#SHAREorg

SHARE
Technology · Connections · Results

Real Kernel Debugging in a Virtualized Environment

Dr. Stefan Reimbold
IBM Research & Development

Aug 10, 9:30 am

Session No. 11844

SHARE in Anaheim 2012

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market. Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

* AS/400®, e business(logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply. All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions. This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area. All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- Introduction
- Basic CP commands for debugging
- Tracing execution of the guest system
- Program Status Word
- Examples
  - Modify system memory
  - Operation Exception

System Setup

- Linux System as a z/VM guest
- z/VM 6.2
- Examples with SLES11 SP1
- Nothing special used, so should work with others as well

CP Commands

- BEGIN
- DISPLAY
- SLEEP
- STOP
- STORE
- TRACE

CP Commands

- BEGIN
- DISPLAY
- SLEEP
- STOP
- STORE
- TRACE
- Need Class G Priveleges

Pausing the Guest System

```
#CP STOP
#CP BEGIN
```

Stop the guest system and continue to run.

Pausing the Guest System

```
#CP SLEEP
#CP BEGIN
```

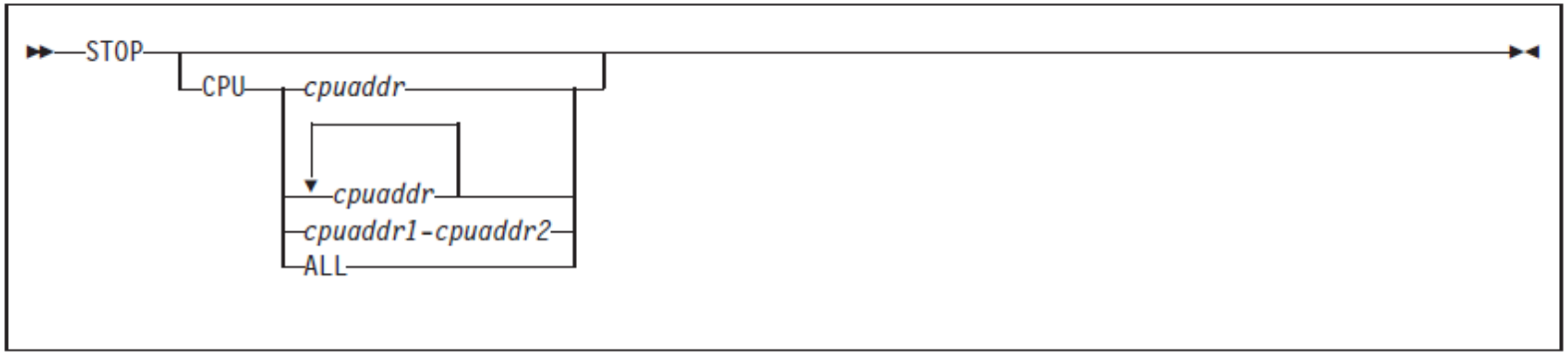
Set the guest system to sleep.
Basically the same as STOP.

