ICL 1900 SERIES
Preservation Project

GEORGE 4

Running paged programs under G3
– It is ‘possible’.

W. J. Gallagher

GEORGE 4 Investigations

An investigation into the world of GEORGE4 programs.

Rationale

Partially unbridled curiosity, spurred on by the discovery one of the main program development tools used to create paged programs. – XPCH, the paged consolidator, plus several of the paged compilers for ALGOL and FORTRAN.

We currently do not have a copy of GEORGE4, neither in binary format nor a complete set of sources to build it from source – a number of essential source segments are missing, and it seems reasonable to assume that the 8.66 and 8.67 edits for the use of GEORGE3 under 2900 emulations have prejudiced the available code's ability to even compile as a valid GEORGE4, even if the missing segments became available.

The ability of the very interesting PLASYD compiler to generate semicompiled output to create paged images triggered my interest in finding out what could be done to run these compilers and consolidators.

First, try to load the consolidator,

```
10.35.25_ lo :lib.program xpch
ERROR  IN LO::LIB.PROGRAM XPCH CONTAINS A SPARSE PROGRAM WHICH CAN ONLY RUN UNDER GEORGE 4
10.39.08_
```

OK, an expected message, produced by a test in the LOADPROG chapter. Some examination of both the layout of paged programs on disc and the LOAD related chapters we do have from GEORGE showed that GEORGE3 simply detects the sparse program bit in the request slip. OK, let's patch that out; the following MEND was added to the end of a full GEORGE3 compilation and the AWJGG4 universal defined as 1 at the head of the source:-

```
#SKI AWJGG4
(
#CON FIDDLES TO ALLOW SOME G4 FACILITIES
#MEN LOADPROG
#TRA 32
 NULL        [ REMOVE CHECK FOR SPARSE IMAGES
#END
)
```

This simple, and admittedly 'dirty' patch allowed the image to load successfully! - That was with the addition of an extra parameter to explicitly request > about 90KW of core. - I also had to set OBJECTQUOTA and COREOBJECT to rather high values (200KW), but the program loaded. Running was, of course, more complicated.

Please note that all of my odd experiments are implemented optionally by wrapping the changes with

#SKI AWJGG4

(

…

)

This requires that the GEORGE restore have the universal AWJGG4 set to a non-zero value at the head of the restore deck.

Running #XPCH under G3

Having managed to get the program loaded into core, the next step is to run it. Unfortunately, we don't have a copy of the relevant manual[1], but using the documentation for #XPCK in TP4431 (Compiling Systems) it was possible to use it to consolidate normal 1900 images. However, it was not possible to have #XPCH leave the image in core as it went illegal upon encountering a HIVE extracode (167 X=2 – See G3/4 Ops Management for a specification).

From the available steering files in the GEORGE sources, we know that this extracode was implemented in a chapter called HIVE and that the code in the OPCA chapter needed to be changed to transfer control to it when it was encountered[2]. This time, rather than a bare #MEND, the extent of the changes made it sensible to create a MENDITed version of the OPCA867 chapter and to include that modified chapter source as a user MENDIT. The relevant code in OPCA867 at XTPINS was originally

```
4^8W    XTPINS
574B          ACROSS   TPINS,1            [167  ORDER
```

Using the AWJGG4 universal to allow this code to be conditionally activated, led to this code which was found in a copy of the OPCA for Mk8.65 GEORGE. It quite simply tests for X=2 and transfers control across to the new HIVE chapter.

```
4^8W    XTPINS
0000    #SKI AWJGG4
0000    (
0000          LDX  4  EVENT5(2)           [ CODE RE-INSTATED FROM 865 OPCA
0000          SLC  4  3
0000          ANDN 4  7                   [ EXTRACT X
0000          SBN  4  2
0000          BNZ  4  XHIVE               [ J IF NOT HIVE XC
0000          ACROSS  HIVE,1
0000    XHIVE
0000    )
574B          ACROSS   TPINS,1            [167  ORDER
```

Now, we must code up a whole new HIVE chapter. The G3/4 Ops Management gives a clear enough definition of what it needs to do:

1. The issuing image needs to be R trusted, error if not.
2. The issuing member must not be a TP, error if not.
3. The two-word area addressed by the instruction is tested for accessibility, error if not.
4. The alignment of the area to be moved down is checked to lie on a quire (64K) boundary, error if not.
5. Finally, after all checks have been completed, the MOVLONG macro is invoked to move the image down. – This is purely experimental and performs multiple MOVE orders in a loop. As the amount of data to be moved is frequently extremely large, this macro should be replaced by code that allows GEORGE to co-ordinate occasionally so that other images can run.
6. The program transfers across to a new entry point in the GIVE chapter so that the new image size values can be returned in the specified accumulators.

