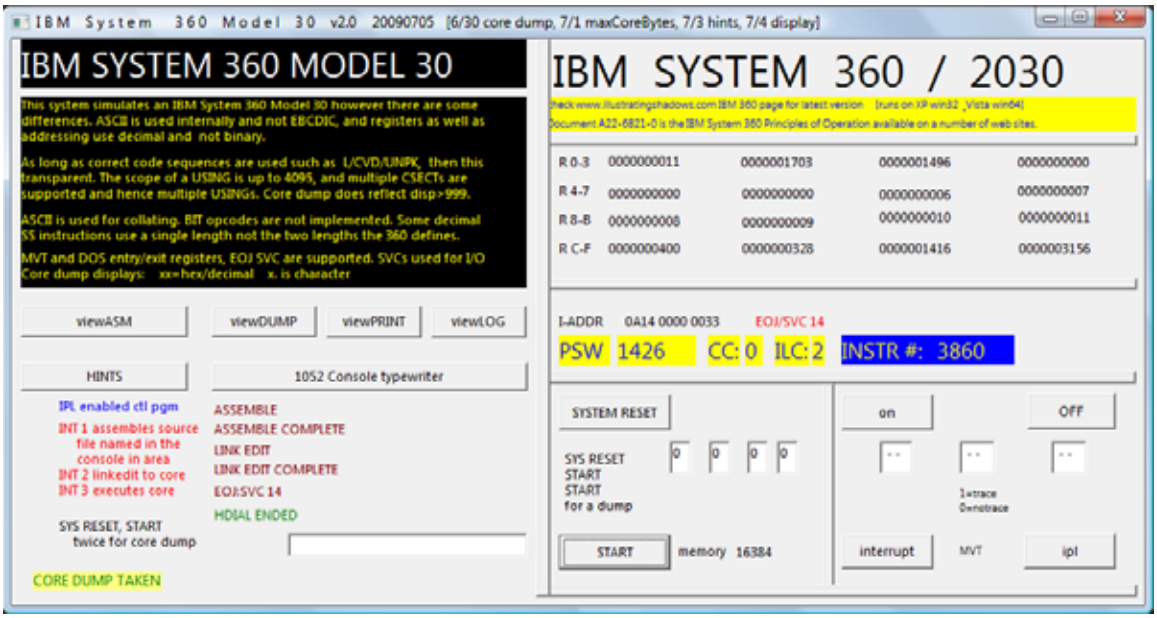
IBM SYSTEM 360 SIMULATOR



---

supporting

- **a graphical control panel**
- **an assembler**
- **with elementary macro support**
- **a link editor**
- **an execution phase with SVC support**
- **a core dump feature**
- **a trace feature**
- **many sample test programs**
- **two sundial programs for a horizontal and a vertical dial**

NOTE: SIM360C has 16k of memory, programs start at 400, address displacements are now 000 to FFF (4095), and an advanced core dump. It runs the H and V sundial programs.

NOTE: To change core storage size, alter *maxCoreBytes* from 16384 to a new value
To change start address, alter *startAddrIs* from 400 to a new value
and then recompile the simulator in Lazarus.

Simon Wheaton-Smith
July 6, 2009
LAZARUS-sim360c-notes.doc

TO GET STARTED WITH THIS IBM 360 SIMULATOR

1. Unzip the sim360c zip file in any folder you so choose

2. Obviously run your virus checker, although all files on

   www.illustratingshadows.com

   are virus and spy-ware checked before all uploads

3. using MY COMPUTER go to the folder you just used

4. double click **system360project.exe**
   or **~0start here.bat**

5. click the **POWER ON** button, then the **IPL** button

6. ensure the CONSOLE IN area has **sysin.txt**
   or your desired source code

7. then click **INTERRUPT** which has a code of '1' and it assembles the file named in the CONSOLE IN area

8. then click **INTERRUPT** again, which should now have a code of '2' which loads core storage. Also you will not that the code by the INTERRUPT BUTTON is now a 3

9. if desired, enter your latitude tens digit into switch 1, and the unit digit into switch 2, similarly load the longitude difference from the meridian into switches 3 and 4. Never place two digits into one switch.

10. click **INTERRUPT** which now has code 3, and the program will run.

11. click **SYSTEM RESET** then **START** and **START** again if you wish, which takes a core dump. Microcode in the IBM 360/2030 used this with 090E in the rightmost rotary switches for a standalone dump.

12. look at the SYSPRINT file for your output, and SYSDUMP has the core dump. Buttons let you do this easily.

NOTE: This system uses POWER ON to establish the GUI display area, and IPL to get things ready internally. INTERRUPT is used for things the old BPS and BOS programs did, in this case, assemble, link, and execute.

NOTE: There are many small test files in the TEST folder and they are all called TESTnn.TXT and you can move them to the simulator's folder, and assemble them by placing their name in the CONSOLE IN area.

NOTE: This system provides a vertical as well as a horizontal sundial program with latitude and longitude difference enterable by switches. A dial west of meridian is assumed, for dials east of the legal meridian, use PM for AM and vice versa.

TO RECOMPILE THIS IBM 360 SIMULATOR

1. Install the Lazarus system, see page 15 approx of this booklet
even for Vista win64, use the 32 bit version
do not use the version with QT in the file name

   http://www.osalt.com/lazarus web site for Lazarus

   And locate the download link:

   http://sourceforge.net/project/showfiles.php?group_id=89339

   and locate **the Windows 32 bit** version even if you have a 64 bit machine.

YES	lazarus-0.9.26-fpc-2.2.2-win32.exe	58455268	i386
NO	lazarus-qt-0.9.26-fpc-2.2.2-win32.exe	58420736	i386

   the version for Windows XP was about 58mb:

   lazarus-0.9.26-fpc-2.2.2-win32.exe

   but **do NOT** download:

   lazarus**-qt-**0.9.26-fpc-2.2.2-win32.exe

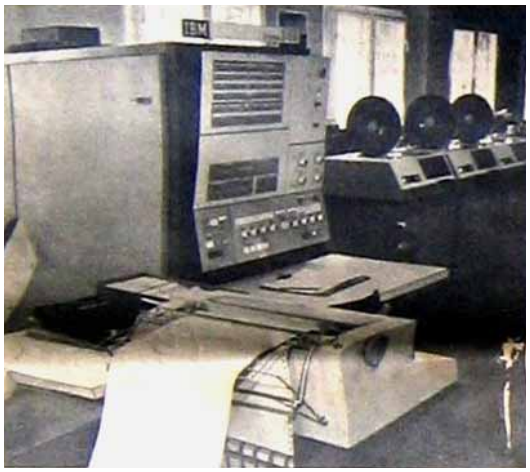
   because you will get very frustrated trying to locate: qtcore4.dll

2. Unzip the sim360c zip file in any folder you so choose
3. Obviously run your virus checker, although all files on

   www.illustratingshadows.com

   are virus and spy-ware checked before all uploads
4. Bring up Lazarus
5. select PROJECT, and then OPEN PROJECT
6. locate the folder from step 2
7 double click on the *.lpi file: system360project.lpi
8. to compile select RUN, if the compiler stops after the build and does not bring up the program, select RUN and RESET DEBUGGER
9. That is all there is to it.

The first IBM 360 the author used at Thos Cook & Son Ltd, this is a 360/30 and the author used BPS, BOS, and DOS on this system.



Later he used the 360/50 and 65, the 370/145, 138, 148, 158 under various operating systems including MFT and MVT, VS1/SVS, VS2/MVS, then GCS under VM.

This simulator is close to the 360 architecture however it has subtle differences. The source code is open source and is very easy to extend and even SVC code can be easily added.

The actual panel is larger than below, but the panel below helps emphasize the features.

key notes

registers – ignore the high order 1

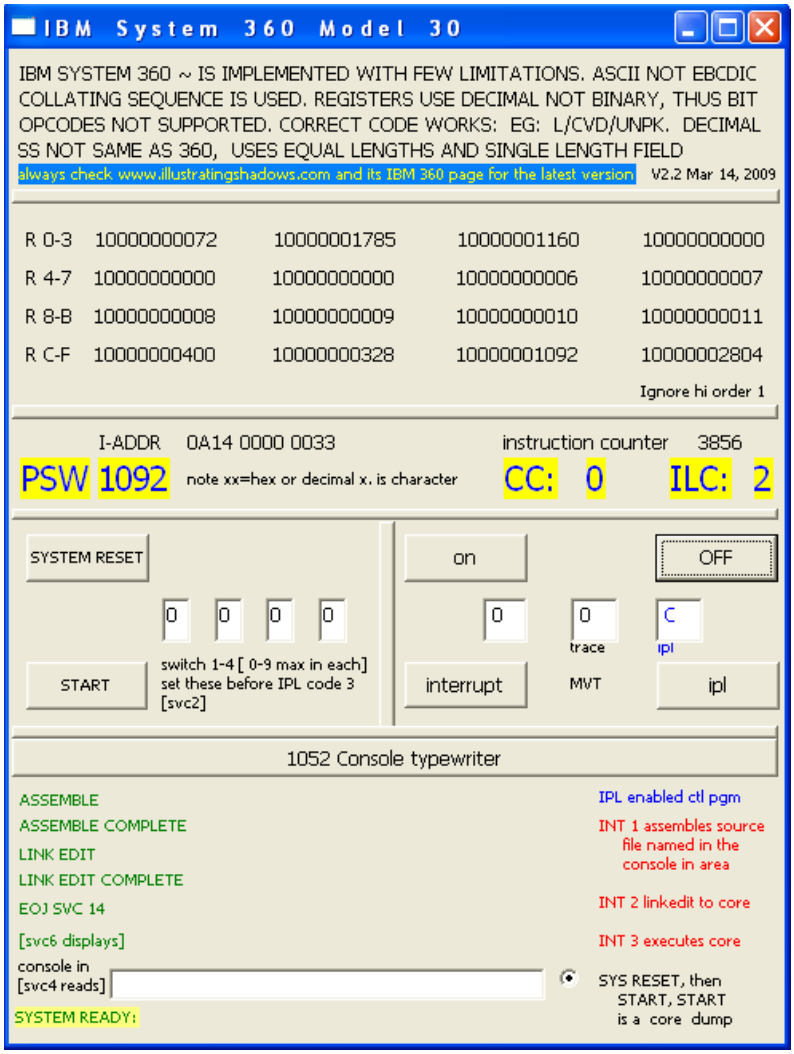
program status

switches and buttons

hints on switches

asm, lnk, exec status

console input current status



SIM360C IS AN IBM 360 SIMULATOR WITH A BUILT IN ASSEMBLER, LINKER, AND INSTRUCTION SIMULATOR, WITH TRACE, CORE DUMP, AND SVC SUPPORT.

Sim360c is a program that gives the flavor of Basic Assembler Language programming for an IBM 360. It is a subset design that does not implement all the machine instructions of the IBM 360 or all of the pseudo instructions of an assembler for that machine.

Sim360c has three distinct phases, an assembler with its own three phases, a linkage editor, and a simulator. The first assembler phase inserts macros, the second reads assembly language instructions and generates machine language and an assembler listing, and also generates a symbol and USING table. The third assembler phase completes symbols and addressability and stores intermediate code and constant data. In the second simulator phase, after the assembler phases, the link edit phase loads the final assembler output into core storage, and additional checks are made. In the third simulator phase the program fetches and executes instructions.

RESTRICTIONS:

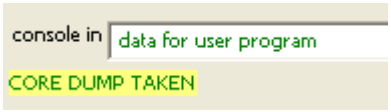
SVC is supported with SVC 1 for printing, SVC 2 for reading the four data rotary switches, SVC 3 for a blank line, SVC 4 for reading 1052 input, SVC 5 for trace on/off, etc, and SVC 14 for EOJ, etc. A list of SVC codes appears later.

