**pBINAC Assembler**

The original BINAC had no assembler.

EDSAC had a thing called “Initial Orders” which is a sort of quasi assembler that used the letters but did not have labels.

Most early assemblers (e.g. SOAP and GOLUX) were positionally based (using tape and card) rather than syntactically based ; the main issue with this is that comments are separated from code positionally, which doesn’t work well in ASCII files with variable size TABs. This syntax is a synthesis of early assemblers and a later comment syntax.

Line Format

[<identifier>] [<command>] [<address>] [<additive> <additive> ….] \ comment

* Comments are delimited by the \
* Case is ignored.
* Identifiers are present if there is a non space character in the first character
* Identifiers are an alphabetic character followed by any number of alphanumerics.
* Commands do not have to be alphanumeric
* Old and New mnemonics are supported
* Everything else is separated by one or more space/tab characters
* Addresses and Additives are either constants or identifiers preceded by + or –
* The ‘result’ is the sum of the command, address and any additives modulo 777 octal
* Constants can be 0000 (octal) $FFFF (hexadecimal) #nnnn (decimal)
* Constants can be 0.nnnnnn and n.nnnnne-xxx (floating point) which are scaled by 2^30.
* Constants on their own (no command) occupy a whole word.
* The assembler automatically inserts a 025 command (SKIP) if required (i.e. an identifier is present on the second half of an instruction, or this instruction is a data word)

Pseudo Operations

ORG The address/additive sums set the current write address (must be defined on pass 1)

= or EQU Set the identifier to the value of the address/additive sums

BSS Allocate a number of words initialised to zero, length the value of address/additive

Extensions

1. From the start, the assembler should cope with the 15-bit code format (address A9 in bits 15 and 30 respectively)