INSTTIME -- Mel's instruction timer

Purpose

To time instructions being executed in the Z390 enulator.
To provide useful statistics for release comparison and extrapolation.

From INSTTIME V7, a new feature was added to time a few SVC services,
see the end of INSTTIME.INS for all the variations supported.
The instruction displayed is the SVC used.

Services currently timed...

```
   GETMAIN  ( 3 forms)
   FREEMAIN ( 1 form )
   XLATE    (14 forms)
   TIME     (27 forms)
   CTD      (10 forms)
   CFD      (10 forms)
   GETENV   ( 1 form )
```

How it works

The main instruction timing is achieved by executing the instruction
a number of times using a BCT loop and timing the process.

1) BCT timing
   In order to separate the instruction timing from the BCT time
   we must first obtain an accurate timing for the BCT itself.
   This only needs to be done once.
   This time is referred to as timerBCT time in the following text.

eg.

```
    -- save start time --
    L   R7,bct_spin_count
    BCT R7,*
    -- save end time --
```

   Calculate end-start.
   Save as timerBCT time.

2) Instruction timing
   This covers the majority of instructions.

eg.

```
     -- save start time --
     L   R7,spin_count
TEST DS  0H
     -- tested instruction --
     BCT R7,TEST
     -- save end time --
```

   Calculate end-start-timerBCT time.

3) Destination or register modifiers
   Instructions like DP, ED and many others cannot be looped because
   they change the destination field or registers.
   In these cases, one or more special instructions are timed before
   the main loop.

eg.

```
     -- save start time --
     L   R7,spin_count
XTRA DS  0H
     -- tested extra instruction(s) --
     BCT R7,XTRA
     -- save end time --
```

   Calculate end-start-timerBCT time.
   Saved as extra time.

```
     -- save start time --
     L   R7,spin_count
TEST DS  0H
     -- extra instruction(s) --
     -- tested instruction --
     BCT R7,TEST
     -- save end time --
```

   Calculate end-start-extra time-timerBCT time

4) Stack and Unstack instructions

   The stack is limited to 50 entries, so stacks and unstacks are
   looped 50 at a time until the required loop limit is reached.
   The reverse operation is also done but untimed.

   If stack/unstack operations are being timed and the spin count
   is modified with the SPIN= parameter, then the SPIN value must be
   a multiple of 50 or a SPIN ERROR will occur.

Input and Output
================
Add the SYSPARM statement to the CALL MZ390 as follows, with your own
path name:

```
   CALL MZ390 ... "SYSPARM(E:\Z390\PROGRAMS)"
```

Input...INSTTIME.TXT...

   The input file is the list of instructions to be timed and other
   special parameters. All the options are described later.

   The input file is not supplied.

   All the instruction variations are in the file INSTTIME.INS
   Copy what you want timed from this file into INSTTIME.TXT
   Any special parameters can then be added as you wish.

Output...INSTTIME.OUT...
   If the output file already exists it will be overlayed by the new one.

At the time of writing, the complete run takes about 20 mins.

INPUT parameters
================
All parameters start in col 1 and are generally inflexible.

*
   A comment line which is ignored.

BCTS=nnnnnnn
   This alters the number of times the timerBCT is executed.
   Devised in order to optimise the initial BCT timing.
   The default is 5000000.
   The number must be 7 numeric digits.
   It can be placed anywhere in the input file and will cause
   retiming of the BCT instruction.
   Modifying this below the default may lead to inaccurate timing.

SPIN=nnnnnnn
   This alters the number of times each instruction is executed.
   The default is 0100000.
   The number must be 7 numeric digits.
   It can be placed anywhere in the input file.

   Modifying this below the default may lead to inaccurate timing.
   Modifying this above the default will lead to longer run times.

   There is one special case:
   SPIN=0000001 may be used to verify that an instruction is being
   formed properly by executing it once. The timings also reflect the
   overhead incurred per execution. These were used to justify the
   default.

   SPIN= must be a multiple of 50 when timing stack/unstack
   instructions. See above for more information.

STATS ON
   Sets an indicator to start the collection of timings.
   This does no resetting.
   All timings from this point, including the retiming of the BCT
   after a BCTS= parameter will be collected.

STATS OFF
   Stops the collection of timings, resets everything.
   Restart the collection of timings with another STATS ON.

STATS PRINT
   Note: In V3 the parameters MEAN and VARIANCE were used. These have
         now been withdrawn and replaced by this parameter.