Certain features that are completely absent include

the floating point hardware,
some decimal arithmetic feature,
some instruction formats of types SI and SS.

Recall that the idea is to give the flavor of, not an exact simulation. However, what is implemented is reasonably faithful to the original.

NOTE: The console in area can be used in two ways:-



NOTE: INTERRUPT and code 1 means assemble, and the source file name comes from the console input area which defaults to "360sysin.txt" but any file name in the simulator's folder can be named.

NOTE: The console input area is always looked at when INTERRUPT and code 1 is used. However, a program can use that area with SVC 4 for a single line of user text input.

FILENAMES:

360sysin.txt	default file that IPL and code 1 assembles, you can type in a different file name
360syslog.txt	console information and tracing is desired
360sysdump.txt	core dump with SYSTEM RESET, START, START
360sysprint.txt	print output that SVC 1 produces
360sysMacro.txt	macro library
~360asmPass1.txt	check for macros expanded properly
~360asmPass1.txt	check for *** ERROR ***, pass 1 output
~360asmPass2.txt	check for *** ERROR ***, pass 2 output and object code

With this simulator comes a horizontal sundial program **hdial.asm** which to be assembled and executed is saved as **360sysin.txt** although at INTERRUPT-1 time any file name can be entered in the console input area. USING is supported for all registers and there can be several CSECTs each with their own USING. The START ADDRESS for all programs is specified in the simulator source code, with **400** being a wise choice.

SAMPLE PASS 1 OF ASSEMBLER OUTPUT

```
000400                          HDIALPGM CSECT ,                       MAIN PROGRAM
000400 >U 12                             USING *,12                    TELL ASSEMBLER
000400 RR 18 ra:rb                       LR    12,15                   SET R12 AS BASE
000402                          *
000402                          *-------------------------------------------------
000402                          *
000402                          *******
000402                          *BEGIN*--         GETSWITCHES
000402                          *******
000402                          * SWITCHES SAVED BY
000402 RR 1B ra:rb                       SR    1,1                     CLEAR R1
000404 RR 1B ra:rb                       SR    2,2                     CLEAR R2
000406 RR 1B ra:rb                       SR    3,3                     CLEAR R3
000408 RR 1B ra:rb                       SR    4,4                     CLEAR R4
000410 RR 0A ra:rb                       SVC   0,2                     LOAD REG
000412                          *******
000412                          **END**--         GETSWITCHES
000412                          *******
000412 RR 18 ra:rb                       LR    5,1                     SEE IF ANY
000414 RR 1A ra:rb                       AR    5,2                     SWITCHES
000416 RR 1A ra:rb                       AR    5,3                     WERE ENTERED
000418 RR 1A ra:rb                       AR    5,4                     BEFORE IPL 3
000420 RR 12 ra:rb                       LTR   5,5                     TEST R5
000422 RX 47 ra xx:bb:dddd               BC    08,BEGIN                   ZERO THEN
```

SAMPLE PASS 2 OF ASSEMBLER OUTPUT

```
000412 RR 18 05:01                       LR    5,1                     SEE IF ANY
000414 RR 1A 05:02                       AR    5,2                     SWITCHES
000416 RR 1A 05:03                       AR    5,3                     WERE ENTERED
000418 RR 1A 05:04                       AR    5,4
000420 RR 12 05:05                       LTR   5,5                     TEST R5
000422 RX 47 08 00:12:0062               BC    08,BEGIN                ZERO THEN
```

SAMPLE SYSLOG

```
POWER ON:
ASM PASS 1: started
ASM PASS 1: ended
ASM PASS 2: started
ASM PASS 2: ended
LINK EDIT: started
LINK EDIT: ended
EXECUTE:
EXEC: ADDR=0      CTR=1 OPCODE(S)=05C0 1B11 1B22     CC=0
                      R0-3: 0 1 2 3     R4-7: 4 5 6 7
                      R8-B: 8 9 10 11     RC-F: 2 7992 14 0
EXEC: ADDR=2      CTR=2 OPCODE(S)=1B11 1B22 1B33     CC=0
                      R0-3: 0 0 2 3     R4-7: 4 5 6 7
                      R8-B: 8 9 10 11     RC-F: 2 7992 14 0
. . .
EOJ:
CORE DUMP: started
POWER OFF:
```

SAMPLE SYSDUMP

```
* CORE DUMP BEGIN - REGISTERS [DECIMAL]*

 GPR  0   56
 GPR  1   1456
 GPR  2   1080
 GPR  3   0
 GPR  4   0
...
 GPR 12   400
 GPR 13   328
 GPR 14   950
 GPR 15   1624

* CORE DUMP BEGIN - CORE STORAGE *
* NOTE - XX IS HEX OR DECIMAL    *
*      - X. IS CHARACTER         *

CONTROL PROGRAM

DEC.ADR    +0        +4        +8        +12       +16       +20       +24   .... etc
000000     ........  ........  ........  ........  ........  ........  ........
000040     ........  ........  ........  ........  ........  ........  ........
000080     ........  ........  ........  ........  ........  E.X.E.C.  ........
...
000280     ........  ........  ........  ........  ........  ........  ........
000320     ........  ....0A14  ........  ........  ........  ........  ........
000360     ........  ........  ........  ........  ........  ........  ........

USER PROGRAM
DISPLACEMENTS ARE DECIMAL, IF > 999 THEN THOUSANDS SHOWN ABOVE DDD
DEC.ADR    +0        +4        +8        +12       +16       +20       +24      ...

000400                                                                  0       ...
  0400     18CF1B11  1B221B33  1B440A02  18511A52  1A531A54  12554780  C0621B00 ...
000440                1                   0                   1                 ...
  0440     1B005000  C0421B00  181341F0  00101C0F  1A145010  C0420A03  0A034100 ...
```

SAMPLE SYSPRINT

```
HORIZONTAL SUNDIAL PROGRAM ON THE IBM SYSTEM 360  - - -  S WHEATON-SMITH
=========================================================================

                    LATITUDE IS        033
                    SIN(LAT) IS      0.544
                    LONGITUDE DIF      003
                    IN MINUTES         012

MORNING HOURS - - - - - - - - - - - - - - -
                    HOUR FROM NOON [HA]  HOUR LINE ANGLE
                         005 [78]              069
                         004 [63]              047
                         003 [48]              032
                         002 [33]              020
                         001 [18]              010
                         000 [03]              002
AFTERNOON HOURS - - - - - - - - - - - - - - -
                    HOUR FROM NOON [HA]  HOUR LINE ANGLE
                         001 [12]              007
                         002 [27]              016
                         003 [42]              027
                         004 [57]              040
                         005 [72]              060
=========================================================================
```

Two sundials programs are provided, hdial.asm and vdial.asm (test98.asm), and latitude as well as longitude are specifiable with the console switches.

THE SYSTEM 360 ASSEMBLER AND LINK EDIT DESIGN

In pass one (Lazarus source code says pass0), macros are expanded. Then pass two (Lazarus source code pass 1) saves labels and assigns instruction and constant sizes, and places a generic instruction model and generic data models for instructions and constants. Pass three (Lazarus source code pass 2) completes instructions, and assigns base/displacement addresses. CSECT and USING is supported however unlike the real assembler, you cannot switch back and forth among CSECTs at assembler time, as some IBM assemblers allowed. But you can have very large programs provided each CSECT is kept to the span of one register. The maintained V2 simulator has 16K which you can expand, and has normal displacements allowed of 000 to 4095.

This link edit phase loads assembler output to core, and manages character constant loading.

THE SYSTEM 360 SIMULATOR ARCHITECTURE

The console, and the core dump, provide clues on how this simulator functions internally. The System 360 obviously supports several arithmetic systems, however, this system internally does things differently.

SYSTEM 360	SIMULATOR
half and full word binary	Pascal decimal
double word as in CVD	Pascal decimal but with a sign byte in the last low order
packed	usable for UNPK
character	stores in alternating Pascal bytes

NOTE: as long as you program correctly, this is transparent to the programmer.

```
*** WORKS ON THIS SIM IF R1 HAS A DC OF
*** F'xxx' BUT IS WONT WORK ON Z390 ETC
 STH   1,SINLATH          SAVE SIN LAT
 UNPK  M3HDRSIN,SINLATH   MAKE PRINTABLE
```

```
*** CORRECT CODE IF R1 HAS A DC F'xxx'
 LA    2,DWD         GET ADDRESS OF DWD
 CVD   1,DWD         CONVERT IT
 MVC   SINLATH,6(2) FROM DWD+6
```

NOTE: however it is possible to misuse the code and still work, because of the internal workings of this simulator.

NOTE: registers are thusly stored internally in Pascal decimal.

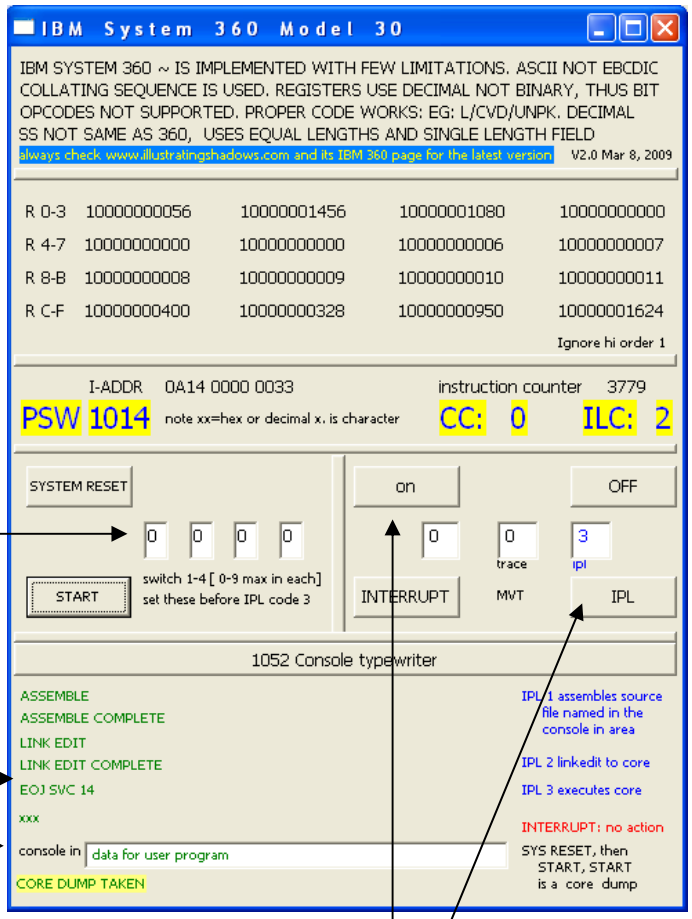
NOTE: instructions look as if they are in hex, in fact the two nibbles of an instruction are stored as two Pascal characters.

NOTE: code that modifies itself will be incorrect. For this reason, code is assumed to be reentrant, not just serially reusable.

KEY POINT: Because decimal is used as opposed to hex, for the most part transparently provided good 360 coding techniques are used, displacements in B-DDD addresses can still be up to 4095 because a special array (only visible in a core dump) handles thousands. Address constants don't need a special array, and can address CSECTs, and there is no limit on the number of CSECTs, each of which has their own USING. Also the old BPS and BOS trick of resetting a base and USING works, although this is terrible programming practice, the sundial programs show this as commented code.

THE SYSTEM 360 CONSOLE

[to the right is an earlier version of the panel. The current panel is more involved]



The four data switches each can hold one digit 0-9, and are read by an SVC if desired.