[1] TP4298 'GEORGE 3 & 4 Compiling Systems'

[2] Luckily a copy of OPCA865 is available, so we know exactly how to retrofit HIVE dispatch there.

```
IN :SOURCE4,HIVE869(2001),T////
0000    #SEG HIVE
0000    #OPT K0HIVE=0
0000    #OPT K6HIVE=100
0000    #LIS 4
0000          8HHIVE
0000          SEGENTRY K1HIVE,XENT1
0000    #
0000    #     IMPLEMENT THE 167 X-2 EXTRACODE TO PERMIT G3
0000    #     TO RUN #XPCH WITHOUT INCIDENT
0000    #
0000    #     N(M) ADDRESSES A TWO WORDAREA WHICH DEFINES THE REGION
0000    #     OF THE CURRENT IMAGE THAT IS TO BE RETAINED.
0000    #
0000    #     WORD  CONTENTS
0000    #     ------------------------------------------------------------
0000    #     0     VIRTUAL ADDRESS OF START OF REGION (QUIRE ALIGNED)
0000    #     1     BIT 0 -> SPARSE
0000    #           LENGTH IS WORDS (PAGES)
0000    #
0000
0000    #
0000    #     CAVEAT
0000    #     ------
0000    #
0000    #     THIS IS MY CODE, NOT ICL'S.  G3 NEVER SUPPORTED THE 167 X=2
0000    #     EXTRACODE AS IT COULD ONLY BE INEFFICIENT AS IT REALLY
0000    #     REQUIRED A LOT OF DATA TO BE MOVED. GEORGE4 ACHIEVED THIS
0000    #     WITH MOVES OF ABOUT 64 WORDS, AS ALL THAT WAS REQUIRED
0000    #     WAS THAT THE SEGMENT TABLE ENTRIES FOR THE RETAINED
0000    #     PORTION BE MOVED DOWN, AND ANY PAGES HELD FOR THE DELETED
0000    #     PORTION OF THE IMAGE FREED.
0000    #
0000    X22AM          #17777777
0000    X64KMASK       #00177777
0000
0000    XILL
0000          UNPLUG
0000          ILLEGAL  ILLINS
0000    XENT1
0000          TRACE 2,HIVEENT
0000          TRUSTED  FX2,XILL,R          [ NOT ALLOWED UNLESS R TRUSTED
0000          TESTTP   2,XILL              [ ALSO NOT IF A TP
0000
0000          LDN   3  2                   [ SIZE OF AREA TO BE TESTED
0000    XST   CHECKB   EVENT2(2),3,XILL,,W,XST [ VALIDATE N
0000          DATUMA   3
0000          MACCS    ,3
0000          LDX   3  0                   [ X3 POINTS TO PROGRAM
0000          ADX   3  EVENT2(2)           [ X3 IS ADDRESS OF PARAMETERS
0000          LDX   4  0(3)
0000          LDX   6  1(3)
0000    #SKI  K6HIVE>99-99
0000    (
0000          TRACE 4,HIVE X
0000          TRACE 6,HIVE X*
0000          TRACE 0,HIVEDATM
0000    )
0000    #
0000    #     TODO: VALIDATE PARAMETERS.
0000    #
0000          BNG   6  XILL                [ NO SPARSE SUPPORT (YET)
0000          STO   6  ACOMMUNE7(2)        [ SIZE TO PASS OVER TO GIVE
0000          LDX   7  X64KMASK(1)
0000          ANDX  7  4
0000          BNZ   7  XILL                [ IMAGE NOT 64K ALIGNED
0000
0000          ADX   4  0                   [ G ADDR OF NEW IMAGE X0
0000          LDX   5  0                   [ X4,X5 SET FOR MOVE
0000          MOVLONG 4,5,6
0000
0000    #SKI  K6HIVE>99-99
0000          TRACE 4,HIVEMOVD
0000    #
0000    #     USE THE IMAGE SIZE FIELD TO INVOKE GIVE /4 CODING
0000    #     SO THAT THE IMAGE SIZE BECOMES THAT OF THE LOADED IMAGE.
0000    #
0000          ACROSS   GIVE,9              [ RETURN TO PROGRAM VIA
0000                                       [  GIVE /4 CODING.
0000    #END
0000
^^^^    0000000000100000000
```