Pausing the Guest System

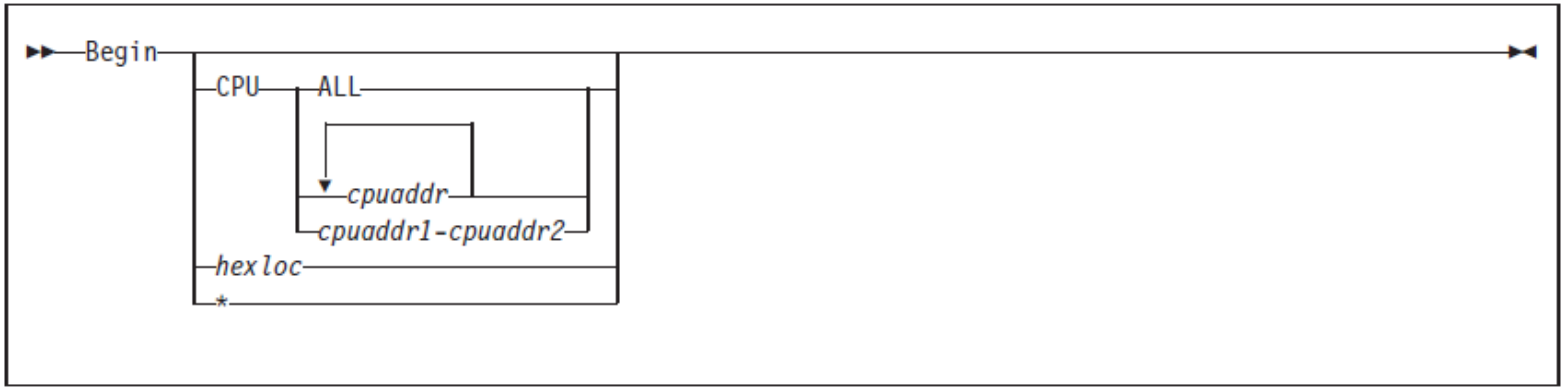
```
#CP SLEEP 10 SEC
```

Let the guest system sleep for 10 seconds. Can also be minutes or hours.

