Vol. 5

National® Brand

College Ruled White Paper

Single Subject

33-986

50 Sheets/11 x 8½

Dennison National Company, Holyoke, MA 01041

Brian R. Page

MSDD2
MB Group

Macrocode Structure and Debugging

8-6-85

Dan Greiner
DFGD

Materials are confidential and cannot be reproduced. Instructor notes will be available. Everything is on script.

Why do we need macrocode:

Need to implement new instructions

370/EF

DAS (3033)

XA

Interpretive Execution.

Have RMS without modifying. IBM operating system.

Provide >16 channels in 370 mode.

Plus the additional benefit: MDF

- Address Translation -

Domain Logical Address

DAT

Domain Real addr.

Prefixing

Absolute Addr.

Mapping

Sys Absolute Addr.

State:

Control State

Supervisor

(Fast) Assist (instruction simulation)

User State

GP Registers: (FAM

two sets - one for control state x'180'
" " user " x'200'

System Regs (SR) x'120'

Control Regs (CR) x'1CØ'

FP Regs

GPRs - contiol & user regs are stored in RAM and are moved by the selector in the I unit. Only one set of regs are normally active at a time.

System reg Ø - prefix for active domain Macrocode can take an interrupte that the domain is masked for.

We can have duplicate opcodes with IBM but ours would be in control state & would take a different path.

Initialization

Read 15x change summary.

Satish Channa
SKC00 - will be on Apache.

Storage Manager - takes care of all storage, even domain storage. Creates MSB to describe chunks.

SST - tracks pages within macrocode

Call Manager

Dan

Transfers control between modules

A module in mac has

1 csect

Segment = EPA (may be multiple)

First segment is the same name as the module & is always the first called entry. The subsequent Entry points can only be called from within the same module. (BAS, not through call manager)

Size < 4K (page)

Allocate WSAs to transfer control.

Does tracing for

sys flow

performance data

module entry

Call Manager changes PSW key to that of called module.

Call Manager: X'A00' to 'FFF' - tight code.

Dispatcher - dispatches work on one or more CPU. Dan

First PCB on each queue (ready + waiting) is a dummy. PSA + PSB are key 5. PCBs are key 3 (dispatchers key). Dummy can anchor other PCBs + stay in same key. The pointers in the key Ø areas need not be changed + the dispatcher can stay in key 3.

— Dump 1 — Satish

Questions:

1. What is the address of STORDATA?
2. " " " SST?
3. How many free pages are there?
4. What is the address of the last SSTE?
5. Write down the characteristics of this page.
6. What is the address of the PSB?
7. " " " " " MVT?
8. " " " " " PST for key Ø?
9. Are there any FSBs for any size (1 to 8) for key Ø? Explain.
10. What is the address of the PST for key 7?
11. Where does the FSB chain start for size 2 type key 7 FSB?
12. How many times was GETSTOR issued against it?
13. If MACEND was configured to CØØØØ and ICA size was 1AØØØ, how many SSTEs will you see?

1. stordate is pointed to by PSBSST and
   PSAMVT → $TORDATA field of MVT
   (PSAMVT: 7FØ, $TORDATA: 1C)

   stordata is at 9ØØØ

2. SST is embedded in stordata.
   addr = 94Ø.

3. Free page count is at +A into SST
   value = ØØØ2

4. Last SSTE?
   # of SSTEs = BE
   first SSTE @ 9Ø5Ø
   2F8
   9348 = end of last SSTE

   BE = 176
   14
   190
   + 4
   7D8₁₀ = 2F8
   = 512
   248

   Last SSTE @ 9344

5. Characteristics of this page = ØØCEØ4ØØ8 =
   key = E
   ICA page
   Storage pool size index for page size.

6. addr of PSB?
   PSAPSB (+'78Ø) = 69ØØØ

7. addr of MVT?
   PSAMVT (+7FØ) = 689ØØ

8. PST for key Ø?

8-7-85

satish

ACS

All mac interactions with the hard disk pass thraugh ACS.

WCCs may perform one function, or combine several legal functions:

Open
Open, read
Open, read, close

WCCs are chained from NFAs.

Set up a WCC and a NAW (Norway Addr word) and issue 8305. (Write Console Command instruction).

Console Management

Calls upon other components: IOS, Norway(ACS) File system, & Frame manager.

Authority check on FE key: check occurs only if check of FE key has not occurred within 1 sec!

1 console control block for a master con, & 1 for each domain con.

One PCB per CCB.

CCB is the main control block for console. Each CCB points to its log of 25 lines. Check here after problems. (System log is off of PSB & holds 50 lines).

Frame Manager

Satish

30-40% of mac is frame manager.
Frame Manager performs functions for the console manager. Only console manager calls frame manager.
Frame manager calls everybody.

Module naming

Pxxy

where xx = frame id.
y = 2 for XA
3 for 370
(some exceptions to this)

See listing of CFRAMTBL.

File Manager

Satish

Has largely replaced the Norway routines.
File manager puts out an ECB to be posted. Whenever another component needs something done, his ECB gets posted & he gets dispatched!
No one ever calls File Manager!
File manager interfaces with ACS for file I/O.

Domain Control

Dan

Logical Processor

Mac looks at user state as a call of a subroutine & is ended by an interupt. The Logical Processor's PCB is the same as any other to mac.

8-8-85

Timer Services

Switch

Only two control blocks! : TQB + TSB

5 timers:

- CPU timer - mac does not get interrupts from here.
- (domain) Interval Timer - for domains (Domain location 80 gets updated)
- Clock Comparator - Doubleword Reg (Set Clock Comparator)
- TOD clock - Doubleword (bit 51 increment/μsec)
- System timer - used for time slicing of logical processors. It decrements.

Decremented compared to ↑

The clock comparator register gets loaded when a domain gets dispatched. If clock comparator value is ≤ TOD, generate interrupt.

TOD clock - only the left half of the doubleword is stored (as the TOD displacement). Thus, the low order time of day will be the same for both domains. (Logical MPs).

PSA + 399 has timer flags that are helpful in timer problems.

TSB - timer synchronization Block

Checkpoint Process

2 control blocks: CKT + CKB. One checkpoint area for entire system.

Only 'static' portion of each CB is copied.

Multiprocessor Support — Dan

FAM

Eliminates the need for SCP modifications.
FAM info seems anchored in CPBs.

I/O Supervisor

IDecode IC - listing (F0EC009C) } read
Augmented Channel Considerations - } (chap. 3)
see Trimble's reference for UCW + CUB blocks.
IOS has proved to be a reliable component.

Recovery — Satish

PSA + DCA are important during recovery.

Diagnostic Techniques — 8-9-85 Dan

OPSYN ~~xxxxxx~~ copy statement
takes BAL
BALR
Using
Drop
LPSW
STCK
Trace
and ~~xxxxxx~~ prefix with $.