The 1052 console typewriter displays output. It thus says if assembling, linkediting, or executing, or has hit a problem or an EOJ (SVC 14), etc.

The 1052 can be read by SVC 4 in the user program

POWER ON is needed before anything is displayed. POWER ON enables internal software, and INTERRUPT with a code of 1 assembles, and with a code of 2 linkedits, and with a code of 3 executes.

The interrupt button is on the left of the three switches, the IPL button is the right most.

The middle switch turns on and off tracing, the right controls assembly, link, or execute. The left switch does nothing at present.

The PSW instruction address shows the last instruction executed when everything comes to a screeching halt. It shows the condition code, the instruction, its address (again), and the instruction counter.

The registers are shown in decimal.

In the simulator code, exceptions are not handled. Provided STRIP and UPX were not used, Lazarus will take you to the source code that got upset. Either way, **if the simulator crashes**, always do the following:-

1. look at pass 1 for errors
2. ditto for pass 2
3. and the link edit may say there was a B-DDD error with a DDD > 999

Implemented mnemonic operation codes by type follow:

RR: most implemented including DR and MR
RS: not implemented LM STM
RX: most implemented

(The following extended mnemonics for BC and BCR are also implemented:

B BE BH BL BNE BNH BNL BNM NOP

The pseudo assembler instructions also exist: DC H F D C P
CSECT USING
END CNOP

The RX instructions can only accept a few combinations of operands. The memory designation, will normally be a label. Base and displacement notation is allowed in some cases, but base, index and offset notation is not allowed. Neither is a label followed by a + and constant.

The mask specification of the BC must be a simple decimal integer in the range 0-15. Normally useful masks: such as B'1100' are not accepted.

Literals are not allowed, use constants, and if code is large, use multiple CSECTs and address constants.

The type specifications for a DC are H, F, D, C and P

UNPK requires you to study the source field, define a DC P or DC C of an appropriate length. That target field's length determines the entire operation, in other words the source length field is not used.

EQU is supported but only with the implied operand of "*", useful for code such as:-

```
LA      0,ENDOFCON      END OF A CONSTANT AREA        defined as EQU *
LA      1,STROFCON      START OF CONSTANT AREA        defined as DC C'...'
SR      0,1             R0 = SIZE, R1 = AREA
SVC     1               WRITE A LINE
```

I/O is provided by SVC, and the SVC codes are shown later in this document. SVC code is implemented in the simulator's I-CYCLE phase (EXECUTE) and not in low or high core as part of a control program. The only control program or any other stuff places in low or high core is the SVC 14 that R14 points to at program start up (to simulate MVT code conventions) and a save area to which R13 points on entry, also for MVT conventions. DOS (IBM 360 DOS) conventions work, namely BALR 12.0/USING *,12 and EOJ also work.

NOTE: Document A22-6821-0 is available online in a number of places and is the IBM System 360 Principles of Operation.

SOME GUIDELINES FOR SIM360C

The format for an assembly language statement is:

```
1.......10....16.....................41.....................
[LABEL]  OPCDE OPERANDS                 COMMENTS
```

where: Label: is an optional 1-8 character label that identifies this statement or variable.

Opcode: one of the instructions of the machine, the opcode determines the format of the operand field. Pseudo opcodes as well as simple macros are supported.

Operands: usually two or three operands with no intervening blanks. Macros only have one operand available, and that must be in column 16, and is &PARM

Comments: Any text following the operands is ignored. A line starting with an asterisk (*) is ignored as a comment.

Variables are declared using the DC (for initialization), there is no DS

A DC often has a label, so that it may be referenced using that label in an instruction. The assembler uses the DC length attribute for SS instructions.

Op-codes implemented

RR	
05	BALR
06	BCTR
07	BCR
0A	SVC
10	LPR
11	LNR
12	LTR
13	LCR
15	CLR
19	CR as CLR
18	LR
1A	AR
1C	MR
1E	ALR as AR
1B	SR
1D	DR
1F	SLR as SR

RX SI	
40	STH
41	LA
42	STC *1
43	IC *1
45	BAL
46	BCT
47	BC
48	LH
49	CH
4A	AH
4B	SH
4C	MH
4E	CVD
50	ST
55	CL
58	L
59	C as CL
5A	A
5B	S
5C	M
5E	AL as A
5F	SL as S
92	MVI
95	CLI

RS	
90	STM
98	LM

SS	
D2	MVC
D5	CLC
F2	PACK
F3	UNPK
F8	ZAP
F9	CP

SS Decimal use only one length in this simulator.

*1 IC/STC is numeric not any character, as sim has decimal registers.

Op-codes not implemented

RR

SPM	04

very unlikely

NR	14
OR	16
XR	17

RS RX SI

EX	44
CVB	4F
D	5D
SRL	88
SLL	89
SRA	8A
SLA	8B
TS	93
CLM	BD
STCM	BE
ICM	BF

unlikely

BXH	86
BXLE	87

very unlikely

N	54
O	56
X	57

SS

XC	D7
TR	DC
TRT	DD
ED	DE
EDMK	DF
SRP	F0
MVO	F1
AP	FA
SP	FB
MP	FC
DP	FD

unlikely

TM	91
NI	94
OI	96
XI	97

very unlikely

NC	D4
OC	D6
MVN	D1
MVZ	D3

CORE STORAGE

ab cd ef	hex as in op-codes is stored looking like hex in 360 nibbles as PC bytes
ab cd eS	decimal packed is in 360 nibbles as PC bytes
A. B. C.	character is stored as A~Z 0~9 etc in one nibble with the next nibble a "."

MACRO FACILITY FOR THIS ASSEMBLER

The assembler has three passes, pass zero expands macros, pass one assigns labels and instruction models, pass two completes the object code. Pass zero reads the macro file and builds a list of macros, and when they are found in the source code they are expanded. Unlike the IBM 360 assembler, &NAME and &PARM are special keywords and do not have to be on the macro prototype.

```
*
* USER SOURCE CODE
*
* 1...5...10...15...20...25
* label    macro parm
* |              |
* |              +---> inOperands -> &PARM
* |
* +------------------> inLabel ----> &NAME
*
* When &NAME found then inLabel is substituted if in cols 1-8
*
* When &PARM found then inOperands is substituted if in cols 16 on
*
* NOTE:  &NAME and &PARMS are special words so use them and do not
*        use other &values
*
* NOTE:  When &PARM is used, you must leave at least 3 spaces
*        for example
*                       DC    A(&PARM)      is wrong
*                       DC    A(&PARM   )   is correct
```

```
         MACRO
&NAME    CALL  &PARM
         CNOP  0,4
&NAME    BALR  15,0
         BAL   15,8(15)
         DC    A(&PARM   )
         L     15,0(15)
         BALR  14,15
         MEND
```

the left is the same as the right

```
         MACRO
         CALL
         CNOP  0,4
&NAME    BALR  15,0
         BAL   15,8(15)
         DC    A(&PARM   )
         L     15,0(15)
         BALR  14,15
         MEND
```

Look at 360sysMacro.txt for macro availability. For example, EOJ is a macro and expands as follows:-

```
000224                        *******
000224                        *BEGIN*--         EOJ            EXIT
000224                        *******
000224 RR 0A 01:04                      SVC   1,4
000226                        *******
000226                        **END**--         EOJ            EXIT
000226                        *******
```

OPERATING SYSTEM SUPPORT FOR THIS SIMULATOR

The simulator includes an assembler with elementary macros, a link editor which mostly loads compiled programs into core, and an execution phase. Programs are of little use if they cannot read requests and cannot output answers.

INPUT ~ The simulator provides two SVC codes for reading the console in area and the switches on the CPU console. Just make sure that the switches are set and the console in area loaded before executing the program with IPL and code 3.

OUTPUT ~ The simulator allows a text area of a given size to be printed to SYSPRINT, and this is handled with an SVC code.

SVC SUPPORT ~ is in the execute phase of the simulator, and is very easy to modify.

SVC 1	R0=area size	R1=DC C area	print ioarea to SYSPRINT.TXT
SVC 2	R1, 2, 3, 4		these four registers have the contents of switches 1, 2, 3, 4 set before IPL code 3
SVC 3			print one blank line to SYSPRINT.TXT
SVC 4	R0=area size	R1=DC C area	place in this area text in the 1052 input area that set before IPL code 3
SVC 5	R0=0 for off, 1 for on		turn trace on or off, uses SYSLOG.TXT
SVC 6	R0=area size	R1=DC C area	print ioarea to the 1052
SVC 7	R0=offset		shift right print lines by an offset

CONTROL PROGRAM (MFT and MVT) or SUPERVISOR (DOS) ~ The simulator loads low core (memory) with a few things. First, if the STARTADDRIS value is less than 100 then there is no such preloading, and all programs should follow DOS conventions. DOS demands the user load a base and terminate with EOJ.

```
PROGRAM CSECT
        BALR   12,0
        USING  *,12
  - - - - - - - - - - -
        EOJ
```

Whereas MFT/MVT preloads R15, and a BR 14 (or a RETURN) terminates a program.

```
PROGRAM CSECT
        USING  *,12
        LR     12,15
  - - - - - - - - - - -
        BR     14
```

As long as STARTADDRIS is appropriate, then low core has a save area established and an SVC 14 as well, and they are above the PSW old and new areas. So, with the default STARTADDRIS = 400 then any program may use DOS or MFT/MVT conventions. While the simulator says what operating system conventions are used, this is informational only.

The core dump will show an EOJ in low core (0A14) and a save area, and the SVC new PSW interrupt address will say EXEC. The EOJ/0A14 and the save area are for real. The EXEC is not for real, it is just a clue that the execute phase of the simulator handles the SVC calls.

LAZARUS Open Source version of, and almost compatible with: DELPHI:

At this website there are some downloads for Lazarus: http://www.osalt.com/lazarus

And locate the download link: http://sourceforge.net/project/showfiles.php?group_id=89339

and locate **the Windows 32 bit** version even if you have a 64 bit machine.

YES	lazarus-0.9.26-fpc-2.2.2-win32.exe Mirror	58455268	i386
NO	lazarus-qt-0.9.26-fpc-2.2.2-win32.exe Mirror	58420736	i386

the version for Windows XP was about 58mb: lazarus-0.9.26-fpc-2.2.2-win32.exe

but **do NOT** download: lazarus**-qt-**0.9.26-fpc-2.2.2-win32.exe
because you will get very frustrated trying to locate: qtcore4.dll

documentation is available online at a url something like:-

http://wiki.lazarus.freepascal.org/Lazarus_Documentation#Lazarus_and_Pascal_Tutorials

The download is one single file, installs first time, and runs first time. The tutorial to get started is helpful, and located at:-

http://www.lazarus.freepascal.org/
http://wiki.lazarus.freepascal.org/Lazarus_Tutorial

DELPHI

This can be compared to DELPHI, which is 332 mb, and the pre-reqs another 234 mb. Delphi can be found at: http://www.turboexplorer.com/delphi

COMMENTS

LAZARUS is an Open Source program, based on PASCAL, and is somewhat compatible with Delphi.

One of the shortcomings of JAVA, and other object oriented languages is their type conversion issues. The graphical program for sundials highlights this problem, namely converting from floating point to integer requires an intermediate string conversion! Harking back to IBM's PL/I language, some lessons that the new language developers could learn emerge.

First: PL/I had as one of its values the concept that if a programmer could write something that made sense to him or her, then the PL/I compiler should also be able to make sense of it. All these newer languages or language adaptations are very weak on real world needs of commercial programmers, and seem to be more suited to those who delight in getting around complexities of a language. LAZARUS is no exception, and the documentation is designed for those who already know the system.