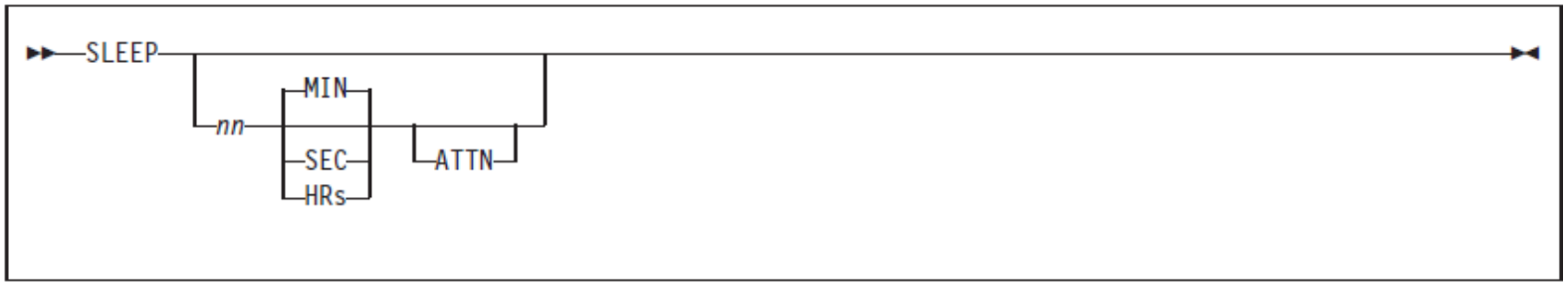
STOP



BEGIN



SLEEP



DISPLAY

- Display general-purpose registers

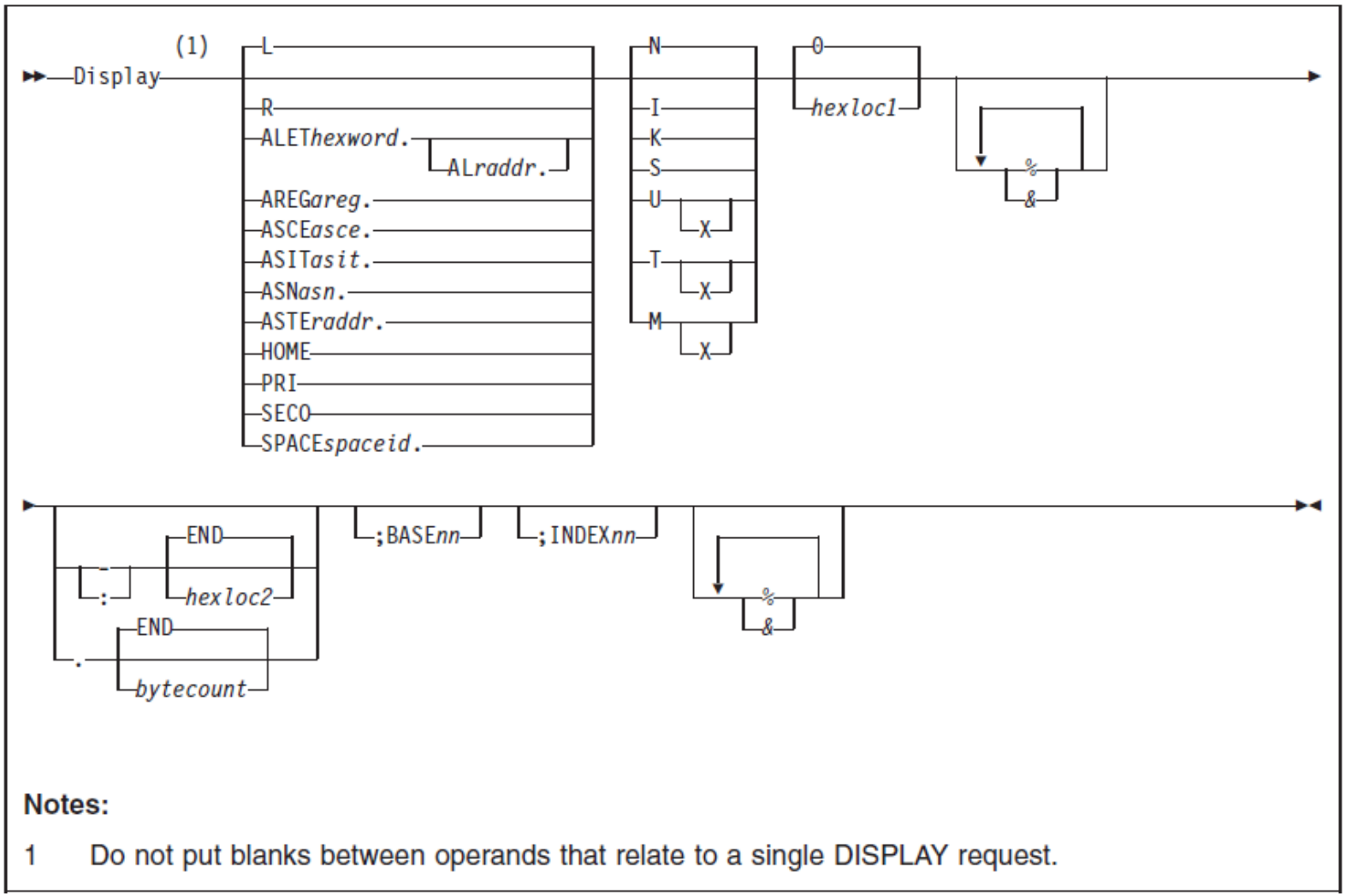
```
DISPLAY G
00: GPR  0 =  00000000  8000060C  00000001  FF9B1518
00: GPR  4 =  FF9B1528  001B8F70  8000066C  FF9B1500
00: GPR  8 =  800004EC  800F17B0  80000670  801C05E0
00: GPR 12 =  001B6000  0016A930  00057898  FF9B12F0
```

DISPLAY

- Display general registers-purpose in 64 bit format

```
DISPLAY GG
00: GRG  0 =  0000000000000000  000000008000060C
00: GRG  2 =  0000000000000001  000003FFFF9B1518
00: GRG  4 =  000003FFFF9B1528  00000200001B8F70
00: GRG  6 =  000000008000066C  000003FFFF9B1500
00: GRG  8 =  00000000800004EC  00000000800F17B0
00: GRG 10 =  0000000080000670  00000000801C05E0
00: GRG 12 =  00000200001B6000  000002000016A930
00: GRG 14 =  0000020000057898  000003FFFF9B12F0
```

DISPLAY



Notes:

1 Do not put blanks between operands that relate to a single DISPLAY request.

Display Storage

- Display Linux virtual storage

```
DISPLAY 80002040
00: V80002040   0000007F                                  06 R2E3FE040
```

Display Storage

- Display Linux real storage

```
DISPLAY R2E3FE040
00: R2E3FE040   0000007F                                  06
```

Modify Storage

- Modify Linux virtual storage

```
STORE 80002040 18
00: Store complete.
DISPLAY 80002040
00: V80002040  00000018                                  06 R2E3FE040
```

Trace

Display current trace settings

```
QUERY TRACE
00: No trace sets defined
```

Trace

Trace any write operation to a memory location
Operation is stopped at this point

```
TRACE STORE INTO 80002060
```

This is a breakpoint

Trace

```
QUERY TRACE
00:
00: NAME  INITIAL     (ACTIVE)
00:
00:   1     STORE   FROM  0000000000000000-FFFFFFFFFFFFFFFF
00:               INTO  80002060
00:         TERM    NOPRINT  NORUN  SIM
00:         SKIP 00000  PASS 00000  STOP 00000  STEP 00000
00:         CMD  NONE
00:
```