Dumps 1 & 2 are real problems — Satish
→ easy →

MVS/XA Debugging

11-4-85
Bob Tennant
BZT4Ø

Recovery Termination Manager - is intimately involved in errors and is therefore a good place to find status. RTM must diagnose the error to attempt recovery. RTM builds a work area for info on the problem.

RTM is called by SVC 'D'
- Pgm Check
- Mch Check
- SVC call by unauthorized routine
- Paging I/O error
- OAT error (ØC4)
- Restart
- Alternate CPU Recovery (ACR) - removes failing processor from the complex. All work is rescheduled for surviving processor.

Two levels of RTM:

RTM1 - for code running disabled or locked. RTM1 gets called for ABTERM, MEMTERM, PROG chk, Mach chk, page I/O error, SVC error, or Restart. RTM1 (IEAVTRT1) mainline performs FRR esculation, retry, & Logrec recording.

RTM2 - Enabled, & unlocked code. May be called by RTM1. Entry is via SVC ØD'. ESTAE/ESTAI routines are called from here. Also, retry, cleanup, and dump. IEATRT2 - mainline routine.

RTM handles normal termination for tasks.

STAE control block (SCB) TCB + 'A1'

+Ø ↑ Next SCB (chained for percolation). Ø = end of chain.
+4 ↑ user routine to get control at error.
+8 ↑ Parm List
+C ↑ TCB
+10 Option flag

The SCB is set up when the ESTAE macro is processed. Percolation halts if the error is successfully recovered. SCB is reviewed by RTM2.

FRR stack

PSA +'380' RSVT - Recovery Stack Vector Table. Used by RTM1 to find a FRR stack for major components (for running disabled).

SVC, I/O, & Dispatcher all share an FRR stack. Normal pointer is alway 'CØØ' to point to the "normal" stack for routines issuing FRRs and not having a special FRR stack. Ordinarily the Current pointer (+380) should be the normal pointer. Else, the component requiring FRR protection will have its pointer in the current location. This just means that recovery is in place, not that an error has occurred. Check the current pointer to see which component has control. SET FRR changes the current pointer. Design is to only have FRR protection when actually needed.

Document as FRRS.

FRR stack: permits RTM1 to find recovery routines. FRR header has

+0 ↕ Top
+4 ↑ Last
+C ↑ current entry (which then points to the FRR)

Can have 16 entries. The first one is fairly generic + just causes documentation.

+28 RTM1 work area - holds some status information gathered by RTM1 to pass on to the FRR.

RTM1W indicators:

+40 - system mode at time of error.
'80' Supervisor
'40' disabled
'20' Global spin lock held

These fields are only set for the duration of the error.

+2 Error Type
'1' - Pgm check

+34 ↑ Current SDWA

SDWA System Diagnostic Work Area.

Extremely valuable. Pointed to by RT1W+34 and RTM2WA + '04'

+4 Flags + Comp Code
+18 GPRs
+60 Abending pgm EPA
+68 Error PSW
+E8 Error type, sys status
+124 error module } 10% of the time it
+12C " csect } may be incorrect. Verify!
+F4 ↑ Variable Recording area

SDWA comes out in the LOGREC entry.

Variable Recording Area - (SDWA +190) used by the recovery routines to save information.
SDWA +190 is embedded VRA. Has flag at that location:

80 - hex - probably FRR work area
40 - EBCDIC - message to you
20 - key-length data - see VRAMAP dsect in Vol. 5 of Debugging Handbook. Data is at SDWA+190.

SDWA +F4 is pointer to VRA.
SDWA +190 is VRA.
First word at +190 describes nature of VRA.

RTM2WA - TCB +E0 Many times this is all the doc needed, particularly for SVC dumps.
If TCB +E0 ≠ 0 then an error has occurred.

SUMDUMP
Print Current Summary } will all format RTM2WA

+1C Comp Code + flags
+3C GPRs
+7C PSW at error
+8C Pgm name
+94 Pgm EPA
+B4 Error type + mode
+D4 ↑ SDWA

Output of RTM: a dump.

SVC - one address space. Global areas not valid.

Sysabend, Sysmdump/sysudump

Wait state / stand-alone dump

User dumps:

Sysudump - IEADMP00 - Parmlib member defining dump.

Sysabend - IEAABD00 -

sysmdump - IEADMR00 -

DD card must be present in job stream to get a user dump.

Sysudump & sysabend go to sysout. Sysmdump goes to disk dump dataset & must be formatted AMDPRDMP or IPCS.

See Init & Tuning for options of the Parmlib members defining each dump.

Sysudump - SVC dump.

Check ID field on title line:

000 - failing task

001 - subtask

002 - originator

Check PSW, ILC, and interrupt code.

If Length = 2, the instruction is an SVC in most probability. Interrupt code will be SVC number.

Lengths = 4 + 6 pgm check

code - PIC (Peach card p. 19)

Completion code 0C1 is a bit different: length = 2, ILC = 1

Back up one instruction from PSW address.

Examine RBs of failing task:

Status is saved upon leaving an RB. RBs are listed in order of creation. Call or SVC causes creation of a new RB.

RB field WLIC and OPSW tell why we left the RB. Interrupt code & instruction length may show us a pgm chk or something. OPSW says where we left.

Trace what happened in each RB.

PRBs have a RSV field at +60 and gives the name of the module represented by the RB.

Registers are saved in the next RB.

RTPSW1 & RTPSW2 fields hold error info at entry into recovery routine (which causes status at WLIC to be overlaid by SVC D and will mask original error). Recovery Routine (ESTAE) uses the failing RB to run under & when it decides to dump a new RB gets created and the status at exit gets stored in the failing RB (thus the overlay of original error).

Note any differences between RTPSW2 and WLIC. If equal, Recovery probably has not been messing around.

A LCE & CDE gets built for every module that gets loaded into the private area. Entries are deleted when the module gets deleted.

Scan TCB summary for non-zero completion codes. Also has pointer to RTM work area.

LPA/JPA MODULE means LPA or JPA location. In SVC dumps, almost always JPA.

Lab 1 - user dump.

Service Aids and Documentation

SVC dumps - written to Sys1.Dumpxx per
component request
operator request
slip request

May be formatted by AMDPRDMP.

Standalone dump - output of SADMP.
SADMP is generated through a simple stage 1 macro deck. This creates a program and produces an IPL text for a specified volume.

AMDSADMP parms:

1. IPL addr and type of device.
2. Volser of IPL volume.
3. console - specify master and alternates.
4. output - usually tape and addr. (for default)
5. type - High or Low speed. Low is to a printer.

It is a good idea to create a SADMP tape to use when you lose the disk holding SADMP If no consoles are working, the first tape drive to come ready will receive output.