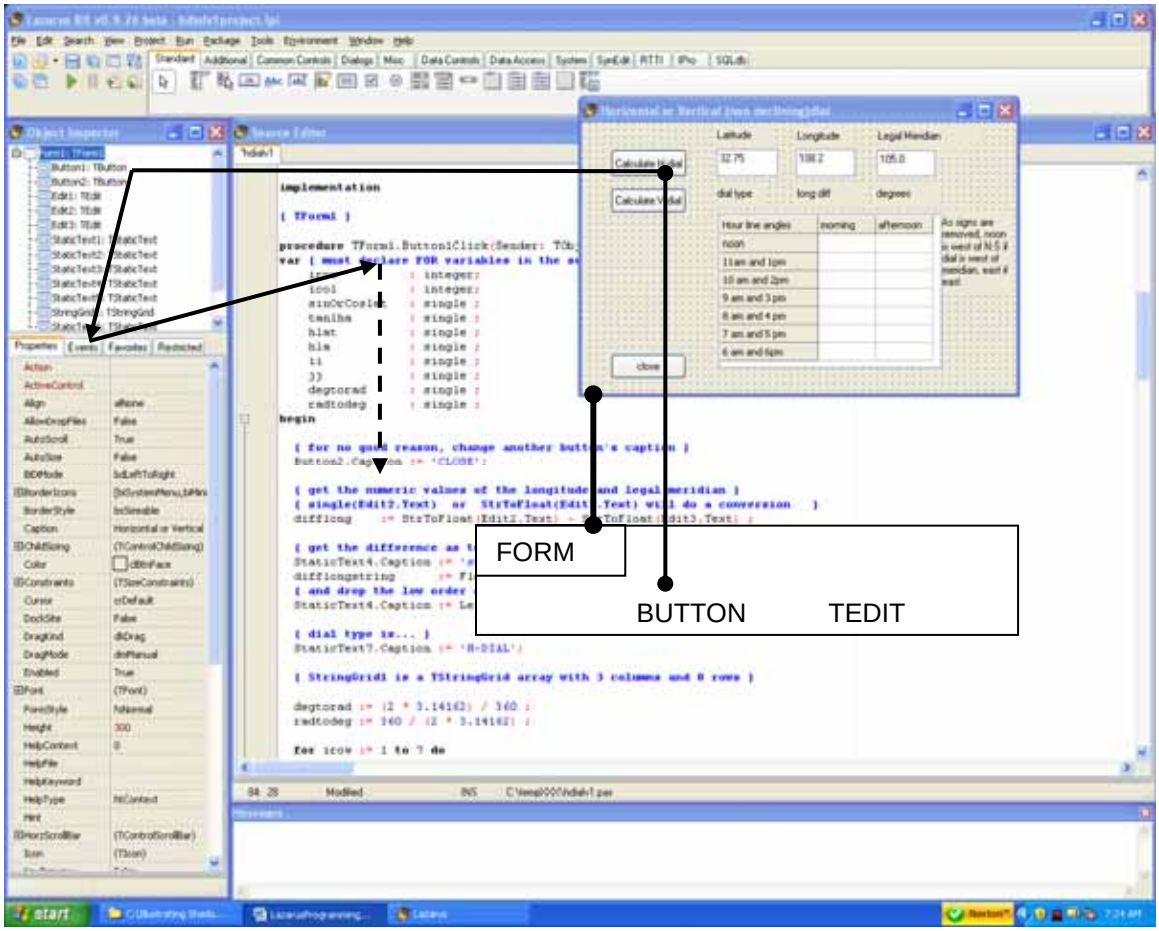
Second: PL/I had the ability for almost any data type to be converted implicitly so that a programmer could take in a string of characters that contained numbers and implicitly convert it to an integer, or floating point number, and vice versa.

A PROJECT'S HIERARCHY

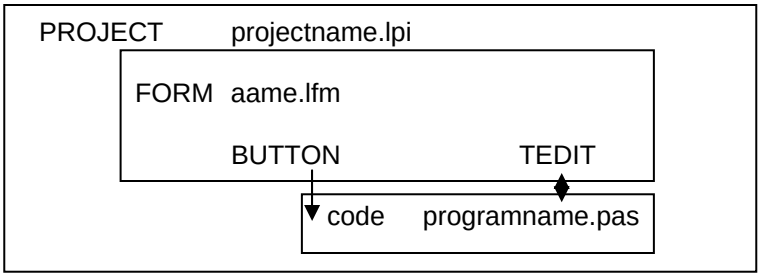
In LAZARUS, as in Microsoft's VISUAL languages, there is a hierarchy. In a conventional program there is always a main program, and it calls sub programs, that use functions as well as language structure.

Lazarus, based on PASCAL, at least doe not confuse inherent language function with functions and classes. Some languages tae classes to extreme and the blurring of those discrete boundaries causes programmers to fight the language in order to achieve the end goal.

There is a hierarchy in Lazarus similar to that of Visual Basic. The front end is the FORM.

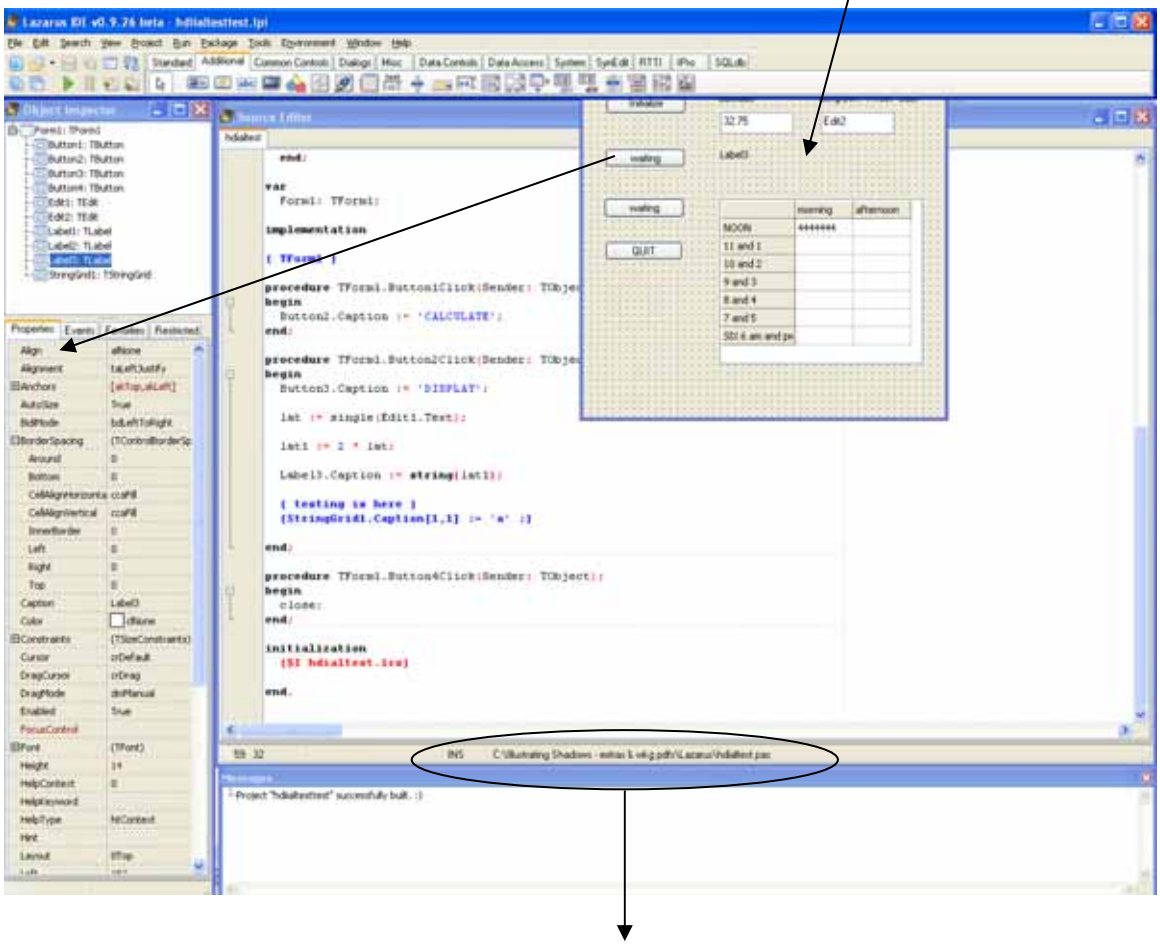


The FORM is what essentially starts up when the "project" is executed, and one or more buttons in the form trigger code. And that code can use variable data such as the TEDIT areas.



NOTES ON LAZARUS

Lazarus has an IDE and the forms window is simple to use to design a form that will drive the program. The content and activities for each button are intuitive.



Running a program that you built as an application may cause the debugger to crash. The debugger is how the IDE runs programs. However, locate the folders in which the programs are stored, bring up those folders and double click the program, and all will run, albeit without debugging. The folder containing the program can be found by SAVE AS, or by looking at the area circled in the about picture of the Lazarus IDE desktop.

The cause of Lazarus problems in execution is simple, the designers and implementers did not manage folders with blanks in a name. So, if you wish to debug in Lazarus, you must save files in a folder or chain of folders whose names from the root folder up to the folder holding the project, have no names with blanks in them.

This issue of blanks in a name is not uncommon, so simply create folders with no blanks in their names for Open Source systems.

NOTE: **Please refer to file: LazarusProgrammingNotes.pdf for detailed notes on Lazarus programming, the preceding two pages are merely an overview.**

LAZARUS – REDUCE EXECUTABLE FILE SIZE

Lazarus files are very large, some 12mb for a small program. This is because of enormous amounts of debug data. And while the compiler can remove that, there are bugs, so that is not an option.

The solution is to locate programs STRIP and UPX, however they only work on 32 bit systems.

STRIP is Pascal specific and locates and removes the debug data after the fact, making the executable some 20% of its original size, 12mb becomes less than 2mb.

UPX works on any executable, and shrinks the executable even further, some 2mb becomes about 0.5mb.

c:\whereever\lazarus\fpc\2.2.2\bin\i386-win32\strip

move to your folder

go to that folder

do "strip --strip-all system360project.exe"

which reduces it from 12mb to just under 2mb

c:\whereever\lazarus\fpc\2.2.2\bin\i386-win32\upx

move to your folder

go to that folder

do "upx system360project.exe"

which reduces it from just under 2mb to just under 0.5mb

WINDOWS VISTA WIN64 ISSUES

The Lazarus 32 bit system works on Windows XP as well as Vista win64, and generated code can be compressed with STRIP and UPX. However, the 64 bit version of Lazarus produces code than will not run on win32 nor on Windows XP, and cannot be compressed either with STRIP and UPX since they do not support 64 bit executables. At best, WINZIP will do a fair job of compression.

LAZARUS APEARS TO HANG AFTER "RUN", AND THE PROGRAM DOES NOT APPEAR

Lazarus 32 bit in Vista sometimes does not bring up the program after a RUN. Click RESET DEBUGER, when all proceeds normally. However you will get an OOPS message from Lazarus when your program ends, which you can ignore.

