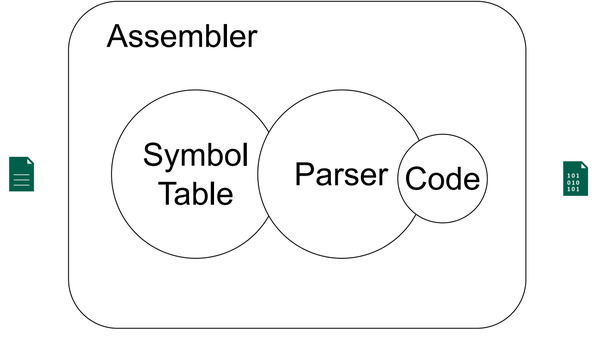
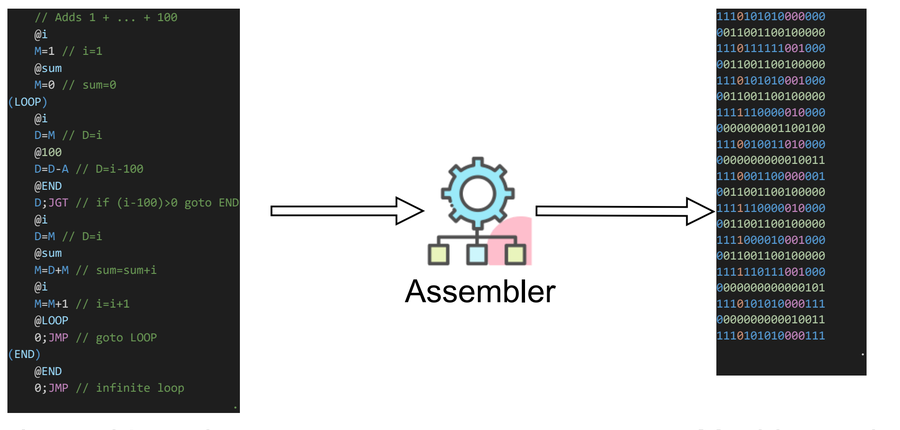




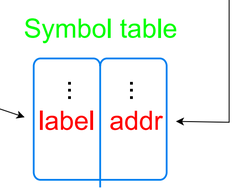
Pre-pass: **Read source and Clean-up codes**

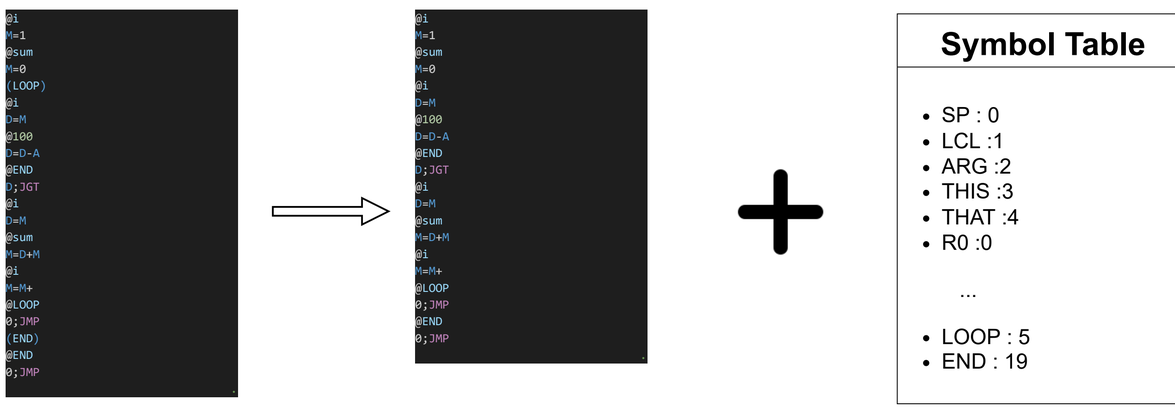
- Read the ASM code form file.
- Remove comments (// and all stuff after it until a EOL '\0')
- Remove Spaces
- Save the cleaned-up code into a queue (FIFO buffer).
- Initialize symbol table (e.g., pre-allocate R0-R15).