Procedure:

Stop
Store Status hardware function to store regs + PSW.
IPL SADMP

Some IBM machines can automatically store status at IPL.

SADMP will query operator for a dump title and tape address. If a tape has an

internal label SADMP will wipe it out. Be sure to have a write ring in the tape.

Standalone dump tapes may be printed using AMDPRDMP. A better way is to use IPCS to view it online. AMDPRDMP options may be specified at SYSIN or from the console. AMDPRDMP can also copy dumps. Do not use IEBGENER. Using the SYSUT2 dd card is all that is needed.

AMDPRDMP options:

Global & Summary

CPUDATA - +store status areas
SUMMARY - w/ options
DAEDATA (SVC dump only)
SUMDUMP (SVC dump only)

Maps & Traces

CVTMAP
NUCMAP
LPAMAP
LOGDATA
TRACE - system trace
MTRACE - master trace

Component

ASMDATA
GRSTRACE
IOSDATA
SRMDATA
VSMDATA
VTAMMAP
JES3
TCAMMAP
RSMDATA

Specific Areas via PRINT option

avoid { NUCLEUS, CSA, SQA
ASID=
JOBNAME=
CURRENT - task (1 per CPU)
STORAGE= } starting and ending addresses.
REAL= }

Mandatory

END - if not present AMDPRDMP will go to console!

Do not use CPUDATA on SVC dumps because the global areas are not valid. The system keeps running as the task abends.

Do not trust the System Summary list of addresses and page numbers.

"JOB SUMMARY" - basic info on each job:

ASCB

ASID

Queues

TCB list with completion code.

"PROBLEM LIST" - located at end of SUMMARY Obvious problem address spaces are indicated.

"TCB SUMMARY" - best place to find problems. Lists TCBs completion codes and page number (often off a bit).

If SUMMARY TCBSUMMARY is also specified the display will be a bit more detailed. The other one is cleaner & easier to scan.

NUCMAP will list nucleus modules by EPA and alphabetically by module name. PSW address plus the EPA map can isolate to module and Csect.

CPAMAP produces comparable output for CPA modules. Majorname indicates that the name is an entry point within a module. If no majorname is present then NAME indicates a module. Entry points will not list a location and size, only EPA.

CPAMAP also lists "ACTIVE QUEUE MAP" of any modules that might be active. Check here. This will appear at the end of the section.

Diagnostic Approach

1. Collect status to define the problem.
   - SADMP - store status info
   - SVC - SDWA (p. 3 of any SVC dump).
   - Task - RTM2WA
   - Regs
   - PSW
2. Identify failing component:
   - Module
   - CSECT
   - component id
3. Determine failing + related address spaces.
4. Search on symptoms.

Key Indicators

CPU → PSW
LOCKS → CLHS
I/O → UCBs
ENQs → QCB, QEL, QXB
RSM → RCE, RIT
ASM → ASMVT
Dispatcher → GSMQ, GSPL, LSMQ, LSPL
Consoles → WQE; ORE

LOGREC

LOGREC records hardware and software failures on SYS1.LOGREC. The Logrec Buffer (LRC) accumulates records prior to output. Always check the LRB. Any recent errors should be of interest. IFCEREPI formats and prints SYS1.LOGREC. EREP parms stink.

The LOGDATA option of AMDPRDMP formats the in-core LRB. Always include this option.

Examine software records:

Abend ØAØD

Invalid SVC (issue SVC while locked or disabled).

Paging I/O error Ø28 - probably hardware error.

Restart Ø7x

Pgm Check

While locked and/or disabled an ØAØD may be issued to abend, but this will show up as an invalid SVC in LOGREC. The problem is <u>not</u> the issuance of the SVC. It is whatever happened to get the module to kill itself while locked and disabled.

RTM uses the SDWA to create the LOGREC entry for software errors. The record may be created by RTM on request of the FRR or ESTAE.

EREP can print an Event Summary ~~that~~ has a one line entry for every record on Sys1.Logrec.

If a record is created while the LRB is being written out, the missing Record Count is incremented and the data lost. This should rarely happen.

Logrec records:

TYPE

TIME

ERRORID - will match associated SVC dump if taken.

SYSTEM Abend Code

Jobname

Module

CSECT - verify before using
Recovery Routine
Component ID
PTF level of module - occasionally not trustworthy.
Hex Dump of Record - the SDWA.
VRA will be at 190. Key-length
data will come first in the VRA
if multiple types are indicated.
Byte 3 in VRA lists length of
key-length data area when present.
LRB has two sets of regs + PSWs.
The top set is most useful for program checks. The bottom set is for abends.
Compare PSW address to each register and see if one might be bad.
Note the flags:
"Disabled Rtn in control"
"System in SRB Mode"
"This Rtn percolated to" - when a recovery
routine percolates it often generates
a record. Look for the original
error, the one Not percolated to.
"Recovery Return Code"
04 = retry
00 = percolate
If "retry" is indicated, the second set of regs are updated prior to retry. In this case, they are probably past the original error.
LOCKS
Be sure to extract the key-length data from the VRA. This is useful.

"SUPER BIT STATUS FROM PSASUPER -
indicates which system component
is in control.

For SOFTWARE (SVC 13) TYPE records, use the second set of regs for conditions leading up to SVC D.

With SYSTEM FORCED SVC 13 due to abend 222, both sets of regs will be the same.

Note the general relationship between abend codes and SVC numbers. Abend of SVC ØA will result in abend codes of xØA, such as AØA, 8ØA, etc.

IEAVESAR is a Nucleus CSECT to repair data areas and generates a LOGREC entry. The VRA will indicate what he did to fix things but nothing in the record will help find who trashed a data area. This is a rare occurance but should be noted.

Calls to LOGREC are SVC 76 and if it appears in Trace should be investigated.

Module names may be related to component id through Debugging Handbook, Vol. 1, chapter 5. Use prefix to pick component id, then go to component summary and see if module name appears under that component. This is to check for prefix exceptions.

Global Status

Processor physical status
Dispatchability
Locks held
Recovery in progress
Execution mode - task
SRB
Wait

Current Activity

CSD - Common System Data Area

Formatted by CPUDATA. CVT +294 points to CSD.

+8 CPUAL - CPUs alive; bit mask
+A CPUOL - CPUs online. Count, not a mask.
+C SCFL1 - flags
'40' system non-dispatchability (Expect this turned on during dump)
+16 ACR - 'FF' = ACR in progress.

bit positions: 0, 1, 2, 3
1 0 0 0 = UP
1 1 0 0 = MP
1 0 1 0 = Diadtic SP
1 1 1 1 = 3084

SVT - Supervisor Vector Table

Used by Dispatcher. CPUDATA formats it.
CVT +354 points to it. Contains dispatcher status and queue anchors.

