	Z
	103	Halt	
X,	104	0023	
Y,	105	FFE9	
Z,	106	0000	

- During the first pass, we have a symbol table and the partial instructions shown at the bottom.

1	X
3	Y
2	Z
7 0 0 0	

X	104
Y	105
Z	106

- After the second pass, the assembly is complete.

1 1 0 4
3 1 0 5
2 1 0 6
7 0 0 0
0 0 2 3
F F E 9
0 0 0 0

Address		Instruction	
	100	Load	X
	101	Add	Y
	102	Store	Z
	103	Halt	
X,	104	DEC	35
Y,	105	DEC	-23
Z,	106	HEX	0000