RELEASE NOTES

```
{*******************************************************************************
*******************************************************************************
***                                                                         ***
***   I B M     S Y S T E M      3 6 0                                      ***
***                                                                         ***
***             GUI version of a simple system 360 model 30                 ***
***                                                                         ***
***             * Supports displacements >999, i.e. 000 to FFF (4095)       ***
***             * Has 16k of main memory, amd maxCoreBytes is this value    ***
***             * Vista win64 and XP win32 as far back as SP1               ***
***                                                                         ***
*******************************************************************************
***                                                                         ***
***   L I N K    E D I T    N O T E S                                       ***
***                                                                         ***
***   B:DDD  displacements are in decimal                                   ***
***          displacements > 999 now handled, inefficiently, by a special   ***
***          separate core array to hold thousands so ddd range             ***
***          is 000 to 4095                                                 ***
***                                                                         ***
***   Core storage is two PC bytes per IBM 360 byte                         ***
***          for character constants they exist as  C.H.A.R.A.C.T.E.R.      ***
***          constants such as A, F, H are saved normally but as decimal    ***
***          Only seen in a core dump is the special array holding the thousands ***
***          for displacements > 999. Remember this is a decimal ASCII 360  ***
***                                                                         ***
***   This sim uses decimal where it possibly can and that plus             ***
***          two for one bytes in core storage make everything              ***
***          simpler most of the time.                                      ***
***                                                                         ***
*******************************************************************************
***                                                                         ***
***   E X E C U T E    P H A S E    N O T E S                                ***
***                                                                         ***
***                                                                         ***
***   SUPERVISOR CALLS IMPLEMENTED                                          ***
***   SVC 1   PRINT CHARS IS DONE BY SVC 0,1 AND R1:DATA R0:SIZE            ***
***   SVC 2   READ 4 SWITCHES                                               ***
***   SVC 3   PRINT ONE BLANK LINE                                          ***
***   SVC 4   GET CONSOLE IN AREA                                           ***
***   SVC 5   TRACE ON OR OFF                                               ***
***   SVC 6   DISPLAY SOME OUTPUT INTO THE CONSOLE AREA                     ***
***   SVC 7   OFFSET PRINTOUT BY THIS MANY BYTES                            ***
***   SVC 14  TERMINATE USER PROGRAM                                        ***
***                                                                         ***
***   MACROS SUPPORTED                                                      ***
***   GETSWITCHES    CALL    SAVEAREA    EOJ  and others, see 360sysMacro.txt ***
***                                                                         ***
***   B:DDD  displacements are in decimal and assembler as well             ***
***          as linkedit detect displacements > 999, and the link phase uses ***
***          another array for thousands in the ddd fild                    ***
***                                                                         ***
***   Core storage is two PC bytes per IBM 360 byte                         ***
***          character constants which are C.H.A.R.A.C.T.E.R                ***
***          other constants such as A, F, H are saved normally             ***
***                                                                         ***
***   This sim uses decimal where it possibly can and that plus             ***
***          two for one bytes in core storage make everything              ***
***          simpler most of the time.                                      ***
***                                                                         ***
***   Code is assumed to be reentrant, namely it cannot                     ***
***          change itself. Actually it can, but it is not                  ***
***          supported mainly because of the HEX and DECIMAL                ***
***          concepts used in this simulator, thus BIT and HEX              ***
***          per se are not properly implemented.                           ***
***                                                                         ***
***   Decimal SS instructions, eg CP and ZAP etc, use one length field      ***
***          and thus both operands must be the same length.                ***
```

```
***                                                                           ***
***   Program with consideration to the above, and it works well.             ***
***                                                                           ***
***   Refer to "LAZARUS-sim360c-notes.doc" for current details and            ***
***          lists of what is and is not supported                            ***
***          check the IBM 360 page of   www.illustratingshadows.com          ***
***                                                                           ***
*********************************************************************************
*********************************************************************************

   ================================ begin  credits ==============================

   Sim360 was inspired by several items. One was the Sim360a and Sim360b written
   in Pascal based on BAL/SX written by the following persons:-

   Stanley A Wileman          1981
   Curt Hall                  1995 2007
   Simon Wheaton-Smith        2009

   Second, on a 370 simulator that ran on the 370 itself, strange you may say, but
   the reason was to provide much better online debugging than MVS provided, and
   third, on the IBM 1401 simulator I wrote in the late 1960s that supported the
   sterling feature, but it had to be object deck compatible. Also, this simulator
   fulfils a fantasy I had about what a 360 would have looked like if it had been
   decimal based as opposed to the much wiser choice of hexadecimal. I had studied
   the Amdahl patent for the 360 in the early 1970s, and it held me in awe.

   So, here is a decimal based IBM 360 which allows 'binary' as in register work
   as well as decimal as in CVD, and character as in UNPK. And this supports a
   start address allowing a control program to be in low core if desired, and it
   supports multiple CSECTs each with USINGs if desired, and the TEST99.TXT program
   for a horizontal dial demonstrates all that. Because of the decimal base rather
   than a hex base system, B-DDD were limited to a displacement of 999 and not 4095,
   and the assembler pass 2 and the linkedit detect this. However, June 24, 2009
   this limit was lifted with a new array for thousands in the displacement.

   Because of all this, the control panel displays registers in decimal.

   This program is open source under the GNU common license terms, you may copy
   it and extend it and so on with the sole provise that these historical notes and
   credits are retained. And while this uses little of Stanley A Wileman's and
   Curt Hall's work, some of their concepts were used and thus their credit should
   be retained.

   Simon Wheaton-Smith
   March 8, 2009
   www.illustratingshadows.com

   ================================ end of credits ==============================


   * Feb  4 2009   basic elements of pass 1 of the assembler are in place
                   most opcodes supported and pseudo opcodes generated to pass 2
                   but operands not generated except for DC and USING
   * Feb  7 2009   symbol table complete and pass 1 partial code outputted
   * Feb 12 2009   pass 1 and pass 2 complete but still need

                   * SS with a label(size)
                   * char limitation is 20 in a DC C'...'
                   * displacements are saved as decimal nnn and thus max 999
                   * displacements > 999 are flagged in assemble and linkedit
                   * need to handle LM and STM operans formats
                   * label+disp  is not implemented
                   * because hex and bit methods are not fully enabled, and
                     because of our hex:char methodology, code should be written
                     as re-entrant, not just serially reusable.

   * Feb 19 2009   Altered character constants from CHARACTER......... internally
                   to C.H.A.R.A.C.T.E.R. which only involved LINK and EXEC
                   and no changes at all to PASS 1 and PASS 2 assembler phases.
   * Feb 19 2009   CR, CH, C, CL works, BC assembles, and BH BE BL BNE is ok
```

```
                in EXEC phase
* Feb 20, 2009  KEY POINT: "IBM binary" in this simulator decimal, and that
                works fine as long as BIT INSTRICTIONS (TM, etc) are not used.
                Binary in coreStorage is actually string, but in registers
                it is longInteger.

                KEY POINT: "IBM Decimal" is saved in this simulator as decimal
                also, however it has a sign "C" eg, as the last low order digit.

                THUS:     4E  CVD  IBM: register binary to core storage decimal
                                   SIMULATOR: BECAUSE OF HOW WE IMPLEMENT
                                         BINARY as decimal
                                         DECIMAL as string decimal plus a sign
                                   actually takes [PC] decimal in a simulated
                                   register to corestorage as string with a sign.

* Feb 21, 2009  DC P added, and PACK and UNPK working, see TEST 11
                BAL r,label     A   AL  added
                BR  added       S       added
* Feb 22, 2009  BCT AND BCTR COMPLETE SEE TEST 14
                BNL BHE and BNH BLE and LTR added along with BZ
                Assembles, links and executes TEST99 as 360sysin.txt which
                is the SIM360A or SIM360B HDIAL program modified to use MR and
                DR and with SVC 1 instead of their PUTM and PUTD, etc.
                multiple USINGs can be used for subroutines and CSECTs
* Feb 23, 2009  SVC 5 added, and a documentation package: LAZARUS-sim360c-notes.pdf
* Feb 24, 2009  CNOP inserts 0700 etc but only if address is on a hwd boundary
* Feb 25, 2009  Available on www.illustratingshadows.com web site as 'SIM360C'
                CLC and MVC added, ZAP CP added (as with PACK, this assumes L1=L2)
* Feb 26, 2009  Better use of color at startup and on B-DDD error in LINKEDIT
* Feb 27, 2009  Source file to assemble with IPL 1 is now named in the console
                input area, default is 360sysin.txt but it can be overridden
                MVI label,X'xx'    and   MVI   label,C'c' added, and if hex
                    then nibbles are xx pair, if char then nibbles are C. with
                    the period '.' indicating this was a character and not hex
                CLI label,X'xx'    and   CLI   labelC'c'  added and same
                    nibble notes at MVI
* Mar 1,  2009  STC and IC added, however it does assume the inserted character
                is numeric as all registers are DECIMAL NUMERIC in this sim. TEST21.
                AH and SH and MH added and M and SL, see TEST22.
                Advisory if any instruction is not true 370 compatible
                EQU added but only for *, and not for a label
                Message cleanup on compatibility issues, check HDIAL.ASM (TEST99)
                for notes. For example, DC H and F are stored as packed no sign,
                so CVD and UNPK work correctly, but you can cheat with just UNPK
                but you will lose compatibility.
* Mar 3,  2009  Just as this uses decimal for internal usage (registers, DC H and F),
                it also uses the ASCII as opposed to EBCDIC collating sequence.
                LM and STM added for r1<, -, > r2 see TEST25
* Mar 4,  2009  Panel header clarifies that while ASCII is used vs EBCDIC, and while
                registers are internally decimal, correct code works, eg
                       L     1,F'1234'
                       CVD   1,DWD
                       UNPK  xxx,DWD
                F1234  DC    F'1234'
                works even though the DC F is stored as decimal. As for ASCII vs
                EBCDIC the main issue is numeric 0 is lower than letter A, not
                higher than letter Z. And since the assembler pass 2 listing
                highlights compatibility issues, it is not a big deal. And allowing
                address constants and CSECTs with their own USINGs, as in TEST99,
                the system is very functional.
* Mar 7,  2009  Elementary MACRO facility added as pass 0, rules are in the file
                called "360sysMacro.txt" as supplied. These are very simply macros
                with one parametric name label and one parameteric operand only
* Mar 8,  2009  MNEM and associated tables simplified. SVCs now read 1052 and
                switches directly, INTERRUPT 1 and 2 no longer used. Program
                STARTADDRIS recommended to be 400 since we set MFT/MVT register
                conventions and that places an SVC 14 above control program
                old and new psws, although 200 works. DO NOT USE 0 because
                the system will object since the initial save area and SVC 14 used
                for MFT/MVT conventions will be upset. TEST27 and TEST28 show DOS
```

```
                  and MFT/MVT conventions working. Clean up code for consistency.
                  SYSTEM RESET and START buttons added for future use.
                  SVC 6 added to send text to the 1052.
   * Mar 9,  2009  INTERRUPT no longer triggers a core dump, it sets a message only.
                  A core dump now happens with SYSTEM RESET, START, START which in
                  days of old was a PSW RESTART on the 2030, but if the rightmost
                  four switches were 090E then a core dump was taken.
                  Added SVC 7 to allow print line offset.
   * Mar 11, 2009  Trivial stuff, and refinements to the sundial H program. And name
                  for linkedit phase is also user inputable, and radio buttons as
                  markers as a visual aid.
   * Mar 15, 2009  Resequenced buttons, so that POWER ON is first, then IPL which
                  enables the "software", and INTERRUPT with 1,2,3 to assemble,
                  link, and execute is next (as opposed to using IPL 1,2,3).
                  hdial.asm (test99.txt) as well as vdial.asm (test98.txt) added,
                  also hdial/test99 shows how to make more code addressable by
                  doing a poor technique used back in BPS and BOS days.
                  Misc refinements to the panel. Corrected loop detect in LOG.
   * Mar 24, 2009  Changes from Tlabel to TstaticText on PSW, instructions, and
                  the registers, so they flash and look pretty. Same in the
                  1401 simulator.
   * Apr 11, 2009  Cosmetic changes.
   * Apr 20, 2009  viewASM and viewDUMP as well as viewLOG and viewPRINT added.
   * Apr 26, 2009  Compiled on VISTA 64 bit.
   * Apr 28, 2009  Common version win32 works on XP and Vista win 64

   ////////////////////////////////////////////////////////////////////////////
   * 20090624      FOR DISPLACEMENTS > 999
   *               coreStorageK: This is a blank or 0, except when a B-DDD DDD
   *                             has 1,2,3,4000
   *               baseDispK:    also is a 12 byte constant in IFETCH logic for
   *                             big displacements
   *               Two lines of code are commonly used for displacements > 999
   *               disp := disp + 1000*strToInt( baseDispK[6] );
   *               disp := disp + 1000*strToInt( baseDispK[10] );
   *               Linkedit tests for ddd>4095 and updates CPU panel and log if found
   ////////////////////////////////////////////////////////////////////////////
   * June 28, 2009 Cosmetic changes
   * June 30, 2009 Core dump only shows 1000s if a displacement was > 999
   * July 1,  2009 maxCoreBytes is user memory area, and the arrays are twice that,
                  originally it was the other way around in the Version 1 sim.
}
```