+1C - DSREQ - Dispatcher Serialization for MPs.
+160 - DACTV — One byte per CPU + Dispatcher puts 4x into appropriate byte to show he is running on that processor. x = CPU id. (40, 41, 42, 43)

LCCA - Logical Configuration Communication Area

One per CPU. PSA + 210 points to LCCA.

+4 CPUA - owning CPU addr (logical).

+20C SPIN - CPU spin bits. (loops c. 20sec. & then calls ACR).

Primarily used as a save area for the interrupt handlers (particularly Pgm FLIH).

+21C DSF1 - Dispatcher flags 1 & 2

80 - Local SRB

C0 - Global SRB

00 - Normal mode

+2B4 CRFL - Indicates why ACR is in progress.

+2E0 IOWA - IOS work Area ↑

PCCA - Physical Config Com. Area

1 per CPU. PSA +208 is pointer.

+10 CPUA - physical CPU address.

+18 PSAV - Virtual Address of PSA

+1C PSAR - Absolute Address of PSA

+84 RPB - External Call Buffer - to communicate among processors.

+88 EMSB - Emergency Signal Buffer

PSA - Prefix Storage Area

Formatted by CPUDATA. Pointed to from PCCA. Also, is @ Virtual addr 0. One per CPU. Holds fixed storage locations used by the hardware (PSWs for swaps at interrupt). Contains interrupt codes. (PICs & EICs are listed on p. 19 of Reach Card).

For SVC interrupts, the SVC number is stored. I/O interrupts cause the UCB address of the device causing the interrupt to be stored.
PSA holds MCIC (p. 27, Peach Card).
Beware of residuals in the PSA. Check the Trace Table for possible indications (Trace may be turned off!).
PSASUPER (+228) - which supervisor component is in control.

+380 Current FRR stack pointer
+384 Normal FRR Stack pointer
+420 PSW of last dispatched SRB
+468 PSW of last dispatched Task.
+49F Mode ↑ this PSW was loaded
- 00 - task (if task, then
- 04 - SRB
- 08 - wait (check PSW to be sure).
- 10 - Dispatcher running

CVT - Communications Vector Table
Pointed to from location X'10' or '4C' in PSA.
Only 1 CVT per system.

+94 MSER - Master Trace ↑
+234 ASCBH - Highest priority ASCB on dispatcher queue.
+238 ASCBL - lowest
+23C RTMCT - ↑ RTCT
+2C0 ASMVT - ↑ ASMVT

11-5-85

Locks

PSA + 2F8 (CLHS) is the Current Locks Held String.

Each lock on the system has a bit in this word to indicate that it is held. See Debugging Handbook, volume 1, p. 5-104. This is a very good summary.

CMSSMF, CMSEDQ, & CMS all share the same bit in CLHS. One or all three will be held at any given time; never two.

A LOCAL lock is one held by its own address space. The same lock is the CML when it is held by another address space. Holding the other guys CML insures that he won't be swapped out. IMS uses this frequently.

Three kinds of locks:

Global Spin
Global Suspend
Local Suspend.

Global Spin - key 0, disabled user. Issue SETLOCK to get lock, this macro will disable the user. User is disabled in order to finish quickly. If lock not available, spin occurs in IEAVELK (check spin bit in LCCA). GPR 11 Points to lockword during spin.

Global Suspend - enabled user upon request (because you may get suspended in IEAVESCA if the lock is not available).

Local Suspend - lockword lives in ASCB. If not available, requestor is queued to ASCBLSQH. PSALOCAL points to ASCB whose local lock is held as a CML, else Ø. ASCBLOCI points back to the holder of an ASCB's CML.

The RSM + Trace locks are shareable. User requests shared or exclusive control.

Lock Rules:

1. A requested lock must be higher in the hierarchy than any one already held.
2. Requestor of CMS locks must already hold its own Local lock.
3. Only one lock in a class may be held at a time by one user.
4. Suspend lock requestor is enabled. (May subsequently disable)
5. Spin lock holder runs disabled.

Each PSA maintains its CLHS, but the lockwords are shareable.

Lock Locations:

IEAVESLA -

DISP	CMS
SALLOC	VSMFIX
SRM	RSM
CMSSMF	VSMPAG
CMSEDQ	TRACE

The others live in various locations.

LIT Lock Interface Table. Pointed to from PSALITA (+2FC). This is a table of pointers. The locks in IEAVESLA have character eyecatchers naming each lock. The LIT is not in the Debugging Handbook.

PSACLHT (Current Lock Held Table) +280 is a pointer for each held lock. If the lock is not held (PSACLHS) then the pointer is residual. Procedure: check PSACLHS and then CLHT. Forget about the LIT. This is used by the lock manager to find locks and routines.

Contents of Lockword

00000000 = lock available

Global spin has CPUID.

RSM + Trace locks have shared/exclusive bit.

Local lock holder will indicate dispatchability of lock holder.

Lock problems rarely occur on MVS. Outside vendor packages are frequently guilty (Tested on UP, installed on MP). Beware of DB packages.

Scenario:

SETLOCK	Local	00 00 00 40
enabled		
page fault (RSM)		7F FF FF FF
dispatchable		FF FF FF FF
dispatched		00 00 00 40

ASCBCMLH points to the TCB or SSRB that was suspended while holding the Local or CML (+EC)

↳ 7F or 4F in left byte of lockword.

System Trace Table

System trace availability may be controlled by the TRACE operator command. CR 12 specifies tracing options.

Buffers are located in Trace private area + are page fixed.

PIC 16 occurs when trace buffer is full (TBVT) and causes switch to another buffer.

Type of entries:

Explicit

Address Space - PC, PT, + SSAR

Branch

Entries are variable length and should be formatted for use by AMDPRDMP.

Each processor has its own trace. When formatted they will be merged.

Explicit Trace records interrupts, dispatcher activity, I/O, and some misc. functions (Lock suspend, ACR, Trace option alteration, + user event).

Branches may be traced but are usually not (BALR, BASR, BASSM), because the table would wrap around too quickly.

Trace stops during a wait. This prevents a table of waits + Ext Int (as in SP1.3).

PTRACE macro permits user to record up to 4K of data in a trace record. For use by key 0, supervisor routines.

Each Trace entry may have unique fields. use the TTE map of Debugging Handbook for an explanation of the fields.

When looking at SVC trace entries, use Debugging Handbook, Vol. 1, chapter 5 to find the significance of Regs 15, Ø, & 1 which are placed in the unique field.

The middle digit of the timestamp increments at approximately one second.

SVC 3 and C will not show SVCR entries since they just branch out and do not explicitely end.

When formatting the trace table AMDPRDMP performs analysis and will flag obvious errors associated with I/O status. Good status is 08, Ø4, & ØC in word 2, bits Ø-7 of the Subchan. Status Word.

