ELLIOTT 900 SERIES SIMULATOR

CITY & GUILDS MNEMONIC CODE

City and Guilds Mnemonic Code is a machine independent assembly code created by the City and Guilds Institute to enable practical programming questions to be included in the examination for their basic and advanced Certificates for Computer Personnel (courses 319 and 320 respectively). The courses were introduced in 1964 and revised in 1968 when course 319 was renamed "Certificate in Computer Programming and Information Processing".

MNEMONIC CODE SYSTEM

Mnemonic Code comprises assembler directives, machine orders (instructions) and numbers. Data are held internally in floating point form.

The code assumes a computer with 1,000 words of store. Each word can contain an order, a (floating point) number or a character for input/output. The architecture requires integers to be held with at least 7 digits precision.

The order code is of the single address form. The first 10 locations of store are reserved for registers. Location 0 always contains the value 0; it cannot be updated. Location 1 is the accumulator (A). Location 4 is used to hold the return link when entering a subroutine. All the registers can be referenced as index registers.

In addition to the index registers there is an addition control register (C) that contains the address of the next instruction to be executed. Unless updated by a jump instruction C is automatically incremented after each instruction is executed.

An order comprises four fields:

F F	Q	n n n	m

1. Order number F
2. Address nnn
3. Modifier m
4. Trace Q

The order number specifies the function to be performed as listed below.

ELLIOTT 900 SERIES SIMULATOR

The address is a natural number in the range 0-999.

The modifier is a natural number in the range 0-9 and, for orders that support modification, the effective address is computed by adding the content of the address field to the content of the nominated index register (i.e., store location). Thus, using 0 as the modifier yields the address field unmodified. Note that, since the contents of store comprise floating-point numbers, the value of the index register has to be rounded before it is added to the address field.

Instructions and numbers to be assembled are written one to a line. The fields of an instruction can be separated by spaces or tabs for improved layout. If an instruction is terminated by the letter Q a diagnostic message will be output at run-time when tracing is enabled.

Mnemonic	Operation	Remarks
LDA n, m	A:=(n+(m))	Load operand into cleared accumulator
ADD n, m	A:=(A)+(n+(m))	Add operand
SUB n, m	A:=(A)-(n+(m))	Subtract operand
MLT n, m	A:=(A)*(n+(m))	Multiply by operand
DIV n, m	A:=(A)/(n+(m))	Divide by operand
LDAN n	A:=n	Load integer
ADDN n	A:=(A)+n	Add integer
SUBN n	A:=(A)-n	Subtract integer
MLTN n	A:=(A)*n	Multiply by integer
DIVN n	A:=(A)/n	Divide by integer
STA n,m	n+(m):=(A)	Store (A) without clearing accumulator
JUN n, m	C:=n+(m)	Jump unconditionally
JGR n, m	If (A)>0 C:=n+(m)	Jump if A>0
JEQ n,m	If (A)=0 C:=n+(m)	Jump if A=0
JSR n, m	4:=Link; C:=n+(m)	Set link and jump; link is the address of the instruction following JSR
JST n, m	Wait; C:=n+(m)	Wait; jump when start button operated
SQT n, m	A:=sqrt(A)	If (A)<0, jump to n+(m)
EXP n, m	A:=exp(A)	If (A) too large jump to n+(m)
LGN n, m	A:=ln(A)	If (A)<0, jump to n+(m)
SIN	A:=sin(A)	(A) in radians
COS	A:=cos(A)	(A) in radians

ARC	A:=arctan(A)	(A) in radians
ENT	A:=entier(A)	Integral part of (A) to A
RCT n, m	character to n+(m)	Read single character from tape
PCT n, m	(n+(m)) to tape	Punch single character to tape
RNT n, m	number to A	Read number from tape; jump to n+(m) if error in the number
PNT n, m	(A) to tape	Print signed number in A to tape with n integral and m fractional digits
PNL		Punch the characters for new line
RCC n, m	characters to n+(m)	Read characters from card
PCC n, m	(n+(m)) to card	Punch characters to card
RNC n, m	number to A	Read number from card; jump to n+(m) if error in the number
PNC n, m	(A) to tape	Print signed number in A on to card with n integral and m fractional digits

RCC, PCC, RNC and PNC are recognized but give an error at run-time.

Numbers are written as integers or floating point numbers introduced by a + or – sign. All numbers are stored in floating point form. A subscript 10 (code 63 – which corresponds to ? in ASCII) can be used to introduce an exponent.

