Topic for today:

More on assembly language and assemblers

Recall: Assembly Language

Assembly language is a human-readable representation of machine language. The basic principle is that one line of assembly language corresponds to one line of machine language. Both operations and addresses are expressed symbolically rather than in binary.

Recall: Fields in an assembly language instruction

An instruction in a typical assembly language (including the Marie assembly language) has four fields: label, operation, operand (or address), and comment.

Recall: Assembler

The software that converts assembly language to machine language is called an assembler.

An assembler takes a source file and creates an object file.

Assembler Directive

An assembler *directive* is a command to the assembler in an assembly language program. It is not translated into machine code.

Two-pass assembler

A two-pass assembler (which is the typical design for an assembler) translates assembly language to machine language by going through a program twice:

1. The first pass assigns line numbers and creates a symbol table to record the line numbers referenced by labels.
2. The second pass "fills in the blanks" by replacing labels with actual line numbers.

Programming in assembly language vs. a high-level language

Programming in assembly language is a slower process than programming in a high-level language (C++, Java, etc.). However, the code it produces is typically more efficient (i.e., runs faster and/or takes up less memory).

So, there is a tradeoff between programmer time and code efficiency.

Instruction Number		Instruction	Meaning
Bin	Hex		
0001	1	Load X	Load the contents of address X into AC.
0010	2	Store X	Store the contents of AC at address X.
0011	3	Add X	Add the contents of address X to AC and store the result in AC.
0100	4	Subt X	Subtract the contents of address X from AC and store the result in AC.
0101	5	Input	Input a value from the keyboard into AC.
0110	6	Output	Output the value in AC to the display.
0111	7	Halt	Terminate the program.
1000	8	Skipcond	Skip the next instruction on condition.
1001	9	Jump X	Load the value of X into PC.

Instruction Number (hex)	Instruction	Meaning
0	JnS *X*	Store the PC at address X and jump to X + 1.
A	Clear	Put all zeros in AC.
B	AddI X	Add indirect: Go to address X. Use the value at X as the actual address of the data operand to add to AC.
C	JumpI X	Jump indirect: Go to address X. Use the value at X as the actual address of the location to jump to.
D	LoadI X	Load indirect: Go to address X. Use the value at X as the actual address of the operand to load into the AC.
E	StoreI X	Store indirect: Go to address X. Use the value at X as the destination address for storing the value in the accumulator.