The coding was not present in GIVE to do this, so a new entry point was added at the end of GIVE to do the needful:

```
0000    #SKI  AWJGG4
0000    (
0000    #
0000    #     ENTRY FROM HIVE EXTRACODE (167 X=2) AS PER GEORGE4
0000    #
0000    #     THE IMAGE IS MOVED DOWN AND THE REQUESTED SIZE IN WORDS
0000    #     FROM N+1 IS IN ACOMMUNE7
0000    #
0000          SEGENTRY K9GIVE
0000          LDX   4  ACOMMUNE7(2)        [ GET NEW CORESIZE PARAM
0000          NGN   0  64                  [ CORE STEP SIZE (WG)
0000          CALL  6  XHIVEENT            [ X4 IS NEW SIZE,X1 ASCBT
0000    #SKI  K6GIVE>99-99
0000          TRACE 4  ,GIVEK9SZ
0000          BRN      XG4BG               [ J TO ACTUALLY SET SIZE
0000    )
```

The CALL to XHIVEENT required a local label to be placed into the code in GIVE /4 processing.

```
3TQS ...SIZE1
3TX= ...      NGN   0  256
3W3W    SZD   LDX   4  0(3)
0000    #SKI  AWJGG4
0000    XHIVEENT
3WHG          BNG   4  SZ                  [ERROR IF -VE
3X36          BZE   4  SZ                  [ERROR IF ZERO
3XGQ          SBN   4  1
3Y2B          SBX   4  0                   [V IS CLEARED BY CALL
3YG2          BVSR     SZ                  [ERROR IF TOO BIG
...
```

The above code allows #XPCH to be used to consolidate normal 1900 programs and to either leave the resulting image in core or to have it written out to *DA file in the normal manner. I have left the implementation of HIVE involving a sparse image for later as we cannot determine the location of the presumed active areas and avoid moving the gaps in the address space. Simply omitting the BNG 6 XILL will allow the code to work, but at the expense of moving several MW about. – Typically #XPCH builds the new image at 1MW (#04000000) and the HIVE extracode will try to move from that address up to COREOBJECT down to program virtual location 0! – This is fairly easy and inexpensive under G4 as the changes will all be the the page tables and no actual data will need to be moved.

Running the ALGOL and FORTRAN G4 compilers.