Trace

End tracing and remove all trace settings

```
TRACE END
00: Trace ended
```

Trace

```
QUERY TRACE
00: No trace sets defined
```

Trace

Set 2 breakpoint

```
TRACE STORE INTO 80002050
TRACE STORE INTO 80002060
```

Trace

```
QUERY TRACE
00:
00: NAME  INITIAL     (ACTIVE)
00:
00:   1     STORE   FROM  0000000000000000-FFFFFFFFFFFFFFFF
00:                 INTO  80002050
00:         TERM    NOPRINT  NORUN  SIM
00:         SKIP 00000  PASS 00000  STOP 00000  STEP 00000
00:         CMD  NONE
00:
00:   2     STORE   FROM  0000000000000000-FFFFFFFFFFFFFFFF
00:                 INTO  80002060
00:         TERM    NOPRINT  NORUN  SIM
00:         SKIP 00000  PASS 00000  STOP 00000  STEP 00000
00:         CMD  NONE
00:
```

Trace

Remove a single trace setting

```
TRACE DELETE 1
```

Trace

```
QUERY TRACE
00:
00: NAME  INITIAL     (ACTIVE)
00:
00:   2     STORE   FROM  0000000000000000-FFFFFFFFFFFFFFFF
00:                 INTO  80002060
00:         TERM    NOPRINT  NORUN  SIM
00:         SKIP 00000  PASS 00000  STOP 00000  STEP 00000
00:         CMD  NONE
00:
```

Trace

Set a breakpoint on a instruction address

```
TRACE END
TRACE INSTRUCTION PSWA 8000060C
```

Trace

Set a breakpoint on a instruction address

```
TRACE END
TRACE INSTRUCTION PSWA 8000060C
QUERY TRACE
00:
00: NAME  INITIAL     (ACTIVE)
00:
00:   1     INSTR   PSWA  8000060C
00:         TERM    NOPRINT  NORUN  SIM
00:         SKIP 00000  PASS 00000  STOP 00000  STEP 00000
00:         CMD  NONE
00:
```