SIMULATOR VERSIONS

sim360a	used the Bloodshed 8mb free Pasal compiler and was limited in what it supported but worked well. The 8mb compiler is short on documentation but uses modern Windows panes.
sim360b	used the Free Pascal 28mb compiler and was otherwise the same as sim60a. The 28mb compier has good documentation but is hard to use in that it styles itself after DOS windows.
sim360c version 1	only allowed displacements in B-DDDs o be 999, had 4k of user memory, and is frozen. First Lazarus version.
sim360c version 2	allows full displacements of 0000 to 4095, has 16k of memory which you may expand (maxCoreBytes), and the core dump also shows full displacements.
Open Source	sim360c is open source.

SOURCE CODE FOR A HORIZONTAL SUNDIAL PROGRAM THAT RUNS ON SIM360C

```
* ********************************************************************* *
* *      HORIZONTAL DIAL FOR THE IBM 360                                * *
* ********************************************************************* *
*
* TEST99    SUNDIAL PROGRAM ALSO SAVED AS HDIAL.ASM
*
* THIS IS FOR SIM360C - OTHER VERSIONS FOR OTHE SIMULATORS
*
* NOTE THE COMPATIBILITY NOTES THAT SIM360C ADDS TO THE END OF
*          A LISTING. AND SEE SOME NOTES IN THIS CODE ALSO.
*
* WWW.ILLUSTRATINGSHADOWS.COM      SIMON WHEATON-SMITH   MARCH 11, 2009
*
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*
*              HORIZONTAL SUNDIAL PROGRAM ON THE IBM SYSTEM 360
*
*                    LATITUDE IS        033
*                    SIN(LAT) IS     0.0544
*
*                    LONGITUDE DIF      003
*                    IN MINUTES         012
*
*              MORNING HOURS - - - - - - - - - - - - - -
*
*                    HOUR FROM NOON [HA]  HOUR LINE ANGLE
*
*                              005 [78]             069
*                              004 [63]             047
*                              003 [48]             032
*                              002 [33]             020
*                              001 [18]             010
*                              000 [03]             002
*
*              AFTERNOON HOURS - - - - - - - - - - - - - -
*
*                    HOUR FROM NOON [HA]  HOUR LINE ANGLE
*
*                              001 [12]             007
*                              002 [27]             016
*                              003 [42]             027
*                              004 [57]             040
*                              005 [72]             060
*
*
*
*              ========================================
*
*              THIS DIAL IS WEST OF MERIDIAN
*              IF EAST OF MERIDIAN SWITCH AM FOR PM
*
*              CHECK   WWW.ILLUSTRATINGSHADOWS.COM
*              FOR THE LATEST PROGRAMS
*
*              DOWNLOAD    MICROSHADOWS.PDF    FROM
*              THE WEBSITE FOR TIPS AND FAQS
*
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*
*
* SIN COS TAN ATN  ARE INVOKED WITH 'L 15' AND 'BALR 14,45'
*
* USES   SVC 2 TO READ SWITCHES
*        FORMAT OF THOSE FOUR SWITCHES IS     A B C D
*        A*10+B => LATITUDE
*        C*10+D => LONGITUDE CORRECTION IF ANY
*        BUT DONE WITH A MACRO 'GETSWITCHES'
```

```
*
* MACROS GETSW    GETS SW1-4
*        EOJ      DOES SVC 14
*        CALL     WITH THE SUBROUTINE ADDRESS INLINE
*        CALLI    CALL BUT THE PARAMETER BEING AN ADDRESS CONSTANT
*
*        FOR NO GOOD REASON, AM HOURS USES MACROS, PM USES INLINE CODE
*
HDIALPGM CSECT ,                       MAIN PROGRAM
         USING *,12                    TELL ASSEMBLER
         LR    12,15                   SET R12 AS BASE AS R15 PRESET BY SIM
*
*-------------------------------------------------------------------------
*
         GETSW                         GETS SW 1-4
*                                      MUST BE SET BEFORE EXECUTE
         LR    5,1                     SEE IF ANY
         AR    5,2                     SWITCHES
         AR    5,3                     WERE ENTERED
         AR    5,4                     BEFORE IPL CODE 3
         LTR   5,5                     TEST R5
         BZ    BEGIN                   IF ZERO THEN
*
*              SWITCHES
*
         SR    0,0                     CLEAR R0 FOR MULTIPLY
*                                      R1 IS SWITCH 1 IS LATITUDE
         LA    15,10                   MULTIPLIER IS 10
         MR    0,15                    IS THUS TIMES 10
         AR    1,2                     ADD UNIT OF LATITUDE
         ST    1,LATITUDE              ELSE GOOD DATA SO SAVE
*
         SR    0,0                     SINCE WE GOT A LAT, ZERO OUT LNG DIF
         ST    0,LONGCORR              OBVIOUSLY
*
         SR    0,0                     CLEAR R0 FOR MULTIPLY
         LR    1,3                     R1 IS SWITCH 3 IS LONGITUDE DIFF
         LA    15,10                   MULTIPLIER IS 10
         MR    0,15                    IS THUS TIMES 10
         AR    1,4                     ADD UNIT OF LATITUDE
         ST    1,LONGCORR              ELSE GOOD DATA SO SAVE
*-------------------------------------------------------------------------
*
*
*
BEGIN    SVC   3                       BLANK LINE
         SVC   3
         OFSET 15                      SHIFT PRINT LINE BY 15 BYTES
*
*              HORIZONTAL SUNDIAL PROGRAM ON THE IBM SYSTEM 360
*
         L     1,M2HDRADR              TEXT
         LA    0,72                    SIZE
         SVC   01                      PRINT TEXT
         SVC   3                       BLANK LINE
         L     1,M0HDRADR              TEXT
         LA    0,72                    SIZE
         SVC   1                       PRINT TEXT
         SVC   3
         SVC   3
         SVC   3
         OFSET 30                      SHIFT PRINT LINE BY 30 BYTES
*
*              LAT IS  XXX    SIN(LAT) IS
*
         L     1,LATITUDE              GET LATITUDE BINARY (PACKED DECIMAL NO SIGN)
         CVD   1,DWD                   AS PACKED DECIMAL WITH A SIGN
         LA    2,DWD                   SET R2 TO DWD
*              LA    2,6(2)                SET R2 TO  NN NC
*              UNPK  M3HDRLAT,0(2)         PRINTABLE CHARACTERS (STRING)
         UNPK  M3HDRLAT,6(2)           PRINTABLE CHARACTERS (STRING)
```

```
*
         L     1,LATITUDE              SET R1 EQ LATITUDE
         CALLI SINADR                  GET SIN - R1 IN = LAT, OUT = SIN
         ST    1,SINLAT                SAVE SIN LAT
*                                      THE FOLLOWING 3 WORK ON ALL 360 SIMS
         CVD   1,DWD                   CONVERT IT
         LA    2,DWD                   GET ADDRESS OF DWD
         MVC   SINLATH,6(2)            FROM DWD+6
         UNPK  M3HDRSIN,SINLATH
*
         LA    1,M3HDR                 TEXT
         LA    0,31                    SIZE
         SVC   1                       PRINT
         LA    1,M3HDR1                TEXT
         LA    0,31                    SIZE
         SVC   1                       PRINT
         SVC   3                       BLANK LINE
*
*              LONGITUDE CORR IS XXX    IN MINUTES    XXX
*
         L     1,LONGCORR              GET IN FULLWORD
         CVD   1,DWD                   IN DECIMAL
         LA    2,DWD                   SET R2 TO DWD
         UNPK  M4HDRLNG,6(2)           PRINTABLE CHARACTERS (STRING)
*
         SR    0,0                     R0, R1 MULTIPLICAND
         L     1,LONGCORR              GET LONGITUDE CORRECTION
         LA    15,4                    R15 MULTIPLIER
         MR    0,15                    MULTIPLY
         ST    1,LONGMINS              SAVE RESULT
         CVD   1,DWD                   IN DECIMAL
         LA    2,DWD                   SET R2 TO DWD
         UNPK  M4HDRMIN,6(2)           PRINTABLE CHARACTERS (STRING)
*
         LA    1,M4HDR                 STATE LONGITUDE CORR
         LA    0,29                    SIZE
         SVC   1                       PRINT
         LA    1,M4HDR1                STATE LONGITUDE CORR
         LA    0,29                    SIZE
         SVC   1                       PRINT
         SVC   3                       BLANK LINE
*
*
*        D O    M O R N I N G    H O U R S
*
*
         L     1,M5HDRADR              TEXT
         LA    0,43                    SIZE
         SVC   1                       PRINT
         SVC   3                       BLANK LINE
         L     1,M6HDR1AD              TEXT
         LA    0,43                    SIZE
         SVC   1                       PRINT
         SVC   3                       BLANK LINE
*
*
*----------------------------------------------------------------*
*        SWITCH TO A NEW BASE TO MAKE THIS ADDRESSABILITY LAST   *
*        NOTE.. THIS PROGRAM MAIN CSECT FIT WITHIN ONE USING     *
*               BUT DUE TO THE SEQUENTIAL MONOLITHIC CODE, THIS  *
*               TECHNIQUE WORKS TO EXTEND THE ADDRESSABILITY     *
*               BACK IN THE DAYS OF BOS AND BPS, THIS WAS USED.  *
*        NOTE.. THIS IS A VERY POOR TECHNIQUE, BUT BACK IN THE   *
*               OLD DAYS WHEN EVERY BIT COUNTED, THIS WAS DONE.  *
*                                                                *
*---         BALR  12,0                    NEW BASE             *
*---         USING *,12                    SAY OK               *
*                                                                *
*----------------------------------------------------------------*
*
*        AM    HOUR ANGLE LOOP FOR THE HOURS
```

```
*
         LA    6,5                 R6 SET R2 TO 5 HOURS FROM NOON
*
HOURLOOP LA    15,15                  R15 IS 15 DEGREES PER HOUR
         SR    0,0                    R0, R1 MULTIPLICAND
         LR    1,6                 R6 R1 IS HOURS FROM NOON
         MR    0,15                R1 WILL BE HRA WHICH IS HR * 15
         A     1,LONGCORR             R1 NOW CORRECTED FOR LONGITUDE
         C     1,HDIAL90              IS ANGLE TOO HIGH
         BH    HOURNEXT               SKIP IF SO
*
         C     6,HDIALZRO             IS THIS NOON
         BH    NOTNOON1               NO
         C     1,HDIALZRO             IS HR ANLG PLUS LONG CORR +VE
         BL    HOURNEXT               NEGATIVE LCLHA SO SKIP HLNA
*
*              HOURS FROM NOON THEN HOUR LINE ANGLE
*
NOTNOON1 ST    6,HOURWORK          R6 SAVE IN A WORK AREA
         CVD   6,DWD                  NOW PACKED DECIMAL
         LA    2,DWD                  POINT TO DWD
         UNPK  M6HDRHRS,6(2)          DECIMAL PRINT HOURS
*----------------------
