Assemblers

How Do Assemblers Work?

Just like we do (for this purpose)!

**Step 1:** Figure out the instruction sequence.
(Humans do this part.)

**Step 2:** Map instructions and data to memory addresses.
First pass of assembler: find where labels fall in memory.

**Step 3:** Calculate and fill in relative offsets.
Second pass of assembler: write bits.

The Assembler Counts to Find Label Values (Addresses)

The first pass produces a symbol table.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>HFLOOP</td>
<td>x3004</td>
</tr>
<tr>
<td>COUNTLOOP</td>
<td>x300C</td>
</tr>
<tr>
<td>NON_ALPHA</td>
<td>x3010</td>
</tr>
<tr>
<td>AT_LEAST_A</td>
<td>x3014</td>
</tr>
<tr>
<td>ALPHA</td>
<td>x3016</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>HIST</td>
<td>x3028</td>
</tr>
<tr>
<td>STRING</td>
<td>x3043</td>
</tr>
</tbody>
</table>

This table is generated for the assembly version of the letter frequency program.
The Assembler Uses the Symbol Table to Calculate Offsets

In the second pass, we count again!

Let's count!

What is HIST?
Look it up!

HIST is x3028.
What's PC?
PC is x3001.
So the offset is x27.

Find the Symbol to Calculate the LEA Offset

Find “HIST” in the symbol table.

HIST loop address is x3000.

HFLOOP x3004
COUNTLOOP x300C
NON_ALPHA x3010
AT_LEAST_A x3014
ALPHA x3016
...
...
HIST x3028
STRING x3043

What Can Go Wrong?

What happens if the assembly file has a mistake?

The assembler finds the bug.
Wishful thinking!
Computers are dumb.
The assembler says “ERROR!” ...
... and waits for the human (you!) to fix the bug.
What Can Happen in the First Pass?

What kinds of errors can the assembler detect in the first pass?

Things like **bad mnemonics**...

- `MULT R1,R2,R3`

and **bad operands**...

- `ADD R42,R0,#-3`
- `ADD R1,R2,#1000`

What about labels?

Is this code ok?

```
BRp NO_LABEL_YET
ADD R1,R2,R3
```

`NO_LABEL_YET ; here it is!`

A symbol in the file may not be found in the symbol table in the first pass, but that's ok.

What Can Happen in the First Pass?

What about this code...

```
BRp NO_LABEL_YET
ADD R1,R2,R3
```

`NO_LABEL_YET ; here it is!`

```
ADD R2,R1,R5
```

`NOLABEL_YET ; here, too!`

The symbol **cannot be added twice**!

If a symbol is already in the table, the **label is multiply-defined** (first pass).

What Can Happen in the Second Pass?

What kinds of errors can the assembler find in the second pass?

We saw one already...
Find the Symbol to Calculate the An Offset

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>HFLOOP</td>
<td>x3004</td>
</tr>
<tr>
<td>COUNTLOOP</td>
<td>x300C</td>
</tr>
<tr>
<td>NON_ALPHA</td>
<td>x3010</td>
</tr>
<tr>
<td>AT_LEAST_A</td>
<td>x3014</td>
</tr>
<tr>
<td>ALPHA</td>
<td>x3016</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>HISTOGRAM</td>
<td>x3028</td>
</tr>
<tr>
<td>STRING</td>
<td>x3043</td>
</tr>
</tbody>
</table>

Find “HIST” in the symbol table.

Oops!

What Can Happen in the Second Pass?

What kinds of errors can the assembler find in the second pass?
We saw one already...

Label not defined

What else might go wrong in the second pass?

What’s Wrong with This Code?

LEA R0,MY_SPACE
BRnzp STOP
MY_SPACE .BLKW x4200
STOP HALT

Data interleaved with code!
Bad style, but not an error.
What’s the offset for BRnzp? More than 9 bits...
(Error: Address/offset out of range.)

Errors Found by the LC-3 Assembler

Found in the first pass
- bad opcode mnemonic
- bad operand (of any kind, such as the wrong number, wrong type, or out of range)
- multiply-defined label

Found in the second pass
- undefined label
- target address too far away