Trace

Set a breakpoint on a instruction address without interrupting operation

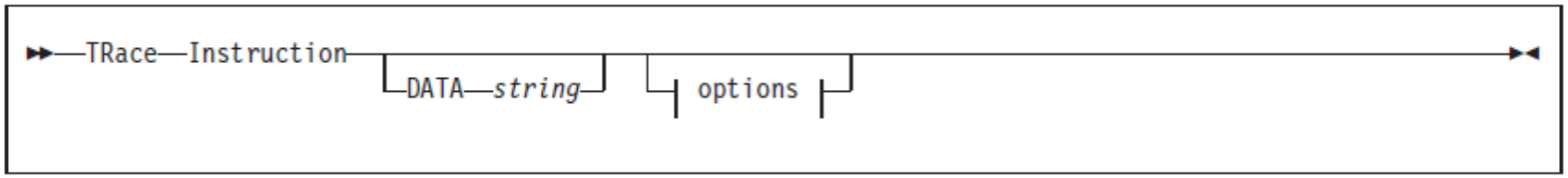
```
TRACE END
TRACE INSTRUCTION PSWA 8000060C RUN
```

Trace

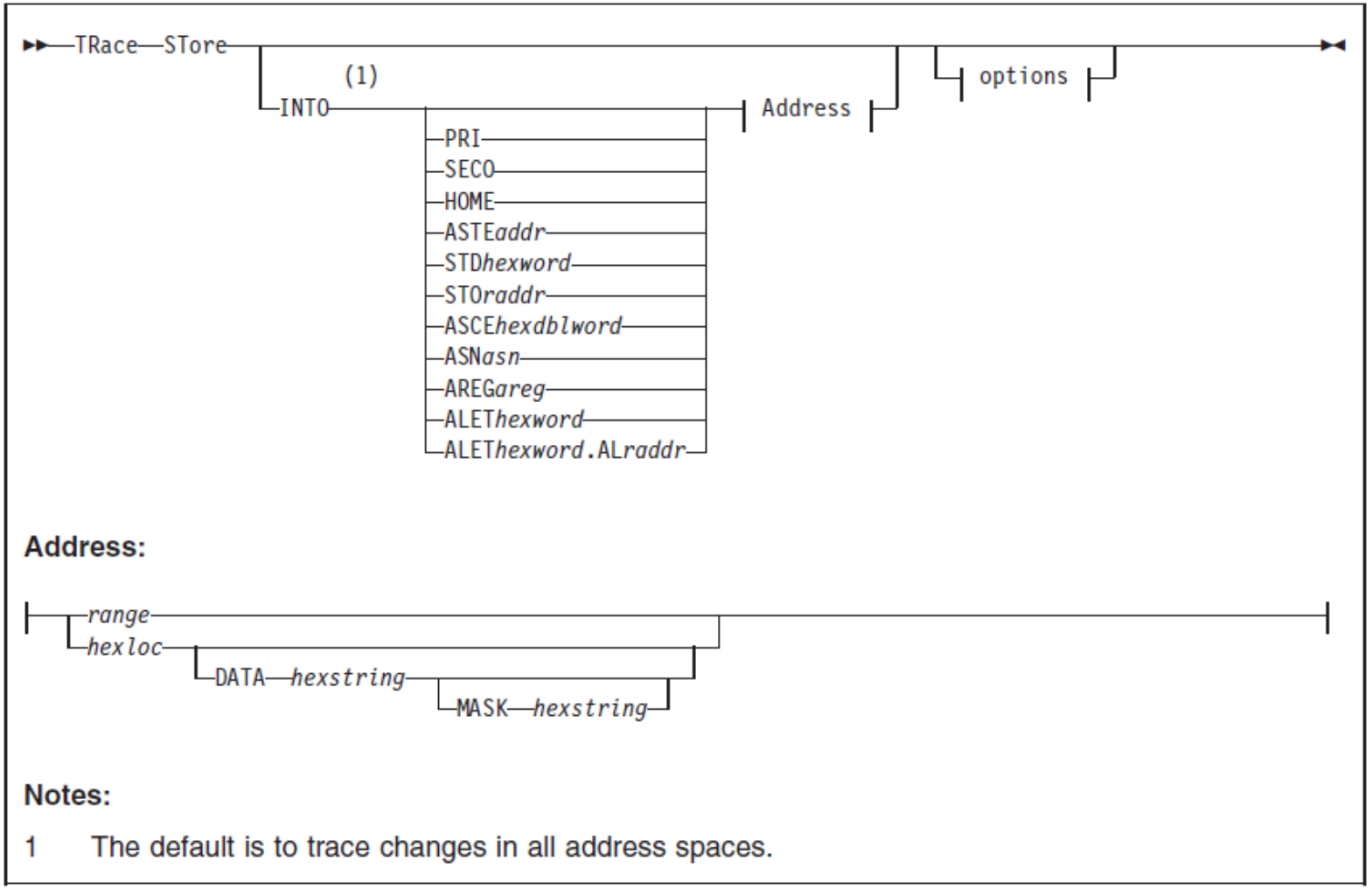
Set a breakpoint on a instruction address without interrupting operation

```
TRACE END
TRACE INSTRUCTION PSWA 8000060C RUN
QUERY TRACE
00:
00: NAME  INITIAL     (ACTIVE)
00:
00:   1     INSTR   PSWA  8000060C
00:         TERM    NOPRINT  RUN    SIM
00:         SKIP 00000  PASS 00000  STOP 00000  STEP 00000
00:         CMD  NONE
00:
```

TRACE



TRACE STORE


TRace STore
(1)
INTO
PRI
SECO
HOME
ASTEaddr
STDhexword
STOraddr
ASCEhexdblword
ASNasn
AREGareg
ALEThexword
ALEThexword.ALraddr
Address
options
Address:
range
hexloc
DATA hexstring
MASK hexstring
Notes:
1 The default is to trace changes in all address spaces.

Program Status Word

- Display Program Status Word

```
DISPLAY PSWG
```

Program Status Word

- Display Program Status Word

```
#CP DISPLAY PSWG
00: PSW = 07060001 80000000 00000000 00115326
```

Program Status Word

- Show Program Status Word

```
#CP DISPLAY PSWG
00: PSW = 07060001 80000000 00000000 00115326
```

Shows always the same
Why ?

Program Status Word

- Show Program Status Word

```
#CP DISPLAY PSWG
00: PSW = 07060001 80000000 00000000 00115326
```

Shows always the same
Why ?