         CVD   1,DWD                  SAVE HOUR ANGLE OF THE SUN
         LA    2,DWD                  INTO THE
         UNPK  DC3,6(2)               PRINT
         LA    2,DC3                  LINE
         MVC   M6HDRHRA,1(2)          FYI ONLY
*-----------------------
         ST    1,HOURANGL             SET R1 EQ SUNS HOUR ANGLE
         CALLI TANADR                 GET TAN OF HOUR ANGLE OF SUN
         ST    1,TANHOUR              SAVE TAN HOUR ANGLE
*
         LA    0,0                    CLEAR R0 FOR MULTIPLY
         L     1,TANHOUR              R1 TO TAN HOUR ANGLE
         L     15,SINLAT              R0 TO SIN OF LATITUDE
         MR    0,15                   GET PRODUCT
         ST    1,TANHLA               RESULT IS TAN OF HR LINE ANGLE
*
         SR    0,0                    CLEAR FOR DR INSTRUCTION
         L     1,TANHLA               GET TAN HOUR LINE ANGLE
         L     14,F1000               DIV BY 1000 AS SIN AND TAN
*                                     ARE 1000 TIMES VALUES SO NOW
*                                     TANHLA IS 1,000,000 TIMES
         DR    0,14                   DIVIDE TANHLA BY 1000
         ST    1,TANHLA               SAVE IT
*
         L     1,TANHLA               GET PROPER TAN HLA (*1000 SO OK)
         CALLI ATNADR                 GET ANGLE OF THIS TAN, R1 IN AND OUT
*
         ST    1,HRLNANGL             SAVE HOUR LINE ANGLE IN DEGREES
         CVD   1,DWD                  NOW PACKED DECIMAL
         LA    2,DWD                  POINT TO DWD
         UNPK  M6HDRHLA,6(2)          DECIMAL PRINT HOURS
         LA    1,M6HDR                TEXT
         LA    0,42                   SIZE
*
         SVC   1                      PRINT IT
*
HOURNEXT S     6,HDIALONE          R6  SUBTRACT FROM HOURS FROM NOON
         C     6,HDIALZRO          R6  ARE THERE MORE TO GO STILL
         BNL   HOURLOOP               REPEAT
         SVC   3                      BLANK LINE
*
*
*        N O W    D O    A F T E R N O O N    H O U R S
*
         L     1,M7HDRADR             TEXT
         LA    0,43                   SIZE
         SVC   1                      PRINT
         SVC   3                      BLANK LINE
```

```
         L     1,M6HDR1AD              TEXT
         LA    0,43                    SIZE
         SVC   1                       PRINT
         SVC   3                       BLANK LINE
*
*        PM    HOUR ANGLE LOOP FOR THE HOURS
*
         LA    6,0                 R6 SET R2 TO 0 HOURS FROM NOON
*
HOURAGIN LA    15,15                   R0 IS 15 DEGREES PER HOUR
         SR    0,0                     CLEAR R0
         LR    1,6                 R6 R1 IS HOURS FROM NOON
         MR    0,15                R1 WILL BE HRA WHICH IS HR * 15
         S     1,LONGCORR              R1 NOW CORRECTED FOR LONGITUDE
         C     1,HDIAL90               IS ANGLE TOO HIGH
         BH    HOURMORE                SKIP IF SO
*
         C     6,HDIALZRO              IS THIS NOON
         BH    NOTNOON2                NO
         C     1,HDIALZRO              IS HR ANLG PLUS LONG CORR +VE
         BL    HOURMORE                NEGATIVE LCLHA SO SKIP HLNA
*
NOTNOON2 ST    6,HOURWORK          R6 SAVE IN A WORK AREA
         CVD   6,DWD                   NOW PACKED DECIMAL
         LA    2,DWD                   POINT TO DWD
         UNPK  M6HDRHRS,6(2)           DECIMAL PRINT HOURS
*----------------------
         CVD   1,DWD                   SAVE HOUR ANGLE OF THE SUN
         LA    2,DWD                   INTO THE
         UNPK  DC3,6(2)                PRINT
         LA    2,DC3                   LINE
         MVC   M6HDRHRA,1(2)           FYI ONLY
*-----------------------
         ST    1,HOURANGL              SET R1 EQ SUNS HOUR ANGLE
         L     15,TANADR               GET ADDRESS OF TAN
         BALR  14,15                   GET TAN RETURNED IN R1
         ST    1,TANHOUR               SAVE TAN HOUR ANGLE
*
         SR    0,0                     CLEAR R0 FOR MULTIPLY
         L     1,TANHOUR               R1 TO TAN HOUR ANGLE
         L     15,SINLAT               R0 TO SIN OF LATITUDE
         MR    0,15                    GET PRODUCT
         ST    1,TANHLA                RESULT IS TAN OF HR LINE ANGLE
*
         SR    0,0                     CLEAR FOR DR INSTRUCTION
         L     1,TANHLA                GET TAN HOUR LINE ANGLE
         L     14,F1000                DIV BY 1000 AS SIN AND TAN
*                                      ARE 1000 TIMES VALUES SO NOW
*                                      TANHLA IS 1,000,000 TIMES
         DR    0,14                    DIVIDE TANHLA BY 1000
         ST    1,TANHLA                SAVE IT
*
         L     1,TANHLA                GET PROPER TAN HLA (*1000 SO OK)
         L     15,ATNADR               GET ATN SUBROUTINE
         BALR  14,15                   GET ANGLE OF IT
*
         ST    1,HRLNANGL              SAVE HOUR LINE ANGLE IN DEGREES
         CVD   1,DWD                   NOW PACKED DECIMAL
         LA    2,DWD                   POINT TO DWD
         UNPK  M6HDRHLA,6(2)           DECIMAL PRINT HOURS
         LA    1,M6HDR                 TEXT
         LA    0,42                    SIZE
*
         SVC   1                       PRINT IT
*
HOURMORE LA    6,1(6)              R6  ADD TO HOURS FROM NOON
         C     6,HDIALSIX          R6  ARE THERE MORE TO GO STILL
         BL    HOURAGIN                REPEAT
*
*             *=============================================*
*
```

```
         CALL  ENDNOTES                END NOTES
*
         EOJ                           TIME TO SHUT DOWN
*
* ************************************************************* *
* * INPUT PARAMETERS                                           * *
* *                               ***  ARE INPUT PARAMETERS     * *
* *                               *                             * *
LATITUDE DC    F'33'              ***  LATITUDE 33              * *
SINLAT   DC    F'0'               *    SIN OF LATITUDE  FWD     * *
SINLATH  DC    H'0'               *    SIN OF LATITUDE  HWD     * *
COSLAT   DC    F'0'               *    COS OF LATITUDE          * *
*                                 *                             * *
LONGCORR DC    F'3'               ***  WEST 3 DEGREES OF MERIDIAN  * *
LONGMINS DC    F'0'                    EQUIVALENT MINUTES       * *
* *                                                             * *
* ************************************************************* *
*
* ************************************************************* *
* * WORK AREAS                                                 * *
* *                                                           * *
HOURANGL DC    F'0'                    AN HOUR ANGLE            * *
HRLNANGL DC    F'0'                    AN HOUR LINE ANGLE       * *
F1000    DC    F'1000'                 CONSTANT FOR DIVIDE ETC  * *
TANHOUR  DC    F'0'                    TAN OF HOUR ANGLE        * *
HOURWORK DC    F'0'                    A HUMBLE WORK AREA       * *
TANHLA   DC    F'0'                    TAN OF HOUR LINE ANGLE   * *
HDIALONE DC    F'1'                    1 FOR LOOPS AS NO BCT/BCTR  * *
HDIALZRO DC    F'0'                    0 FOR TESTS              * *
HDIALSIX DC    F'6'                    6 FOR TESTS              * *
HDIAL90  DC    F'90'                   LIMIT OF 90 DEGREES      * *
DC3      DC    C'VVV'                  INTERMEDIATE AREA        * *
         CNOP  0,8                                              * *
DWD      DC    D'0'                                             * *
* ************************************************************* *
*
*
*
*
*              ADDRESSES OF SUBROUTINES
*
*
SINADR   DC    A(SINSUBR)              ADDRESS OF SIN SUBROUTINE
COSADR   DC    A(COSSUBR)              ADDRESS OF COS SUBROUTINE
TANADR   DC    A(TANSUBR)              ADDRESS OF TAN SUBROUTINE
ATNADR   DC    A(ATNSUBR)              ADDRESS OF ARCTAN SUBROUTINE
*
*
*              ADDRESSES OF HEADERS
*
*
M0HDRADR DC    A(M1HDR)
M2HDRADR DC    A(M2HDR)                ADDRESS OF HEADER *H-DIAL
M5HDRADR DC    A(M5HDR)                ADDRESS OF HEADER *AM~NOON
M6HDR1AD DC    A(M6HDR1)               ADDRESS OF HRS NOON HR LN ANGLE
M7HDRADR DC    A(M7HDR)                ADDRESS OF HEADER *PM~NOON
*
*
*
*              HEADERS THEMSELVES
*
*
*        31 BYTE LINES ABOUT LATITUDE
*
M3HDR    DC    C'      LATITUDE IS   '
         DC    C'     '
M3HDRLAT DC    C'XXX'                  UNPK CAME FROM CVD
         DC    C'   '
*
M3HDR1   DC    C'      SIN(LAT) IS   '
         DC    C'   0.'
```

```
M3HDRSIN DC    C'XXX'                 UNPK CAME FROM HWD
         DC    C'    '
*
*        29 BYTE LINE ABOUT LONGITUDE
*
M4HDR    DC    C'      LONGITUDE DIF '
         DC    C'     '
M4HDRLNG DC    C'XXX'                 UNPK FROM CVD
         DC    C' '
*
M4HDR1   DC    C'      IN MINUTES    '
         DC    C'     '
M4HDRMIN DC    C'XXX'                 UNPK FROM HWD
         DC    C' '
*
*
*
*        42 BYTE HEADER
*
M6HDR    DC    C'                 '
M6HDRHRS DC    C'XXX'
         DC    C' ['
M6HDRHRA DC    C'XX'
         DC    C']             '
M6HDRHLA DC    C'XXX'
*
*
*
*
*
*
* ********************************************************************* *
* ********************************************************************* *
* *                                                                   * *
* *      C O N S T A N T S    T H A T    A R E    N O T                * *
* *      BASE-DISPLACEMENT    A D D R E S S A B L E                   * *
* *                                                                   * *
* ********************************************************************* *
* ********************************************************************* *
*
*
*
*
*
         CNOP  0,8
HEADERTX CSECT ,
*
*
*        72 BYTE HEADER
*
M1HDR    DC    C'==================='
         DC    C'==================='
         DC    C'==================='
         DC    C'==================='
*
*
*        48 BYTE HEADER BUT UP TO 132 IS OK
*
M2HDR    DC    C'HORIZONTAL SUNDIAL'
         DC    C' PROGRAM ON THE IB'
         DC    C'M SYSTEM 360  - - '
         DC    C'-  S WHEATON-SMITH'
*
*        43 BYTE HEADER
*
M5HDR    DC    C'MORNING HOURS - - - '
         DC    C'- - - - - - - - - - '
         DC    C'- -'
*
*        43 BYTE HEADER
```

```
*
M6HDR1   DC    C'      HOUR FROM NOON'
         DC    C' [HA]  HOUR LINE ANG'