   Three stats are shown: MEAN, VARIANCE and STD. DEV.

```
   This does no resetting, so 'running' stats are possible.

LINREG/LL
LINREG/L1
LINREG/L2
LINREG/SL
   Linear regression on single length (LL), on either L1 or L2
   lengths or string length (SL). The latter only applies to CUSE.

   I have imposed a minimum of 3 collected timings.
   This does no resetting, so 'running' linear regression is possible.

   This was designed as a predictor for some instructions that are
   impractical to time, eg. an MVCLE of 10MB. Although this is its
   primary use, it may be useful for other instructions. More on
   those later.

   Two lines are shown, TIME and CC.

   This example is a good one...
      MVCLE       50000    15.378707
      STATS ON
      MVCLE        1000     2.981785
      MVCLE        5000     4.126426
      MVCLE       10000     5.780736
      MVCLE       20000     8.102098
      TIME =       2.81759060 + (   0.0002700189 * LENGTH )
      CC=          0.996774
      STATS OFF

   TIME is a simple formula used for prediction.
   In the above example, the 50K move is outside the stats collection.
   The formula yields a predicted value for 50000 at 16.32uS which is
   pretty close to the timed value of 15.38uS.

   CC is the correlation coefficient, any value above 0.9 means the
   the formula may be a good predictor, the nearer to 1 the better.

   For linear regression using L1 or L2, some care needs to be taken
   for the results to be statistically valid. When used for instructions
   with two lengths; if L1 is used for regression, then L2 must be
   identical for all collected timings and vice versa.
   eg.
      STATS ON
      SS2   MP      P    01  08
      SS2   MP      P    04  08
      SS2   MP      P    08  08
      SS2   MP      P    16  08
      LINREG/L1
      STATS OFF

Input format
============
This is largely for information only.
```

```
Only length fields may be modified.
Each instruction is independent of any other, there is no sequencing
and the same instruction may be repeated.

*FMT  -INST-  TYPE --LL-- -SL   Comments
*                  L1  L2
*                  XB

FMT, INST and TYPE must not be modified.

FMT   The instruction format.
      I reserve the right to invent my own variations.

INST  Instruction mnemonic.
      I may add some synonyms.

TYPE  Documented in the source at label INSTTYPE.

The XB heading means:
   00   Don't use index or base registers.
   X0   Use index but not base registers.
   0B   Use base but not index registers.
   XB   Use both base and index registers.
     S  Store or formatted data type.
Note: Instructions that may operate incorrectly or damage system
      integrity with any of the above are rejected.

--LL-- for instructions that use a single length.
       Invalid lengths are rejected.
       1-999999 depending on instruction.

L1  L2 for instructions that use two lengths.
       Invalid lengths are rejected.
       SRP ignores L2.
       LM/STM types use L1 as no. of registers.

SL     for instructions that use a string length (CUSE)

The RESULT file
===============
INST, TYPE, XB or lengths.
Time in microseconds.
Dump of instruction.
Stats as requested.

Other data is used for integrity checking.
   Instructions that alter registers have the result after SPIN
   executions. Both even/odd registers are displayed in 32 or 64-bit
   format irrespective of which parts are actually used.

   Decimal instructions have a 16-byte field displayed, at least
   part of which has been used for a decimal instruction after SPIN
   executions.
   Floating point instructions have the resulting value(s) converted to
   decimal using the CTD macro.

Notes
=====
The EX (Execute) instruction timing is the overhead involved and not
the timing of an execute and target instruction.

Synonyms...only base synonyms are supported as instructions. Others
can be timed using parameters.
eg. You can't specify NOPR as an opcode, but you can time a BCR with
a zero mask. The same goes for JNOP and BRC 0,label etc.

I don't support branch mnemonics with masks that are not 0 or 15.

FAQ
===
Q1) I have timed repetitions of the same instruction, so why aren't
    all the times the same.

A1) The Operating System (eg. Windows) has priority over any application
    programs. It can therefore 'steal' cycles from the Z390 environment
    to process other applications or to do general stuff like paging,
    interrupts, clock update etc.

    The best timings are obtained by not running any other applications
    and by not touching the keyboard or mouse.

    A small improvement may be made by running the program in a Command
    Prompt window and not in the Z390 GUI.

Author: Melvyn Maltz
Publication date: September 27, 2007
Z390 version: V1.3.08
INSTTIME version : 10
→
```