bcc	**title** INTERACTIVE MICROPROCESSOR SIMULATOR	**prefix/class-number.revision** PIG/M-13		
checked	**authors** Paul Heckel Paul Heckel	**approval date** 10/28/69	**revision date** —	
checked Bo Lewendal		**classification** Manual		
approved Mel		**distribution** Company Private	**pages** 16	

ABSTRACT and CONTENTS

IMS is a simulator which may be used to help debug programs for the microprocessors that will be used with the B.C.C. computers. The use of IMS requires a knowledge of DDT because IMS was designed to take advantage of its capabilities. The user may examine and change both instructions and registers via pops which may be executed by DDT's;U. The user may set breakpoints at various microprocessor locations, and when they are reached IMS prints out a differential dump of the microprocessor registers.

One of the commands to IMS allows the user to generate a symbolic file which lists the diode configuration. This may be used as either a dump of the simulator program or as a bit list. This command will also generate files which may be fed into a program to generate paper tape files for the drilling machine.

The program to generate paper tape to be punched at STANFORD is also described.

TABLE OF CONTENTS

Compiling and Loading Files:

Programs may be written in MICRO. (SEE MICRO/M-8) Having been compiled with MICRO, the object file produced must be run through FNARP to produce a Binary File.

Programs can be loaded into the dump file (PIRTLE)DIMS. QSPL and machine language programs may be used with the micro program to simulate branch conditions and special conditions; but they must be loaded first with a ';T' command. When all of these files are loaded, the binary Micro programs are loaded with 'Ø;T'. When all of the binary files have been loaded an

```
INIT;U
```

must be executed. The location of any instruction which uses more than 32 bits will be printed out at this time. A dump file may now be made as the program is ready to run.

Running a Program:

The Nth machine instruction is in location SLØ+N. This location retains the label supplied to MICRO: thus the user may transfer to location FOO by saying

FOO;G

He may place a breakpoint at location MUMBLE by saying

MUMBLE!

The user may look at location BRLØ to BRL6 to find out the addresses of the last 7 branches.

Tracing a Program

Before a program executes an instruction with a breakpoint, or wherever TRACESW is set to ON, IMS prints out the names of the first XTR register and core location that have changed, their old values, and their new values. XTR, which is initially set to 8, may be changed by the user. No more than 16 changed memory locations will be printed. If a trace output is incomplete, an up arrow (↑) will be printed to indicate this.

Tracing may be turned off by setting TRACESW to OFF. If the user wants a long trace on a line printer, he should say

```
                  FILOUT OFNO;U
```

and supply a file name and proceed as before. The output will go to the specified file.

Register Modification

The following registers may be referenced symbolically: M,Q,Z,RØ,R1,R2,R3,R4,R5,R6, SKØ. The Nth scratch pad is at SKØ+N.

If FOO has been defined as a scratchpad register or as a holding register in MICRO then FOO will be defined appropriately for the user.

Any of these registers may be changed by the user if he stops at a breakpoint at a microprocessor instruction. The following locations are setup at the end of an instruction (they may not be changed by the user): LPATH (left bool box output), RPATH (right bool box output), XPATH (the X BUS), YPATH, WPATH, UPATH and HPATH.

Instruction Modification

The instruction in location FOO is of the form:

SIMPOP ZFOO

where ZFOO is the location of a four word vector containing the 90 bits that specify the instruction.

To modify FOO it must first be opened thus:

OPEN FOO;U

If the user wants to see what is in the instruction he says:

.1SET;1 .2SET;2 OFF;#

This will cause a printout of all of the bits which are on, and the value of all of the fields symbolically. Those quantities that are logically fields and their possible symbolic values follow. The quantity in parenthesis is MICRO symbology for setting the field.

LLOC: (.BL) Left bool box: L.MAQ, L.MANQ, L.M, L.Q L.MXQ, L.MOQ, L.Ø etc.

RLOC: (.BR) Right bool box: R.ZAQ, R.ZANQ, R.Z, R.Q, R.ZQ, R.ZOQ, R.Ø etc.