PC entries record the PC number: the number of the program we are calling. These may be found in Debugging Handbook, Vol. 1, p. 5-76.

The trace table ordinarily wraps in a second or less of time. Expanding the size increased the page fix requirements.

GTF

Since system trace is a bit limited, GTF may be used for documentation of known problems. Output may go to disk or tape. The type of events to be traced may be very explicitely specified.

Parms may be supplied through the IISYSLIB dd statement.

Service Aids, chapter 1 describes options.

GTF overhead is when GTF has to do I/O to record events - everything stops while it does this.

GTRACE macro is for key 0. supervisor routines who wish to write GTF entries. GTF must be looking for USER events.

AMDPRDMP formats GTF via the EDIT option (which can further select data). Pgm Int + SVC numbers are in decimal! In MP environments a GTF record from one processor may be split by another record from another processor. Formatting is controlled absolutely by timestamp!
GTF will not trace paging I/O.

Master Trace

Holds about 350 console messages for the default size of 24K. TRACE operator command controls Master Trace recording (+ size).

CVT+94 → MSRDA +8C → MTT ↑

MTT
+4 Current Entry ↑
+C End of Table ↑

Use MTRACE option of AMDPRDMP to format master trace buffer.

Sometimes ASM may communicate with the console I/O going through console services and thus would not appear here or Syslog.

SVC Dump Analysis

Generated upon request:
- failing component - SDUMP macro
- Operator - Cancel, dump
- Slip request

Two types:

1. synchronous - Task in wait for dump to complete and posting of ECB. Title line of dump will show module SVCDUMP for a synchronous.
   Failing task will be current task.
2. Scheduled - the address space has not detected the error (invalid request for a system service, for example). An SRB gets scheduled to post the dump task in the offending address space. Title line module is IEAVTSDT (should be the third TCB in the addr space).
   Failing task will not be current task.

Dump options:
- IEASYSxx - specify where dumps go.
- ADYSETxx
  - DAE
  - SVCDUMP(~)
  - SYSMDUMP(~)

DAE - goal is to prevent duplicate dumps.
ADYSETxx contains criteria.
- DAE = Start|Stop
- SVCDUMP | SYSMDUMP
- Match
- Suppress

Update - sys1.DAE updated when suppressed.
Records

DAEDATA - option in AMDPRDMP describing what DAE has seen in all the SDWAs and VRAs that it has looked out at and then suppressed. Particularly: "Symptoms Present For Use As A Unique Dump Identifier by DAE."

Analysis:

Dump Title - Appendix B of Diagnostic Techniques. Provides guidance on why dump was taken.
Errorid to associate w/ Logrec record.
RTM2WA
TCB Summary ("SUMMARY FORMAT" option)
Find earliest dump of error.
LOGdata.

Occasionally the errorid on the dump may not exactly match Logrec. Try using timestamp, ASID, and CPUID.

The address space's Local lock will probably be held to keep from being swapped out during dump.

The failing task may not be the failing task. Current will be the dump task.

0. Look up dump Title.
1. Check TCB summary for non-zero comp code.
2. Check failing TCB status.
3. Check RTM2WA Summary.
   A.) "EC PSW at time of error" holds:
      PSW, WLIC, Translation address.
   B.) SDWA address

c.) Previous RTM2WA. Find the first (oldest) RTM2WA (NO previous pointer) to examine the original error.

d.) RTM2WA Bit Flag Summary.

If no SDWA exists then RTM1 has probably not been active - as in the case where a dump is requested as opposed to an error intercepted.

4. Examine Logrec software records for error history. Always include LOGDATA when printing SVC dumps. Many problems may be resolved with logrec information and the VRA.

Sumdump is of marginal value. It records the global data areas at the time of areas. SUMDUMP command will read Sys1.Dumpxx with relatively accurate copies of PSA, LCCA, etc.

Sumdump register areas - prints 2K on either side of each register pointing to a location in storage.

Dump No. 1
Synchronous dump
Title - error in ADYTRNS, ADYIO, or ADYMSG.
SDWA created. Retry attempted.
Logrec entry made.
Completion Code = ØC4
Asid = 5
Compid = SC143

Error TCB @ 4FFØ7Ø CMP = 9ØØC4
RTM2WA @ 3F5480
Errorid = SEQØØØØ3
PSW = Ø7Ø D1ØØØ 8ØBC7AØ2
SDWA @ 3EBB4Ø
Abend module = ADYTRNS (LPA module)
" addr = BC41CØ
WLIC = 40004
Translation address = BCA474
No records in incore Logrec.
Trace on p. 63
VRA

Abend instruction = 5ØDØBA7A
ST D, B+A7A
B = 8ØBC99FA
A7A
BCA474

checks

Register B is probably bad.

BC99FA is in ADYMSG. (LPAMAP)
ADYMSG is in the CDE.
ADYMSG @ base JBB2110
Error @ $+36_{10}$ into ADYMSG
ADYMSG is not re-entrant & refreshable.

LPA active modules will not appear in the LPA map. Check the active list at the end of the section.

IPCS

Use IPCS to examine storage. Let AMDPRDMP do the formatting of CBs but avoid killing trees.

TSO commands for use with IPCS:

IPCSDDIR - initializes dump directory with seed record.

SYSDSCAN - displays SYS1.Dumpxx dataset information.

IPCS - invokes IPCS.

Features:

Locate & Display data
Analyze data
Symbolic manipulation
CLIST interface
AMDPRDMP verb exit
Copy dumps (& keep it machine readable).

Print file may be dynamically opened and closed; and verb exits may be routed to print without going to the screen. Then use split screen to browse verbexit output without leaving IPCS. ISPF interface if set up will also handle this.

Cist - display data (or "CIST TITLE" for dump title).
ListUCB - with device number
Find - locate data
Findmod - locate modules in LPA
Finducb - old. Don't use.

IPCS keeps a symbol table of addresses and labels. EQUATE may be used to assign a label. IPCS also builds symbols every time it finds something. Then it never has to search again.
CISTSYM will display the table.

DROPDUMP cleans up the dump directory when a dump is no longer needed. Entries are made by dsn. The symbol & map entries must be removed to re-use a dsn (like X35.Dump00).

IPCS Analysis:
ASMCHECK
COMCHECK - console
ENQCHECK
IOSCHECK - not very good for problems.
STATUS - store status data, dump title
SUMMARY ANOMALY - finds failing task.
RUNCHAIN - automatically run a chain by defining start and pointer.

idea: Write a CLIST to create a PDS and run each of these guys through Print output and place each as a member. Also PROMP verbs.

DSPL3270 - only supports 24 bit addressing.

PSW Analysis

11-6-85

The first step in analyzing a standalone dump is to examine the store status PSW for execution mode: Waiting/Running, Enabled/Disabled.