Where is the program execution currently

Program Status Word

```
#CP STOP
DISPLAY PSWG
00: PSW = 07060001 80000000 00000000 00115326
```

Look up instruction address in /boot/System.map

```
0000000000114ef4 T arch_setup_additional_pages
0000000000115064 T vdso_free_per_cpu
0000000000115100 T vdso_alloc_per_cpu
00000000001152a0 T vtime_stop_cpu
000000000011537c T s390_get_idle_time
0000000000115408 T init_virt_timer
0000000000115420 T init_cpu_vtimer
```

Program Status Word

Display all Program Status Words

```
DISPLAY PSWG ALL
00: PSW = 07060001 80000000 00000000 00115326
00: EXT 1004  130 OLD 07060001 80000000 00000000 00115326
00:           1B0 NEW 04040001 80000000 00000000 001184CC
00: SVC 00A8  140 OLD 0705C001 80000000 00000200 003C136A
00:           1C0 NEW 07040001 80000000 00000000 00117DAC
00: PRG 0004  150 OLD 0705D001 80000000 00000200 0011A43E
00:           1D0 NEW 04040001 80000000 00000000 00118068
00: MCH 0000  160 OLD 00000000 00000000 00000000 00000000
00:           1E0 NEW 00000001 80000000 00000000 001185BA
00: I/O 0003  170 OLD 07060001 80000000 00000000 00115326
00:           1F0 NEW 04040001 80000000 00000000 001182F4
```

Program Status Word

```
00: PRG 0004  150 OLD 0705D001 80000000 00000200 0011A43E
00:           1D0 NEW 04040001 80000000 00000000 00118068
```

```
grep 118068 /boot/System.map
0000000000118068 T pgm_check_handler
```

Program Status Word

```
00: I/O 0003  170 OLD 07060001 80000000 00000000 00115326
00:           1F0 NEW 04040001 80000000 00000000 001182F4
```

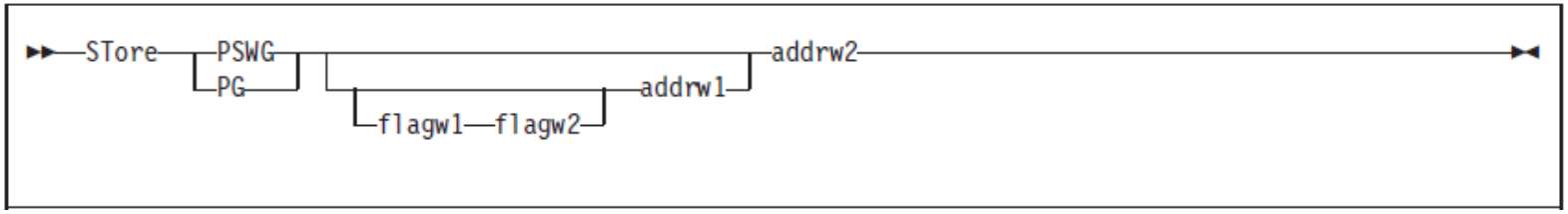
```
grep -i 1182F4 /boot/System.map
000000000001182f4 T io_int_handler
```

Program Status Word

```
00: SVC 00A8  140 OLD 0705C001 80000000 00000200 003C136A
00:           1C0 NEW 07040001 80000000 00000000 00117DAC
```

```
grep -i 117DAC /boot/System.map
0000000000117dac t __critical_start
0000000000117dac T system_call
```

STORE PSWG



Modify system memory

Simple example code prints a variable, changes the variable and prints it again

```
#include <stdio.h>

int a = 34;

int main() {
        printf("Code Segment   = %p\n", &main);
        printf("Data Segment a = %p\n", &a);

        printf("a = %d\n", a);
        a = 127;
        printf("a = %d\n", a);

        return 0;
}
```

Modify system memory

Normal execution

```
./memory
Code Segment    = 0x8000060c
Data Segment a = 0x80002040
a = 34
a = 127
```

Modify system memory

Set break point at entrance of main()

```
#CP STOP
TRACE INSTRUCTION PSWA 8000060C
BEGIN
```

Modify system memory

```
./memory
```

```
00:   -> 0000000080000060C"   STMG     EBBFF0580024 >> 000003FFFF86B348      CC 2
DISPLAY PSWG
00: PSW = 0705E001 80000000 00000000 80000612
DISPLAY 80002040
00: V80002040   00000022                                 06 R2E3FE040
```