KLOC: (.C) Constant: Ø - 2^{24}-1

BLOC: (.B) Branch Address: SLØ-SLØ+3777B;

CLOC: (.MC) Branch Condition: NOGO,ALLGO,..XZ, (NOGO+N)

OLOC: (.MCONT) Branch Type: GOIF, CALLIF, RETURNIF, GOXIF

PLOC: (.SSP) Scratchpad register: SKØ-SKØ+63

ULOC: (.RRN) Gate out Lower Register: RØ...R6

WLOC: (.LRN) Load Lower Register: RØ...R6

SLOC: (.MS) Special Condition: NOCND, FETCH, (NOCND+N)

Summary of Bits Changeable by User:

FIELD	USE
IHR	Increment holding register output.
TCX	Gate constant field onto X buss.
TCY	Gate constant field onto Y buss.
TSPY	Gate scratch pad register onto Y buss.
THY	Gate holding register selected by RRN onto Y buss.
TXW	Transfer X buss to holding register input.
TYW	Transfer Y buss to holding register input.
TAX	Gate adder output onto X buss.
LOC	Adder low order carry.
TOSY	Gate OS register onto Y buss.
LRØ	Load holding register RØ from X buss or Y buss.
LSPX	Loads scratch pad word addressed by SSP or Z register from the X buss.
LMX	Load M from X buss.
LMY	Load M from Y buss.
LQX	Load Q from X buss.
LQY	Load Q from Y buss.
LZX	Load Z from X buss.
LZY	Load Z from Y buss.

FIELD	USE
VCYP	Don't force 2ØØ nsec. cycle.
DGO	Deferred conditional branch.
TE1Y	Transfer E1 buss to Y buss.
TE2Y	Transfer E2 buss to Y buss.

The Nth special (branch) condition is in NOCND (NOGO)+N, and its symbolic value may be determined by:

```
NOCND (NOGO) + N ←
```

The user may modify any of the above fields and any of the bits not part of one of the above fields after he opens an instruction. Bits are allowed to have two values: ON and OFF.

After all of the modifications are made the register must be closed

```
CLOSE;U
```

When this has occurred the register is reopened so the user may doublecheck his changes. This has the advantage that changes to field are reflected in the bit values (example: all of the constant bits). This also provides a check against the user putting something out of range in the field; if he has, the field will take on a new value.

Simulated Memory

The user may use either paged QSPL memory or normal memory. The first location of normal memory is in SIMMEM and must be initialized by the user. The last legal location (value RØ may take) must be placed in MAXMEM.

If the user wishes to use QSPL paged memory he should set up a pointer to it in SIMMEM, and before executing his program (but after INIT;U) he should set LOCMEM to Ø.

If the user wishes to trap a reference to a particular memory location he puts the location in MEMTRAP (with the 4B7 bit set if he only wishes to trap stores). When the error message

location: R

is printed it means that RØ > MAXMEM or RØ = MEMTRAP. The user may proceed with ;P, allowing the memory operation to proceed.

Patching Programs:

A program may be patched in the obvious manner. Two examples of a patch to set Q to M are now given.

```
FOO+3/   SIMPOP ZFOO+14  ;)
) LABEL: SIMPOP ZLABEL  cr
;F/ ZLABEL:   Ø; Ø; Ø; Ø;   cr
OPEN LABEL;U
   LQX/   OFF   ON  cr
   TAX/   OFF   ON  cr
   LLOC/   L.MAQ   L.Q  cr
   RLOC/   R.ZAQ   R.Ø  cr
```

And a second, easier way:

```
FOO+3/   SIMPOP ZFOO+14  ;)
) LDA  Q; STA M  cr
```

The second method obviously required recompiling before PRINT;U can be executed or paper tapes generated.

Pretty Printouts:

MICRO provides an incomplete bit listing (B and C field don't list external references). There is a QSPL program called FIXEL on PIRTLE's disk area which will generate complete listings. It should be loaded into DIMS along with the entire program for which expanded listings are required. Be sure it doesn't overwrite the simulator or the QSPL routine. Start it with .GO;G. It will ask for the name of an expanded listing file. Say you type /XYZ. It will produce a fixed-up file /N.XYZ and exit. To do another one, start it again with .GO;G.

One peculiarity: it lists the bits (for .B and .C) in increasing numerical order; MICRO does it the other way around.

A Matrix listing of the bits may be generated by saying:

PRINT;U

and supplying a file name. Unfortunately, several punch file names will be asked for, for which the file NOTHING should be provided.

The file thus generated must be printed on the IBM Printer.

Paper Tape Punching:

The File (HECKEL) SNOW will, if loaded into SNOBOL punch paper tapes, by saying:

```
GO.
--OK.
```

and supply when asked:

1) The files to be punched. This may be a dump file, a list of symbolic files, or a list of binary files. After the confirming period for a file name, another period is typed--if this is the last file. Otherwise, a comma is typed to indicate more files to come.

2) The Board Numbers to be printed are suffixed by a 'P' or a'T' for each board to indicate the manufacturer (PRINTEX or TELLER) and separated by commas. The numbers must be decimal and zero indexed.

3) The micro-processor name, 4 character maximum.

4) The tape unit the tape is on.

5) The name of the tape.

SNOW will output a list of cards for a job to be run at STANFORD to put the files on paper tape so they can be punched.