Wait – bit 14 on

Enabled – bits 6 + 7 on.

If either bit 6 or 7 is on, then the machine is considered enabled.

A disabled wait may include a wait state code in the instruction address field.

Four possible states:

Enabled Wait - most probably, a critical resource has become unavailable. Check out each component:

IOS - Queued I/O? UCBIOQF (-2C) ≠ Ø, IOSDATA, ACTVUCBS

ENQ/DEQ - GRSTRACE

RSM - Frame Shortage; RITSRMNT(+4), RITOPQF(+8C), RSMDATA

ASM - Outstanding I/O; ASMIORQR (+28) ≠ ASMIORQC (+ swap)

Dispatcher - Queued SRBs.

Console -

Subsystem -

Be sure to validate all components. Do not stop at the first error.

Disabled Waits - the easiest. Check the wait state code. Some are restartable. If no wait state code was loaded, then probably a bad PSW has been loaded. Look for PSA PSW area overlay. Also, check system mode (SRB, Task) and last dispatched PSW and PSAAOLD and PSATOLD to locate suspect.

If an overlay is found, set a SLIP.

If the PSW is trash, examine closely the registers (esp. R12) to find possible base regs and branch addresses (R14 + 15).

When diagnosing a wild branch, take a look at regs 12 + 14. Reg 12 may be your old base and reg 14 is usually the next instruction after the wild branch. This would be useful for an ØC1.

Dispatcher PSWs are at PSA +420 and 468. See if the dispatcher loaded the wait. One of these should be equal to the store status PSW. If not, the dispatcher is not guilty of the evil deed.

PSA +420 SMPSW — SRB PSW
PSA +468 PSWSV — Task PSW

Enabled Loop / Disabled Loop

Use store status PSW and registers and "play computer." Use the system trace to find modules and tasks/SRBs involved in the loop. Check LOGREC for any error which may have created the circumstances permiting the loop (Frequently a creative hardware failure).

A disabled loop will not record the loop itself in trace, but it will show events just prior to disablement.

Enabled loops are usually very large.

Disabled loops are usually very small.

Take particular note of the S bit in the PSW. S bit = 1 indicates Cross Memory Services. Data is moved in and out of secondary ASID when S = 1. Instructions should be coming from a commonly addressable area.

When S = Ø, no CMS is active.

The PSA mode flag (+ 49F) is only set by the dispatcher. Thus, if an I/O interrupt comes in while waiting, the flag will remain 'Ø8' until control goes back to the dispatcher. Therefore, in unusual circumstances it is possible that the mode flag will lie.

Global Resource Serialization

GRS runs in its own address space. It handles ENQ/DEQ and manages CTCAs. GRS runs even in a UP.

GQSCAN macro extracts data from the GRS address space.

A resource is requested via the ENQ macro specifying a major and minor name. The resource can be anything, even a control block. It is only protected if all users abide by the convention. The requestor must also specify shared or exclusive control, and scope as step (1 addr space), local (1 Proc. Comp), or Global.

With scope = step the same major/minor name may be held in multiple address spaces

For a Global request GRS queries the other machines GRSs to see if it is available. If so, it reserves it.

Parmlib member GRSCNFxx must describe the environment. If not present, GRS is for local use only. If incorrect, GRS will not come up (& neither will MVS). Few installations use GRS in cross system mode. Sometimes GRS won't come up if the CTCA is broken.

ENQ Control Block

Queue Work Block - describe request
Queue Control Block - describes resource
Queue Element (QEL) - system id
ASID
Info about request
Queue Extension Block - TCB
SVRB
Jobname

These CBs can be formatted with GRSTRACE.

GRS maintains separate queues for step, local, and Global requests. A QCB, QEL, and QXB is built for the first request. For subsequent requests, only a QEL and QXB are built, and they are chained off the previous QEL.

When the queues are formatted problems are easy to spot.

In coding the macro and in looking at the formatted CBs, the scope appears as: ——→

step = Local
local = system
global = systems

Check the scope, status, and jobname for each chain.

MVS major/minor names are documented in Debugging Handbook, Vol. 1, p. 5-108. MVS ownership is usually indicated by SYS prefix.

If GRS is not cross system, global requests will appear on the local queue. AMDPROMP will flag with an * any contention situations.

Scenario:

Master catalog has lots of shared users and then along comes a DISP=OLD. This causes an Exclusive enque request that will probably never be satisfied. However, all subsequent shared requests will also queue behind the Exclusive ENQ and work will grind to a halt.

Multiple resources can be requested on one ENQ macro. Do this. GRS will prevent deadlocks which might otherwise occur if multiple ENQ macros are used by one program. GRS will ENQ all or none of the resources specified by the macro.

GRS problems usually involve misuse of queuing, not errors in GRS itself.

Beware of a swapped out holder of an exclusive ENQ. SRM tries to avoid this.

– Lab 2 –

Problem - User abend 064
Failing Component - RTM2
Search argument
Sequence of events - Recursive abend

Failing TCB @ 8D8E88 (p. 31)
comp code = ØC4
PSW @ failure = 078D1000 00077406
ØAØD instruction @ 77404
Issued from ADSTHØIØ - some sort of archive system (SCI).
No SDWA.
RTM2WA Completion code = user Ø64 & Reg 1.

Reg 7 is bad. SØC4 module BCSQIFMT
@ displacement 9BØ
Check fiche.

Moral: Carefully check the RB structure.

IOS

OPEN fills in DCB and creates DEB.

GET update DCB + DEB, and branches into Access Method. The AM builds the channel program (simple; with Virtual Addresses). AM issues EXCP which expands to SVC Ø to enter EXCP module. It also causes ECB to be built and WAIT gets issued. AM builds IOB.

IECVEXCP IECVEXCP translates channel program (virtual to real address) and uses info in IOB. IOSB gets built (main IOS CB for process). Then issues macro STARTIO.

IOSVSSCQ builds IOQ and chains it to a UCB. All IOS modules begin with IOS,

IOSVSSCH (Start Subchannel) - check UCB for busy. If yes, it stays queued. If not, ORB is filled in and SSCH instruction is issued. UCB gets marked busy. The I/O is underway as far as IOS is concerned. This is called IOS front end.

Back end:

Starts with I/O interrupt. FLIH passes control to IOSVSCIH. Test Subchannel (TSCH) instruction to fill in IRB. Check status is done by IOSVIRBA. Two actions (happen in parallel):

1. Post status (inform user) - runs as SRB in user address space. Updates IOSB, and posts UCB.
2. Redrive (is their any more work for the device). If the device has a problem, the UCB will show a queue of IOQs.

Path selection is handled by the Channel Subsystem. Also, it picks which CPU to interrupt upon completion.

HSA (Hardware Storage Area) holds the subchannels (which represent devices). One subchannel per device.