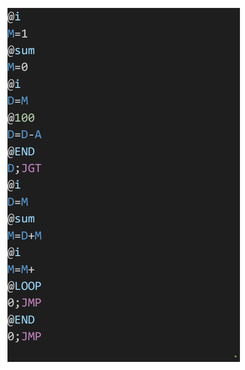
Label	*RAM address*	*(hexa)*
SP	0	0x0000
LCL	1	0x0001
ARG	2	0x0002
THIS	3	0x0003
THAT	4	0x0004
R0–R15	0-15	0x0000-f
SCREEN	16384	0x4000
KBD	24576	0x6000

First-pass: **Build symbol Table**

- Count lines and find labels "(Xxx)".
- The label has the same ROM address as the instruction next (below) to it.
- Xxx can not begin with a number.
- Add the XXX label and its corresponding ROM address to the symbol table.
- Erase the (Xxx) line.
- Save the processed code into a queue.

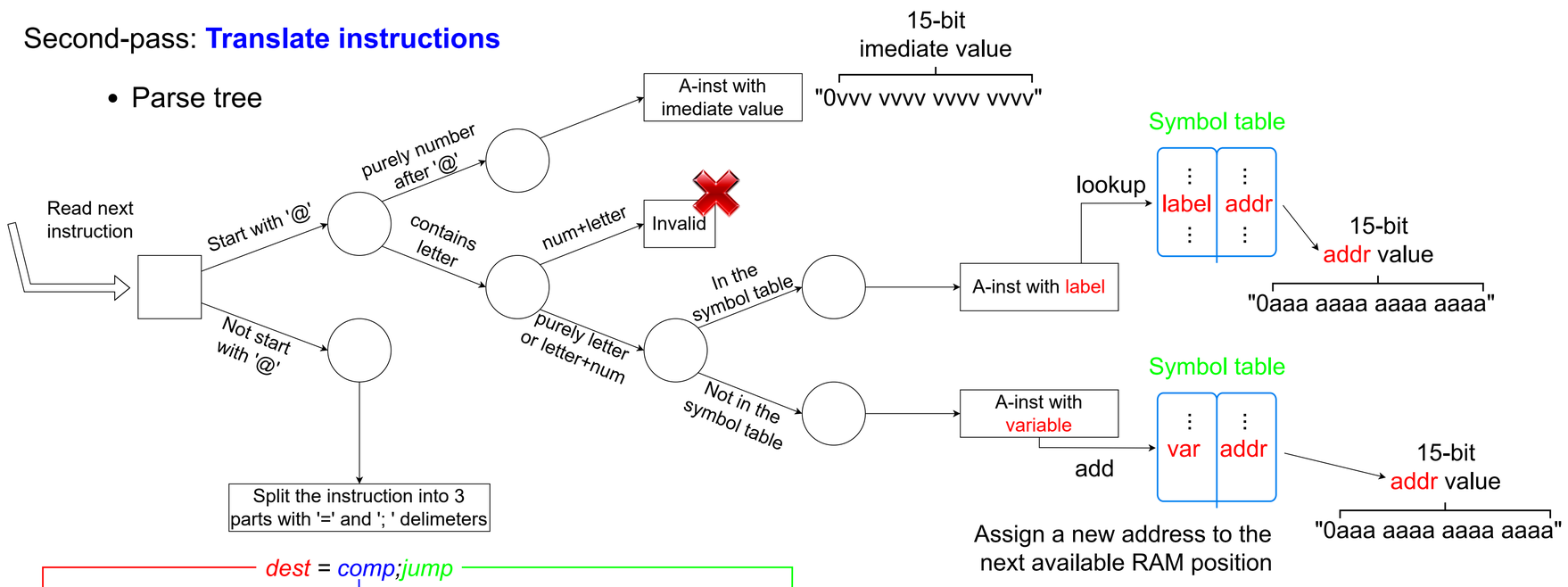


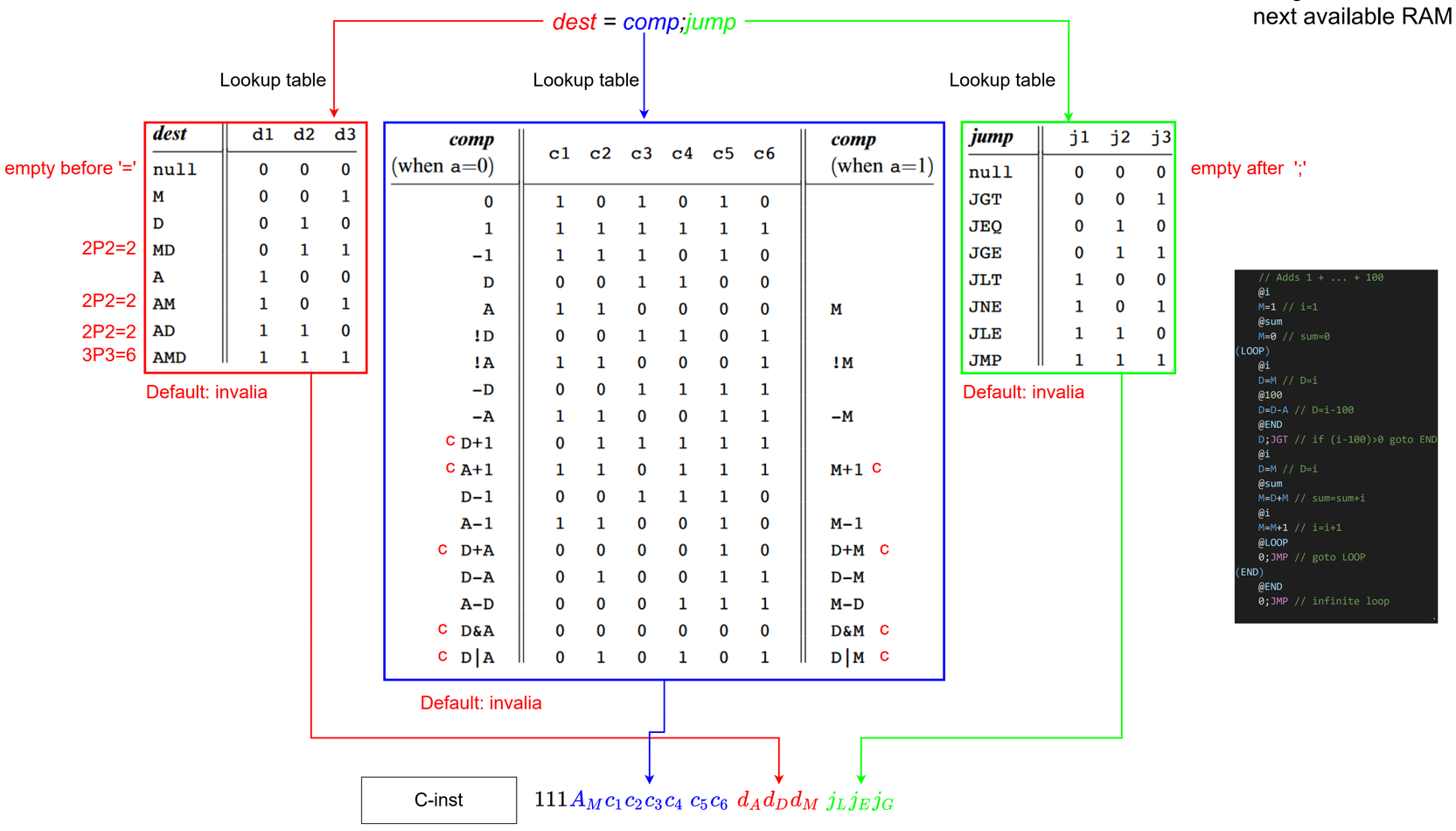




Second-pass: **Translate instructions**

- Parse tree