There are 4 directives to the assembler:

(TITLE)	Reads the next line as the program title and copies to the output.
(STORE n)	Stores the following program from location n onwards.
(WAIT)	Pauses input, awaiting a further program tape to be input.
(EXECUTE n)	Marks program complete and sets location n as the location at which execution should commence.

There is no parity checking on input to the translator or the interpreter, nor is halt code (20) recognized.

There is no check made on the length of input lines to the translator or interpreter. Lines longer than 120 characters can overwrite the system and give rise to unpredictable behaviour.

The interpreter detects when attempt is made to access a location containing a number as an instruction and vice versa: a run-time error is reported. Attempting to read from uninitialized store can sometimes provoke such errors depending on what a previous program has left in the 903 store.

Numbers read as input using RNC may omit the leading "+" and several numbers can be input on the same line provided they are separated by at least two spaces.

If the argument to the SQT, EXP or LGN orders is out of range, the interpreter reports error 18 (see below). If execution is resumed, the interpreter takes the exception branch as specified in the order code table above.

Entry Points

8 – Read and translate program.
9 – Run program.

Note: a program cannot be rerun by re-entering at 9 after execution.

Level 1 Interrupt: Turn tracing on. (Can be used at any point after translation, prior to or during program execution).

Level 2 Interrupt: Continue translation after WAIT directive. Should only be used when reading a previous tape has stopped at a WAIT directive.

Level 3 Interrupt: Continue after JST. Should only be used when execution has halted at a JST instruction.

Error Messages

Error messages are punched as

```
ERR n m
```

Where n is the error number and m is the current placing address for assembly errors or the address of the current instruction for run-time errors.

```
ERR  1 - invalid character
ERR  2 - syntax error
ERR  3 - numeric overflow reading number
ERR  4 - current placing address out of range 10-999
ERR  5 - unknown directive
ERR  6 - number not expected in directive
ERR  7 - modifier field out of range 0-9
ERR  8 - invalid function mnemonic
ERR  9 - address field out of range 0-999
ERR 10 - EXECUTE address out of bounds 10-999
ERR 11 - attempt to execute a number
ERR 12 - index register contains an instruction
ERR 13 - modified address out of bounds 0-999
ERR 14 - function code not implemented.
ERR 15 - attempt to load instruction as data
ERR 16 - in PCT, character code out of range
ERR 17 - No program to run
ERR 18 - QF error, e.g., real overflow
```

EDITOR

The C&G system also includes an editor, somewhat similar to 903 EDIT.

Entry Points

```
 48 - Read steering tape.
 49 - Read first input tape and punch output tape.
 50 - Read subsequent input tape / resume after halt code.
 51 - Read input tape and print and punch output.
```

Note entry 51 assumes a 903 telecode printing device driven by 15 4352 instruction which is not supported by the Simulator.

Commands

The steering tape contains editing commands, one per line terminated by halt code. The commands are:

```
FL string - copy to line starting with string.
DL string - delete to line start with string.
    If string ends @ line must be exactly the characters
    preceding @.
    FL/DL stop after matched string.
```

FC string - copy to characters in string successively.
DC string - delete to characters in string successively.
    FC/DC stop after last matching character.
IS string - insert string at current position.
CS string - change string at current position.
    Replace characters from the current position by the string.
IB string - insert block.
    The string and all following lines until terminated by a
    line starting ` (grave, code 64, @ in ASCII) are copied to
the output.
RE - read to end of tape (i.e., halt code).

Error Messages

ERROR 1 - illegal character (e.g., parity error).
ERROR 2 - Unrecognized command.
ERROR 3 - characters found after @ in FL/DL command string.
ERROR 4 - Halt code found in Insert Block.

Demonstration Programs.

The CANDG directory contains two simple demonstration programs.

DEMO1.DAT: Prints out π.

DEMO2.DAT: Reads in data about steel samples and prints a report for each sample showing composition and grade.

DEMO3.DAT: Prints a currency conversion table. The input consists of five numbers: the number of rows to be printed, the starting value in pounds, the increment between rows, the pound to dollar exchange rate and pound to Deutsche Mark exchange rate.

DEMO4.DAT: Calculates day of week on which a date falls.

DEMO5.DAT: Solves quadratic equations.

DEMO6.DAT: Worked examples taken from the book "Electronic Data Processing", by G. Emery, Pitman, 1968, Appendix V.

SYSTEM FILES

CANDG.BIN - "903 CITY & GUILDS MNEMONIC CODE 319 ISS 1". The Mnemonic Code system. This tape was found in a collection held by Mr Christopher Pugh-Jones, with some errors, corrected by Terry Froggatt.