On return, PSA at B8 and BC get updated with subchannel number (which is not device number) and interrupt code (which will always be UCB address). IOS uses these to begin building the IRB.

IOS handles I/O interrupts disabled, and when done issues TPI to see if more are pending. As a result overhead is reduced.

IOQ

+4	Chain	↑ next IOQ
+8	IOSB	↑ to IOSB
+18	UCB	↑ UCB
+1C	ASID	ASID

Issuers of STARTIO macro are called IOS drivers. They are:

VTAM	'04'
VSAM	'03'
JES3	'08'
ASM	'0E'
IOS	'00'
EXCP	'02'

} IOSDRVID field

> See System Logic Library, Vol. 8. pp. IOS-40+41 for module flow. Also, IOS pp. 19-38 contain good diagnostic hints.

Sys Logsirol 8
pp 105 40-41

ORB - interface between IOS & channel subsystem.

+0	IP	- Int. Parm (↑UCB)
+4	FLO-1	- Flags (ccw format, suspend)
+6	LPM	- Logical Path Mask
+8	CPA	- ↑ Channel Program

One ORB per CPU and used repeatedly.
The ORB lives in IOWA.

UCB - Unit Control Block (tracks one device).

-2E	MIHTI	- MIH Flags
-2C	IOQF	- ↑ First queued IOQ } will be the same if only 1 IOQ is pending.
-28	IOQL	- ↑ Last queued IOQ }
-4	IOQ	- ↑ most recent active IOQ

Active and First will not usually be the same. IOQF is always pending. UCBIOQ is indicated valid by flags at +6. If FLAG1 is '08' (SSCH), '04' HSCH, or '02' CSCH, then IOQ is valid. Else UCBIOQ is not valid.
If IOQF and IOQ are equal, the the IOQ is a redrive I/O and the IOQF field has not been updated.

+4	CHAN	- Device number (binary)
+6	SFLS	- Flags1, FlagB(error flag).
+D	Name	- EDCDIC device number.
+10	TYP	- Device Type.

IRB Interruption Response Block

+Ø SCSW - Subchannel Status Word (p. 24 Peach Card).
+C ESW - Extended Status Word. (Timing info) see Peach Card and P00.

One IRB per CPU.

Status of an I/O operation:
UCB -4 → IOQ+8 → IOSB

+Ø Flags A, B, +C
+4 DVRIO - Driver id (who created)
'ØØ' IOS
'Ø2' ExCP
+D COD - Completion Code
'7F' - good.
Anything else indicates a problem.
'4x' - disasterous error.
+10 UCB - ↑UCB
+18 STATUS - SCSW (Subchan Status Word)
Word 2, bits Ø to 15 of IRB.
See p. 24, Peach Card.
04ØØ
08ØØ } Good status.
0CØØ
Ø2ØØ - unit check
+2C SNS - Sense data describing error exactly.
First two bytes go here. Debugging Handbook, Vol. 1, chapter 4. May also appear on msg IOSØØØI.

+34 ERP - ↑ ERP Work Area - for all sense.

ERP	EREP Work Area
+18	Device Number
+20	Sense data

IOWA — IOS workarea (LCCA + 2E0)

One per CPU.
Holds ORB + IRB.

+244 ↑ UCBLOCK word (which is @ UCB - 8).
+24C ↑ Active IOQ

IOSDATA is the AMDPRDMP option to format IOS control blocks. Don't worry about the IOS work Areas that come out. Check each CPU's IOWA.

IOSDATA ACTVUCBS - prints all active UCBs. Queued IOQs will be formatted immediately following the UCB followed by the IOSB.

Dump 3 —

Prob Description -

Failing Comp - IOS

Search Arguments - Abend ØC1, , Reg 8, IGEØØØI

Title: error occurred in IECVERPL or an ERP that does not have a recovery routine.

Comp - IOS

DAE data:

ØC1

Reg 8 is possible base for PSW addr

Error TCB @ 8FDE4Ø

Error PSW FBCD12

Data at PSW addr - is bad.

PRB - WLIC = 02 0001

- RSV = IEAVARØØ

Bad instruction executed is ØØØØ

at location FBCD1Ø (which is Reg 8!).

Data around this location is EBCDIC =

Ø38_...

Errorid: 000053

Pgm Check, Enabled R13 in control.

IEAVARØØ is in LPA @ 19CAØØ with EPA of 19CA120 for a length of 1D60. (end @ 19E76Ø). No register could be a base for this module.

Master Trace has messages:

IEF170I

IEA995I

No system trace.

LOGDATA (p. 27) -
Pgm Check entry
IECVERPL - Mod + CSECT
ERPLESTA - FRR
No hardware records.

FBCDIØ is the CVT address.
Reg 14 = 1AB8E14
Reg 12 = 1AB8000 addr of IGEØDØØI
Reg 14 is @ +E14 into module.

Moral:
For ØC1, suspect a wild branch and investigate Reg 12, + Reg 14. Look for possible base reg of PSW addr to identify bad reg. Check instruction prior to Reg 14 + see if that would produce the result.

— Consoles —

Console manager lives in its own address space. WTO causes your msg to be moved to a buffer in this address space and then console manager queues it for a console. If the message cannot get out the buffer is not freed. SRM will swap out folks when the console buffer resource becomes scarce.

No formatting routines are available with AMDPRDMP to print the Console CBs. The console address space must be identified and dumped.

CVT +64 ↑ UCM Base

+18 ↑ First WQE
+1C ↑ ORE (used for outstanding messages).
+20 ORE LIMIT
+2E WQE Buffer Limit > if equal, problem.
+34 WQE in use count
+38 ORE in use. If equal to Limit, then problem exists.
+3C ↑ Last WQE
+48 ↑ First UCME (1/console)
+4C UCME Length each (usually 50_{16} bytes)
+50 ↑ Last UCME.

WQE Write Queue Element

+0 ↑ next WQE
+4 Flags
Regular - one text line
major - first line of multi-line
minor - subsequent lines of multiline msg
+1F - Text of msg

ORE Operator Reply Element
If an ORE exists, go find associated WQE & find the outstanding message.

UCME

One exists for each console.

+C ↑ UCB

+18 Status Flags - output pending should be noted.

+1C ↑RDCM - Device type
Only built for screens. If Ø then console is a printer. Messages should not queue for CRTs but may for printers.

+24 ↑CQE Pointer for pending output.
If Ø, no output pending.
If ≠Ø output may be pending.

CQE

May hold up to 50 WQEs and then point to another CQE. CQEs are tables of WQE pointers w/ flag byte in high order position.

UCM Base and UCMEs live in Nucleus. WQEs and CQEs live in Console private area.