It was now possible to load and enter the G4 compilers[3], but I quickly encountered a group of extracodes that were used – GIVE with N(M)= 16 through 19. These extracodes allow a program to check and alter the permissions on pages in the program's image. Enter yet another new chapter, again using the original name GIVEPQ.

```
IN :SOURCE4,GIVEPQ869(2001),T////
0000    #OPT K0GIVEPQ=0
0000    #OPT K6GIVEPQ=100
0000    #SEG GIVEPQ
0000    #LIS 4
0000          8HGIVEPQ
0000          SEGENTRY K1GIVEPQ,XENT1
0000    #
0000    #     CAVEAT
0000    #     ------
0000    #
0000    #     WHILST THIS CHAPTER HAS THE SAME NAME AS THE GEORGE4 ONE
0000    #     THAT PROCESSES THE GIVE 16, 17, 18 AND 19 EXTRACODES, THIS
0000    #     IS NOT ICLS CODE, BUT MINE, PURELY TO INVESTIGATE RUNNING
0000    #     GEORGE4 BINARIES UNDER GEORGE3.
0000    #
0000
0000    #
0000    #     IMPLEMENT THE GIVE 16,17,18 AND 19 EXTRACODES IN A MANNER
0000    #     THAT ALLOWS DETECTION OF NEW REGIONS. INITIALLY I WILL
0000    #     SIMPLY TRACE THE RELEVANT PARAMETERS.
0000    #
0000
0000    #
0000    #     SUBROUTINES, USEFUL CODE
0000    #
0000    XILLXC
0000    XILLINS1
0000          UNPLUG
0000    XILLINS
0000          ILLEGAL  ILLINS
0000    XGIVE
0000          LDN   4  0
0000    STSLR
0000          STO   4  0(3)               [ SET SINGLE-LENGTH REPLY
0000          TRACE 4,GVPQSLR
0000          EVENTFIN
0000    XENT1
0000          LDX   7  EVENT5(2)          [ GET X ADDRESS
0000          SLC   7  3
0000          ANDN  7  7
0000          DATUMA   3                  [ X3 := OBJECT DATUM
0000          MACCS    ,3                 [ X0 IS ADDRESS OF ACCS
0000          LDX   3  0                  [ CAN USE X3 TO ACCESS IMAGE
0000          ADX   3  7                  [ X3 POINTS TO X,X* IN IMAGE
0000
0000          LDX   4  EVENT2(2)          [ N-OPERAND
0000          SBN   4  20
0000          BPZ   4  XGIVE              [ RETURN QUIETLY IF NOT 16..19
0000    #
0000    #     JUST OBTAIN PARAMETERS AND TRACE THEM
0000    #
0000          LDX   4  0(3)               [ GET X
0000          SBN   7  7
0000          BNZ   7  XNOT7
0000          SBN   3  8                  [ FIDDLE ADDRESS TO ACCESS X0
0000    XNOT7
0000          LDX   5  1(3)               [ GET X*
0000    #SKI  K6GIVEPQ>99-99
0000    (
0000          TRACE 4,GVPQREP1            [ LET'S SEE THE RESULTS
0000          TRACE 5,GVPQREP2
0000    )
0000          EVENTFIN                    [ RETURN TO PROGRAM
0000    K99GIVEPQ
0000    #END
0000
^^^^    0000000000100000000
////
```

This new chapter, again requires an edit to OPCA to transfer control to GIVEPQ when a GIVE extracode with N(M) in the range is encountered.

The code actually does nothing except (optionally) traces the parameters. This seems to be enough to allow both the ALGOL and FORTRAN G4 compilers to run.

[3] This is not 100% true, the compilers both HALT: WE. The compiler has detected that it is running under a GEORGE other than 4. A quick examination of the code in the compiler shows that it will continue ir simply resumed. Alternatively the compiler program could be patched.

In order to handle the *very* large address space assumed by all sparse programs, it was necessary to configure the G3 system with a *prodigious* amount of swap space, and to set several of the IPs to frankly unbelievable values:-

```
IP SIZEDEFAULT,2048000
IP COREOBJECT,2048000
IP OBJECTQUOTA,2048000
```

Observations

That it was possible to do the above on a GEORGE3 system without encountering serious problems showing that the architectural design of GEORGE 3 & 4 was rather well compartmentalized and that there were no *deep* dependencies. I/O to all devices, as implemented by executive, simply worked, even with I/O buffer addresses well above what could have been encountered on any unpaged machine. EWG3 and EWG4 allow I/O transfers to buffers anywhere in the 4MW address space possible with 22-bit addressing.

A radically different set of addressing assumptions[4], as output by the XPCH consolidator produced no problems except the expected performance hit of swapping enormous core images in and out.

The possibility of re-creating a true GEORGE4 remains however rather remote. We do have a source for the required EWG4 executive, and have also run *FLIT with a rather experimental paging unit instead of the normal D&L mechanism. The *big* problem is we do not have sources for the most important segments and chapters of a true GEORGE4 system. – The code that allocates core store, manipulates the page tables and performs the swap transfers on them being chief among the absentees. Maybe they will turn up in a garage or attic somewhere. – Please?

[4] See separate document ‘Sparse_Program_Layout.docx’ – Some extra record types were encountered in sparse *DA program files.