It will also list the message placed on the front of each paper tape that identifies it. This contains:

1) The microprocessor name.
2) The board number.
3) The document number.
4) The version number.
5) The manufacturer.
6) The time and date the magnetic tape was made.

This also updates a file on the disk (PIRTLE)PTLOG which contains that information for all tapes yet punched.

Coding of Special Conditions and Branch Conditions

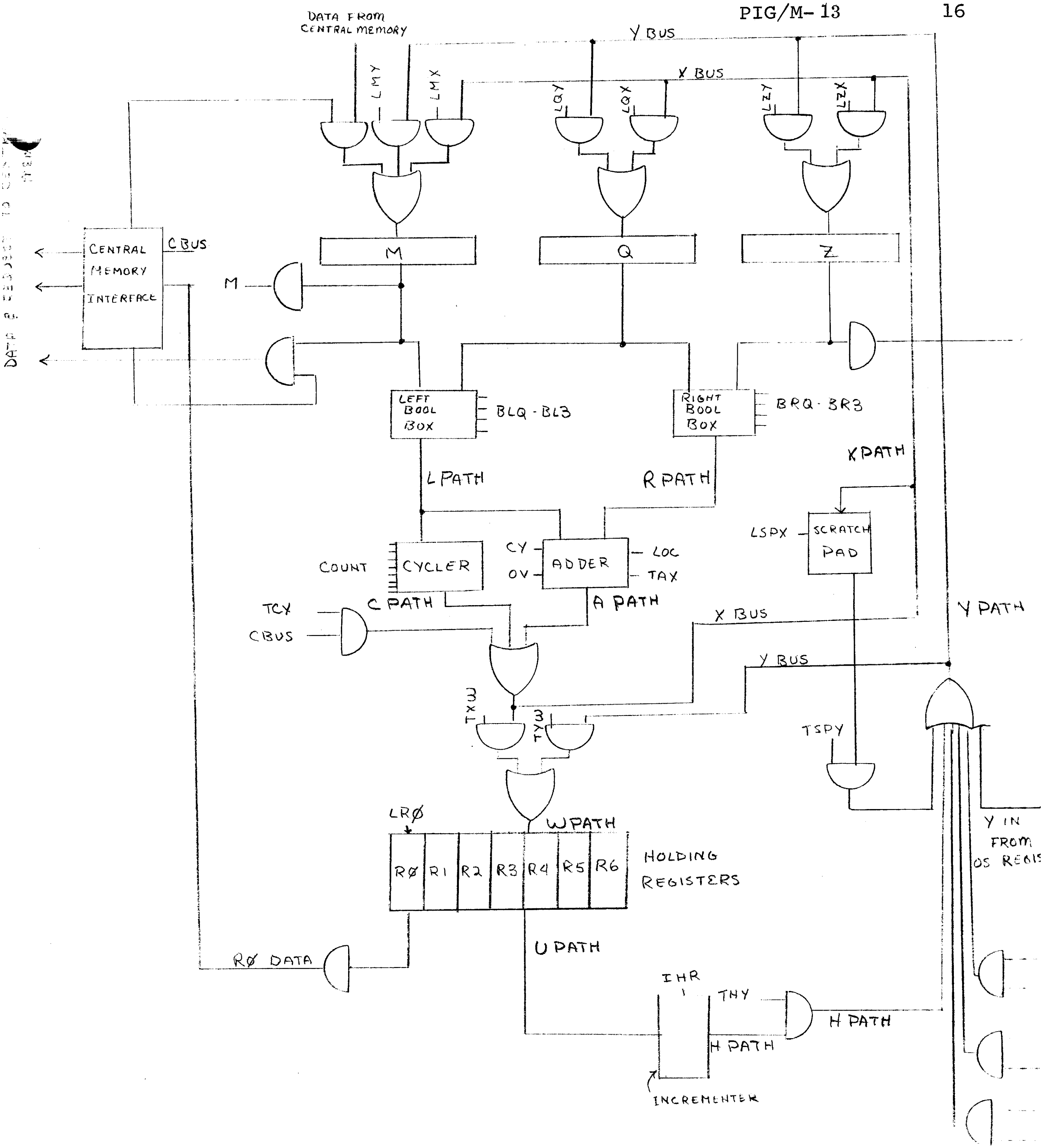
MICRO allows the user to define a special condition or a branch condition as one machine language instruction. Most of these instructions are calls to subroutines which will simulate the hardware.

Branch conditions may reference (but not change) any of the following locations: M, Q, Z, LPATH, RPATH, XPATH, YPATH, UPATH, HPATH, RØ...R6, SKØ...SK63, E1PATH, E2PATH, and WPATH. The code may of course have a temporary storage of its own.