dest	d1	d2	d3
null	0	0	0
M	0	0	1
D	0	1	0
MD	0	1	1
A	1	0	0
AM	1	0	1
AD	1	1	0
AMD	1	1	1

comp (when a=0)	c1	c2	c3	c4	c5	c6	*comp* (when a=1)
0	1	0	1	0	1	0	
1	1	1	1	1	1	1	
-1	1	1	1	0	1	0	
D	0	0	1	1	0	0	
A	1	1	0	0	0	0	M
!D	0	0	1	1	0	1	
!A	1	1	0	0	0	1	!M
-D	0	0	1	1	1	1	
-A	1	1	0	0	1	1	-M
c D+1	0	1	1	1	1	1	
c A+1	1	1	0	1	1	1	M+1 c
D-1	0	0	1	1	1	0	
A-1	1	1	0	0	1	0	M-1
c D+A	0	0	0	0	1	0	D+M c
D-A	0	1	0	0	1	1	D-M
A-D	0	0	0	1	1	1	M-D
c D&A	0	0	0	0	0	0	D&M c
c D|A	0	1	0	1	0	1	D|M c

jump	j1	j2	j3
null	0	0	0
JGT	0	0	1
JEQ	0	1	0
JGE	0	1	1
JLT	1	0	0
JNE	1	0	1
JLE	1	1	0
JMP	1	1	1

Implementation: **C++**

- I know C++.
- C++ is Object-oriented
- C++ supports STL e.g., map and queue.