Modify system memory

```
./memory
```

```
00:   -> 0000000080000060C"  STMG    EBBFF0580024 >> 000003FFFF86B348     CC 2
DISPLAY PSWG
00: PSW = 0705E001 80000000 00000000 80000612
DISPLAY 80002040
00: V80002040  00000022                          06 R2E3FE040
TRACE STORE INTO 80002040
BEGIN
```

Set a breakpoint on the variable a

Modify system memory

```
./memory
```

```
00:   -> 0000000080000668"   ST       50201000 >> 0000000080002040      CC 0
DISPLAY 80002040
00: V80002040   0000007F                              06 R2E3FE040
```

Modify system memory

```
./memory
```

```
00:   -> 0000000080000668"  ST       50201000 >> 0000000080002040      CC 0
DISPLAY 80002040
00: V80002040  0000007F                          06 R2E3FE040
STORE 80002040 18
00: Store complete.
```

Modify value of variable a

Modify system memory

```
./memory
```

```
00:   -> 0000000080000668"  ST       50201000 >> 0000000080002040      CC 0
DISPLAY 80002040
00: V80002040  0000007F                          06 R2E3FE040
STORE 80002040 18
00: Store complete.
DISPLAY 80002040
00: V80002040  00000018                          06 R2E3FE040
```

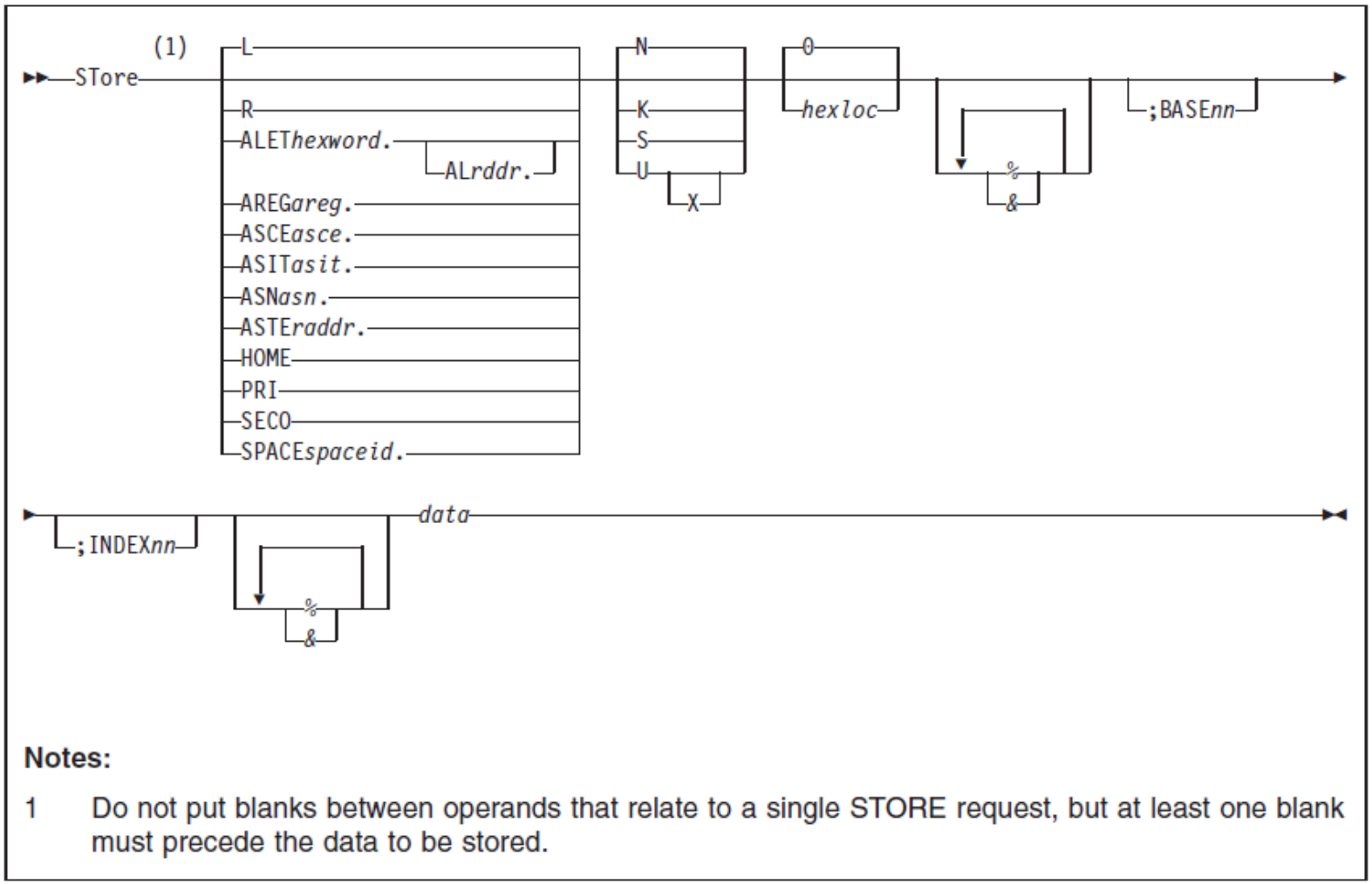
Modify system memory

```
./memory
Code Segment    = 0x8000060c
Data Segment a = 0x80002040
a = 34
a = 24
```

