THE BROOKER-MORRIS COMPILER-COMPILER ON THE G 21

1. THE BROOKER-MORRIS COMPILER-COMPILER ON THE G21

1.1. Introduction

The Brooker-Morris Compiler-Compiler has been implemented in part on the G21. This first chapter will give details of how to use the C-C on the G21, the limitations of the system, peculiarities of the G21 system, etc. It should be pointed out that this implementation was carried out purely as a means of educating the author in the detailed coding of the C-C. Consequently, no claims for efficiency or usefulness are made. In fact, unless the system is run as a 64k program, it is only useful for student examples and familiarizing the user with the system. This, in part, is due to the crude method of implementation.

Following chapters will describe the bootstrapping process used to implement the system from the Atlas version and details of how the fundamental Master Phrases are implemented.

1.2. How to Run a C-C Program

At the moment the Compiler Compiler is situated on a Logical File Type. It requires approximately 60 blocks and a user may either use the standard version or obtain a copy of the system on his own logical file. The current custodian of the system will know the position of the C-C. At the moment it resides on FILE 46/1 of User CS07FH04, but this will probably be changed.

Details of the facilities available in the C-C are given in the paper, "The Compiler Compiler" in the Annual Review of Automatic Programming Vol.III. Knowledge of this paper is assumed and remarks will be made with respect to it.

An example of a simple C-C run on the G21 would be:

```
$$ 330 AND CS07FH04  HOPGOOD
AN FILE 46/1(12);  FILE 48/1(13); TEXT;
       PHRASE [XYZ] = AB, CD, EF
       DEFINE COMPILER 13
AN RUN, (12), TAPE;
$$
```

This illustrates the method of bringing down the Compiler-Compiler and also the Master Phrase for redefining an augmented version of the C-C. The 'DEFINE COMPILER' Master Phrase has just one argument which is the number of the logical file that the system is to be dumped on. In the example the phrase XYZ has been added to the C-C.

In order to run the C-C as a 64k program it is necessary to patch two instructions in the C-C. This can be done by UPDATE and the above program would then be:

```
$$ 3 30 AND  CS07FH04  HOPGOOD
AN FILE 46/1(12); FILE 48/1(13);  TEXT;
       LXS  12;
       PATCH  12021, 120000  .160000 ;
       TRA  12000;
       PHRASE [XYZ] = AB, CD, EF
       DEFINE COMPILER 13
AN RUN, UPDAT, TAPE;
$$
```

Note that the C-C defined will now be a 64k version and to convert back to a 32k version requires:

```
PATCH   12021, 60000 .67776;
```

A 64k program cannot, of course, be run from the teletypes in the normal manner.

1.3. Storage Layout

The present storage layout is:

Input Buffer	11300
Masks and useful constants	11540
B variables	11600
Index	12000
G21 op-code skeletons	14000
Record Store	15000 to 52000 (approx)
Stack	60000
Chain	67776 (downwards)

The positions of stack and chain are moved in the 64k version so that about 16k of chain store is available and the record store can increase until it hits the monitor. If the record store does become this large, then it would be necessary to increase the LFT size from 60 to 70 blocks.

At present an interactive scope version is under construction which also uses the space below 11300.

1.4. G21 Characteristics of Bootstrap

The Bootstrap part of the C-C defined in Chapter 2 is machine dependent, and this section will describe G21 features of the bootstrap. This part was written in SPITE using a set of Macros which simulated the Atlas Machine Orders in the original version. It is hoped to be able to use the same macros in the 360-67 version.

1.4.1. Down Sequence

The routine entry sequence on Atlas used the B variables B91, B92, B93, B94 and B99. The G21 version does not destroy these variables. B99 is still used by the END sequence.

1.4.2. Input Routine 238

This routine takes the input and produces an internal form which is equivalent to the G21 internal character codes with the meta-characters defined as follows:

eol	104
EOL	107
comma	105
[	106
space	101

The character 10 is used in place of ? and the operators $\leq$, $\geq$ are replaced by < =, > =. The equivalence sign ≡ is written as (=).

Routines added to the normal C-C routines are:

241	Prints B-variables from B40 to B127 (no parameters)
283	Prints store from B125 to B126

Two additional Master Phrases added to turn listing of the input on and off are:

LIST
DO NOT LIST

1.5. G21 Plant Routine

The Atlas PLANT basic statement has been replaced by the statement:

```
LOAD [MNEMONIC][COMMA][ABN][COMMA][ADDRESS][REG?] INTO [B][SEP]
```

where

```
[ADDRESS] = [WORD], • [BFJX][N] + [WORD], • [BFJX][N]
[BFJX]    = B,F,J,X
[REG]     = [COMMA][ABN]
[MNEMONIC]= G21 MNEMONIC
```

The field [ABN] is a node field which is compulsory. The phrase [REG] defining the register part can be ignored. [B] defines the location where the instruction is to be placed and is automatically incremented.

In order to understand the address part of the field it is necessary to first define the G21 fields for the various fields in the data described in Chapter 2.

Field	Bits
A	0 - 15
F	16 - 17
J	18 - 19
S	20 - 21
Z	22 - 31

As the higher fields appear in the operation part of an instruction it is necessary to provide constant locations containing the relevant masks for these bits. The masks for the F and J fields are stored in addresses F_1, F_2, F_3, J_2, J_4, J_6. Also, it may be required to put an index position or B variable address in an instruction. The dot notation is used to denote that the address itself is required rather than the contents of the address.

For example, if B32 = 6 then

```
    LOAD   CAL, 2, B32  INTO  B1
```

will load the instruction CAL 2 6 while

```
    LOAD   CAL, 2, •B32 INTO B1
```

will load the instruction CAL 2 /11640

The +[WORD] in the second [ADDRESS] alternative will add the value of word to the address defined in the first part, •[BFJX][N].

1.6. The Phrase [WORD]

This, to a large extent, was Atlas oriented in the definition of constants. This has been replaced by the following definition:

[WORD] = [ADDR], ([ADDR]), -[N], [N][JF?], [JF]

[JF] $= F_1, F_2, F_3, J_2, J_4, J_6$

1.7. Limitations of G21 Version

The G21 version of the C-C is approximately equivalent to the Atlas version with the Auxiliary Statements undefined. On the Atlas version all simple forms of the Basic Statements were compiled rather than interpreted. On the G21 all Basic Statements are interpreted. No 'COMPILER' versions of the Basic Statement routines have been added. This was done partly because of storage problems and also because this is the point where G21 dependent routines would have to be added. As the first aim is to have the C-C implemented on the 360-67 this work to a large extent would be wasted.

The Auxiliary Statements have been omitted because, without the COMPILE versions of the Basic Statements, it is impossible to get the routines compiled in the 32k G21 store. These are all written in terms of CC orders and very little work is necessary to add these if a 360-67 version becomes available.