         DC    C'LE '
*
*        43 BYTE HEADER
*
M7HDR    DC    C'AFTERNOON HOURS - - '
         DC    C'- - - - - - - - - - '
         DC    C'- -'
*
*        72 BYTE HEADERS
*
M9HDR    DC    C'THIS DIAL IS WEST '
         DC    C'OF MERIDIAN. IF EA'
         DC    C'ST OF MERIDIAN THE'
         DC    C'N SWITCH AM FOR PM'
*
MAHDR    DC    C'CHECK    WWW.ILLUS'
         DC    C'TRATINGSHADOWS.COM'
         DC    C' FOR THE LATEST PR'
         DC    C'OGRAMS            '
*
*
MBHDR    DC    C'DOWNLOAD    MICRO-'
         DC    C'SHADOWS.PDF   FROM'
         DC    C' THE WEBSITE FOR T'
         DC    C'IPS AND FAQS      '
*
*
*
*
*
* ********************************************************************* *
* ********************************************************************* *
* *                                                                   * *
* *      T R I G O N O M E T R Y    A N D    M A T H    C O D E       * *
* *                                                                   * *
* ********************************************************************* *
* ********************************************************************* *
*
*
* MLT    IS DONE BY 'MR' INSTRUCTION HERE, FOR THE MLT SUBROUTINE
*        SEE THE OTHER IBM 360 SIMULATOR'S H-DIAL PROGRAM
*
* DIV    IS DONE BY 'DR' INSTRUCTION HERE, FOR THE DIV SUBROUTINE
*        SEE THE OTHER IBM 360 SIMULATOR'S H-DIAL PROGRAM
*
* SIN    INVOKED BY BALR 14,45
* COS    INVOKED BY BALR 14,45
* TAN    INVOKED BY BALR 14,45
* ATN    INVOKED BY BALR 14,45
*
*
* ********************************************************************* *
*        BEGIN SIN SUBROUTINE          TESTS OK JAN 18 2009         * *
* ********************************************************************* *
*        INPUT       R1 EQUALS THE NUMBER WE WANT SIN OF             * *
*        OUTPUT      R1 EQUALS THE SIN OF THE INPUT PARAMETER        * *
*        USES        R2 AS A WORKING REGISTER                        * *
* ********************************************************************* *
         CNOP 0,8
SINSUBR  CSECT ,
         USING *,15                    CALLED WITH BALR
SIN      ST    2,SINWORK               SAVE WORK REGISTER
         AR    1,1                     ANGLE IS NOW ANGLE * 2
         AR    1,1                     ANGLE IS NOW ANGLE * 4
         L     2,SIN00ADR              R2 IS SIN TABLE OF FULL WORDS
         AR    2,1                     R2 IS NOW OUR ENTRY
         L     1,0(2)                  R1 NOW IS SIN OF ANGLE
         L     2,SINWORK               RELOAD WORK REGISTER
```

```
         BR    14                    RETURN
*
         CNOP  0,4
SINWORK  DC    F'0'                  SAVED WORKING REGISTER
SIN00ADR DC    A(SINTABLE)           ADDRESS OF TABLE
*
*
*
* ********************************************************************* *
*        BEGIN COS SUBROUTINE          TESTS OK JAN 19 2009        * *
* ********************************************************************* *
*        INPUT       R1 EQUALS THE NUMBER WE WANT COS OF           * *
*        OUTPUT      R1 EQUALS THE COS OF THE INPUT PARAMETER      * *
*        USES        R2 AS A WORKING REGISTER                      * *
* ********************************************************************* *
         CNOP 0,8
COSSUBR  CSECT ,
         USING *,15
COS      ST    2,COSWORK             SAVE WORK REGISTER
         LA    2,90                  PLACE 90 IN A REGISTER
         SR    2,1                   RS IS NOW 90-INPUT ANGLE
         AR    2,2                   ANGLE IS NOW (90-ANGLE) * 2
         AR    2,2                   ANGLE IS NOW (90-ANGLE) * 4
         L     1,COSINADR            R1 IS SIN TABLE OF FULL WORDS
         AR    2,1                   R2 IS NOW OUR ENTRY
         L     1,0(2)                R1 NOW IS SIN OF ANGLE
         L     2,COSWORK             RELOAD WORK REGISTER
         BR    14                    RETURN
*
         CNOP  0,4
COSWORK  DC    F'0'                  SAVED WORKING REGISTER
COSINADR DC    A(SINTABLE)           ADDRESS OF SIN TABLE
*
*
*
* ********************************************************************* *
*        BEGIN TAN SUBROUTINE          TESTS OK JAN 18 2009        * *
* ********************************************************************* *
*        INPUT       R1 EQUALS THE NUMBER WE WANT TAN OF           * *
*        OUTPUT      R1 EQUALS THE TAN OF THE INPUT PARAMETER      * *
*        USES        R2 AS A WORKING REGISTER                      * *
* ********************************************************************* *
         CNOP 0,8
TANSUBR  CSECT ,
         USING *,15
TAN      ST    2,TANWORK             SAVE WORK REGISTER
         AR    1,1                   ANGLE IS NOW ANGLE * 2
         AR    1,1                   ANGLE IS NOW ANGLE * 4
         L     2,TAN00ADR            R2 IS TAN TABLE OF FULL WORDS
         AR    2,1                   R2 IS NOW OUR ENTRY
         L     1,0(2)                R1 NOW IS TAN OF ANGLE
         L     2,TANWORK             RELOAD WORK REGISTER
         BR    14                    RETURN
*
         CNOP  0,4
TANWORK  DC    F'0'                  SAVED WORKING REGISTER
TAN00ADR DC    A(TANTABLE)           ADDRESS OF TAN TABLE
*
*
*
*
* ********************************************************************* *
*        BEGIN ATN SUBROUTINE          TESTS OK JAN 19 2009        * *
* ********************************************************************* *
*        INPUT       R1 EQUALS A TAN WE WANT THE ANGLE OF          * *
*        OUTPUT      R1 EQUALS THE ANGLE OF THE INPUT TAN PARAMETER* *
*        USES        WORKING REGISTERS                             * *
*                       R2 IS ANGLE SO FAR                         * *
*                       R3 IS ENTRY IN TAN TABLE                   * *
* ********************************************************************* *
*
```

```
ATNSUBR  CSECT ,
         USING *,15
ATN      ST    2,ATNWORK2              SAVE WORK REGISTERS
         ST    3,ATNWORK3              FOR LATER
*              SHOULD USE STM/LM BUT SIM NOT DOING IT YET
         LA    2,0                R2   IS RESULTING ANGLE
         L     3,ATN00ADR         R3   IS TAN TABLE OF FULL WORDS
ATNLOOP  ST    2,ATNANGL               SAVE R2 ANGLE SO FAR IN ATNANGL
         C     2,ATN89                 IS RESULTING ANGLE 90 YET
         BH    ATNEXIT9                EXIT WITH R2 AT 90
         C     1,0(3)                  COMPARE INPUT TO TABLE CONTENT
         BL    ATNEXIT                 INPUT LOW SO EXIT
         BE    ATNEXIT                 INPUT EQUAL SO EXIT
*              INPUT IS LOWER THAN TABLE SO TRY AGAIN
ATNNEXT  LA    3,4(3)             R3   ADD 4 TO TABLE ENTRY FOR NEXT
         LA    2,1(2)             R2   ADD 1 TO CURRENT ANGLE
         B     ATNLOOP                 AND RRY AGAIN
*
ATNEXIT9 LA    1,90                    SET RESULT TO 90
         ST    1,ATNANGL               SAVE IT THEN EXIT AS BELOW
ATNEXIT  L     2,ATNWORK2              RELOAD WORK REGISTERS
         L     3,ATNWORK3              TO STATUS QUO ANTE
*              SHOULD USE STM/LM BUT SIM NOT DOING IT YET
         L     1,ATNANGL               LOAD ANGLE FOR RETURN
         BR    14                      RETURN
*
         CNOP  0,4
ATNWORK2 DC    F'0'                    SAVED WORKING REGISTERS
ATNWORK3 DC    F'0'
ATNANGL  DC    F'0'                    ANGLE THAT RESULTS FROM TAN
ATN89    DC    F'89'                   IF HIGHER THAN ANGLE THEN EXIT
ATN00ADR DC    A(TANTABLE)             ADDRESS OF TAN TABLE
*
*
*
*
*
*
* - - T A B L E S    F O R    S I N    A N D    C O S I N E - -
*
*
         CNOP  0,8
SINTABLE CSECT ,
SIN00    DC    F'0000'                 EACH ENTRY IS 1000 * SIN
         DC    F'0017'
         DC    F'0034'
         DC    F'0052'
         DC    F'0069'
         DC    F'0087'
         DC    F'0104'
. . .
. . .
         DC    F'0996'
         DC    F'0997'
         DC    F'0998'
         DC    F'0999'
         DC    F'0999'
         DC    F'1000'
* ************************************************************* *
*        END SIN FUNCTION SUBROUTINE                            * *
* ************************************************************* *
*
*
* - - T A B L E S    F O R    T A N    A N D    A R C T A N - - - -
*
*
         CNOP  0,8
TANTABLE CSECT ,
TAN00    DC    F'000000'             EACH ENTRY IS 1000 * TAN
         DC    F'000017'
         DC    F'000034'
```

```
         DC    F'000052'
         DC    F'000069'
         DC    F'000087'
. . .
. . .
         DC    F'011430'
         DC    F'014300'
         DC    F'019081'
         DC    F'028636'
         DC    F'057289'
         DC    F'999999'
*
*
* ************************************************************* *
*        FINAL FOOT NOTES                                       * *
* ************************************************************* *
*        HERE TO SAVE MEMORY IN MAIN CSECT                      * *
* ************************************************************* *
*
ENDNOTES CSECT ,                       ENTER
         USING *,15                    SET USING
         SVC   3                       BLANK LINE
         SVC   3                       BLANK LINE
         SVC   3                       BLANK LINE
         OFSET 15                      SHIFT PRINT LINE BY 15 BYTES
         L     1,M1HDRADR              TEXT
         LA    0,72                    SIZE
         SVC   1                       PRINT TEXT
         SVC   3                       BLANK LINE
         LA    0,72                    R0 IS SIZE
         L     1,M9HDRADR              R1 IS TEST
         SVC   1                       ADVISE ON EAST/WEST DIALS
         SVC   3                       BLANK LINE
*
         LA    0,72                    R0 IS SIZE
         L     1,MAHDRADR              R1 IS TEST
         SVC   1                       ADVISE ON WEB SITE
         SVC   3                       BLANK LINE
*
         LA    0,72                    R0 IS SIZE
         L     1,MBHDRADR              R1 IS TEST
         SVC   1                       ADVISE ON NOTES
         SVC   3                       ON WEB SITE
         BR    14                      EXIT
*
M1HDRADR DC    A(M1HDR)                ADDRESS OF HEADER *=====*
M9HDRADR DC    A(M9HDR)                ADVISOR
MAHDRADR DC    A(MAHDR)                ADVISOR
MBHDRADR DC    A(MBHDR)                ADVISOR
*
         END
```