Modify system memory

```
QUERY TRACE
00:
00:  NAME   INITIAL      (ACTIVE)
00:
00:    1      INSTR    PSWA   80000060C
00:           TERM     NOPRINT  NORUN   SIM
00:           SKIP 00000   PASS 00000   STOP  00000   STEP  00000
00:           CMD   NONE
00:
00:    2      STORE    FROM   0000000000000000-FFFFFFFFFFFFFFFF
00:                    INTO   80002040
00:           TERM     NOPRINT  NORUN   SIM
00:           SKIP 00000   PASS 00000   STOP  00000   STEP  00000
00:           CMD   NONE
00:
```

STORE



Notes:

1 Do not put blanks between operands that relate to a single STORE request, but at least one blank must precede the data to be stored.

Operation Exception

- Force an operation exception

Operation Exception

```
#CP STOP
TRACE INSTRUCTION PSWA 8000060C
BEGIN
```

Operation Exception

```
./memory
```

```
00:   -> 0000000080000060C"  STMG    EBBFF0580024 >> 000003FFFF86B348     CC 2
DISPLAY PSWG
00: PSW = 0705E001 80000000 00000000 80000612
STORE 80000060C CC000000
BEGIN
```

Operation Exception

```
./memory
Code Segment    = 0x8000060c
Data Segment a = 0x80002040
a = 34
a = 127
```

Surprise !
The modified instruction was not hit !

Operation Exception

Let’s do it again !

```
./memory
```

```
00:   -> 000000008000060C"  ????    CC0000000024     0000000000000000
00:                                                  0000000000000024
00: *** 000000008000060C"       PROG    0001 -> 0000000000118068’
00:                OPERATION
```

Operation Exception

```
00:   -> 000000008000060C"   ????     CC0000000024     0000000000000000
00:                                                    0000000000000024
00: *** 000000008000060C"        PROG     0001 -> 0000000000118068'
00:                 OPERATION
BEGIN
```

```
./memory
Illegal instruction
```

Operation Exception

```
00:   -> 0000000080000060C"   ????     CC0000000024     0000000000000000
00:                                                     0000000000000024
00: *** 0000000080000060C"       PROG     0001 -> 0000000000118068'
00:                 OPERATION
BEGIN
```

Let's look where the exception is handled

```
grep 118068 /boot/System.map
0000000000118068 T pgm_check_handler
```

Summary

- Full guest memory accessible
- You might have to find out real addresses
- Can modify anything
- Done by z/VM
  - Disassembly
  - Virtual to Real address translation
  - Program-Interuption codes

Links

- VMCP Commands and Utilities Reference
  `http://publibz.boulder.ibm.com/epubs/pdf/hcse4b21.pdf`
- Debugging on Linux for S/390 & z/Architecture by Denis Joseph Barrow
  `http://www.kernel.org/doc/Documentation/s390/Debugging390.txt`
- Linux for S/390 - ELF Application Binary Interface Supplement
  `ftp://archive.download.redhat.com/pub/redhat/linux/7.2/en/os/s390/doc/l3`

Questions ?

Dr. Stefan Reimbold
Diplom-Physiker

Linux on System z Service

Schoenaicher Strasse 220
D-71032 Boeblingen
Mail: Postfach 1380
D-71003 Boeblingen

Phone +49-7031-16-2368
Stefan.Reimbold@de.ibm.com

Please Evaluate