Similarly, when a special condition is executed the above locations will be set up. In addition NEWM, NEWQ, and NEWZ will have the values of M, Q and Z at the end of the instruction. The special condition code is allowed to store into E1PATH, E2PATH and any register. After the code has been executed, the simulator will re-execute the instruction without re-executing the special condition. Thus the code could look at NEWQ and set E2 as a function of this, even though E2 is gated out earlier than Q is loaded. Needless to say, if Q is a function of E2, then its value will be different when the instruction is re-executed.

Code for special conditions and branch conditions is normally put on a separate file and loaded separately with a ';T' command before any of the micro programs are loaded.

Since DIMS lives in QRUN the special and branch conditions may be coded in QSPL and use all of its features.



Appendix I

Figure 1: microprocessor data path
arithmetic section

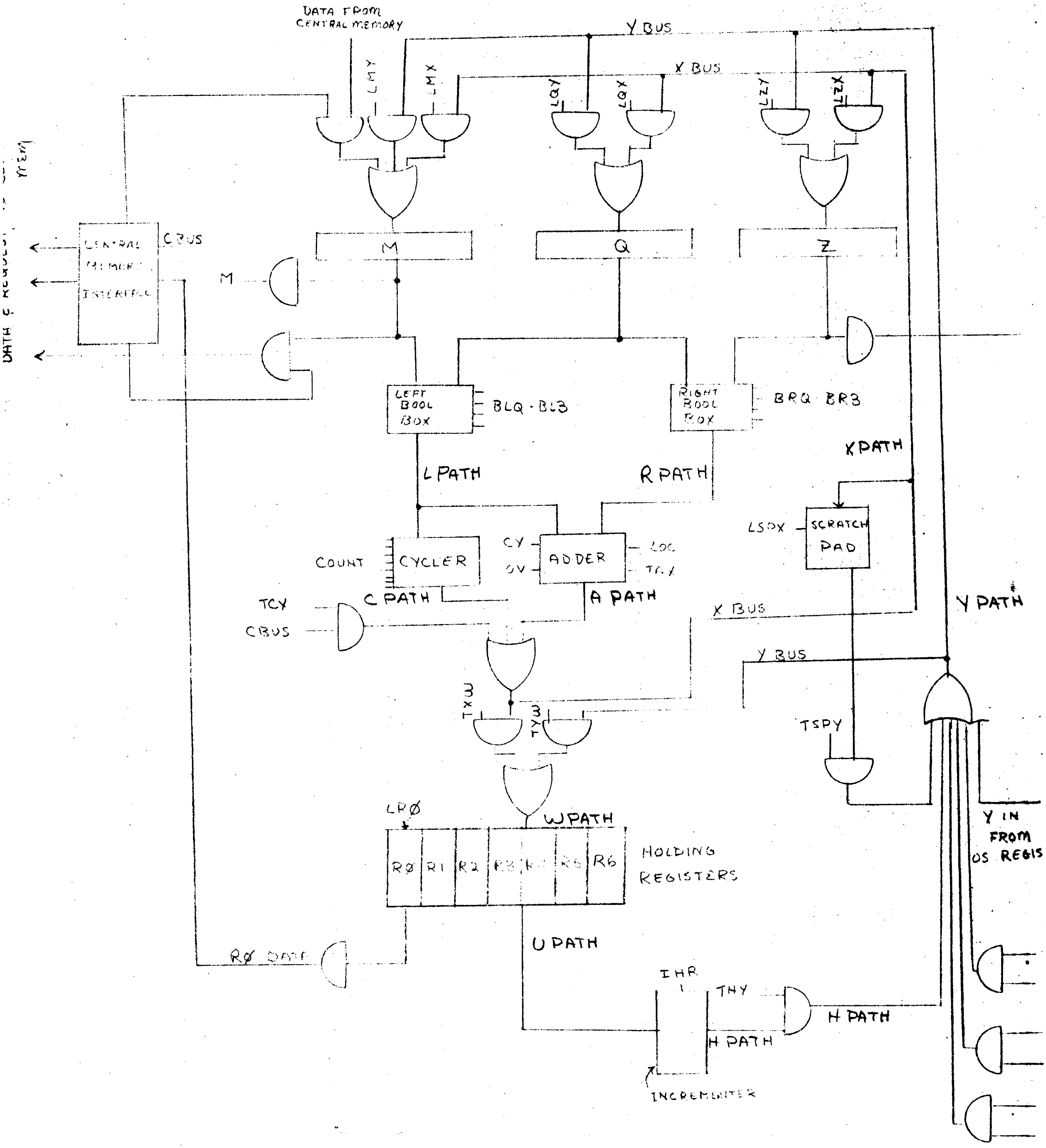


FIGURE 1: MICROPROCESSOR DATA PATH
ARITHMETIC SECTION

APPENDIX I

EXTRA COPY