UCME +1C → RDCM +Ø → TDCM

+30 ↑ Screen image
+118 Screen status } K.S specs.
+128 Communication flags } Is it in Roll mode.

Debug:

Any OREs?

WQE # = WQE Limit?

ORE:

System ORE

TCB

WQE

WQE:

WQE Chain (is chain damaged)

UCME - RDCM - TDCM

UCB (FLGB)

When 80% of Buffers are gone a msg will be issued. (Also at 100% out).

D. C will display buffer usage by console.

K. L To re-route message.

K. M alters number of buffers.

Dispatcher

11-7-85

If Global SRBs are outstanding no other work is likely to be dispatched.

Source places data in common area, issues SCHEDULE to request a service routine (causes SRB to be built) + data gets moved into different address space.

Service Routines may not issue SVCs. They are often disabled. The IOS post status routine is a frequently used service routine.

A Global SRB is higher priority than a Local SRB; otherwise no difference. Schedule is key Ø, supervisor macro.

When SCHEDULE is issued, the SRB is first placed on the GSMQ (SVT + 2Ø). When dispatcher checks GSMQ it will move requests to GSPL (Global Service Priority List) (SVT + 24).

The local queues are anchored out of an ASCB. LSMQ is pointed to by ASCB + DØ. LSPL is from ASCB + D4.

AMDPROMP will format these queues. All queues should be empty since Service Routines are very quickly executed.

SRB - holds target ASCB (target address space).
- EPA of Service Routine. This must be in common storage and is therefore mapped by NUCMAP or LPAMAP and is identifiable by EPA.

If CPU is in "Global Service Request Mode" and only one SRB is on GSPL, then that SRB is running.

For queued SRBs:
All same target ASCB?
All same EPA of service routine?

Check CSD system non-dispatchability bit.
If on, examine only exempt address spaces.
If off, check address spaces.
Dispatcher checks lots of ASCB fields, particularly
+67 FLG1
+72 DSP1
+73 FLG2
+64 AFFN - CPU affinity. If this is different from the CPU upon which the dispatcher is running then the ASCB is by-passed.
Note: the queues of ready ASCBs is maintained by SRM. The dispatcher just runs queues.

TCB

+20 FLGS4 } possible non-dispatchability flags.
+21 FLGS5 } look here to see why the dispatcher
+AC SCND4 } does not want to run this TCB.
+F2 - CPU affinity.
+114 FBYT1 - Flag
'20' Non-disp
'01' Page fault while holding local lock.

SRB Service Request Block

+25 FLG5 - Locks required. If not available, will not get dispatched.
If a page fault occurs, the SRB gets turned into an SSRB to hold Regs + PSW for use when page comes in and execution resumes. +58 Regs
+98 RPSW

Why not running ?

System wide non-disp
AS failure
CPU affinity
Local Lock needed & held
ASCB valid
STATUS stopped SRB
More ready task exist in addr space than CPUs.
TCBs non-dispatchable. (FLG54, 5, & SCNDY)
RB waiting
TCB suspended for lock or PIC 11.
CPU affinity.

Dispatcher Serialization -
Avoid collisions of SRM & dispatcher
Dispatcher Lock - spin.
Intersect -
Global - Requires Dispatcher Lock. (Global SVT +1C)
SRM updates ASCB disp. queues.
Local - SRM updates TCB dispatcher queues.
Requires Local Lock.

SVT +160 - dispatcher active flag. One flag per CPU. The CPUid (logical) is placed in the flag when the dispatcher is active.

Dispatching:

Runs disabled

Look for:

Special Exits: (branched to)

ACR

Vary CPU

Timer Recovery

Global SRBs:

GSPL (LCCA +DF2 = 'C0')

PSAAOLD = target ASCB

TOLD = 0

PSA +420 - build PSW to run SRB.

PSAMODE (+49F) = "04"

LPSW from location 420.

GSMQ - try to move contents to GSPL.

LSMQ - not used.

Examine Addr Spaces in Priority Order:

LSPL (LCCA +DF2 = '80')

+ then same as above.

Local Supervisor Routines - routines holding the Local Lock. Branched into from dispatcher. Used to change TCB priority.

Asychronous Exits - STAGE3 exit Effectors - rare.

TCB - examine TCB queues for ready work.

(LCCA + DF2 = 00, normal)

PSAAOLD = ASCB

TOLD = TCB addr.

PSA + 468 - Build PSW for execution

PSAMODE = "00"

LPSW from location 468.

WAIT Task - Before loading the wait

task, the entire process is repeated with enablement for interrupts. SVT +160 is set to CØ to indicate the dispatcher is enabled while running on the indicated CPU.

Wait (CCCA +0F2 = ØØ)

PSAAOLD = Master ASCB

TOLD = Ø

PSASUPER = Dispatcher in control.

PSAMODE = '08'

PSA +468 builds wait PSW. Ø7ØEØØØØ~ CPSW of +468.

All GPRs will be zeroed.

- CAB 4 -

Use worksheet -

Problem description -

Module -

Component -

PSW addr = 1A40E84 is in module IEAVTSCP looped stopped at offset E84 into module.

SC1CM comp.

GSPL first SRB EPA = 105B29Ø this is in IEAVTRER.

Take a look at current FRR address to identify component in control (if ≠ CØØ).

RSM — Real Storage Manager

Maintains the tables used by DAT for address translation. Manages real storage. RSM maintains a pool of available frames to be used for page in operations. If low, RSM notifies SRM to swap out users.

RIT — RSM Internal Table

Principal RSM control block.

CVT +164 → PVT +4 → RIT

Beware Debugging Handbook info regarding RSM.

RIT

+4 Flags - SRMNT low frame flag = '40'

+24 PFTE Queue anchors

Available -

SQA -

CSA -

+8C PCB Queues - Each PCB represents a page-in request to ASM.

+D0 ↑ Page Frame Table - describes each frame in system.

+13C ↑ RSM Trace Table - never seems turned on.

RSMDATA is the AMDPRDMP option to get RSM control blocks formmatted

RCE — RSM control & Enumeration Area

+88 Total Avail frame count
+1C OK " " thresh-hold
+80 Low " " limit

If Total = Low, sysevent is issued to notify SRM to swap users (Bit in RIT is set). Swapping will continue until total = Okay.

+78 Total Fixed Frame count - once fixed RSM can't do much with. If high, a problem may exist. Batch should have about 12. IMS/TSO etc. may have lots. Get ballpark figure from Omegamon for comparison.

PFTE — Page Frame Table Entry

+Ø Forward pointer
+4 Backward "
+8 Queue ID - in use, available; SQA, common, etc
+D UIC (about 1.5 sec. SRM increments). Count is zero when referenced.
+E # Times Fixed (# of users requesting this frame to be fixed).
+12 ASID of frame owner. All `FF' indicates common storage
+14 Virtual address of frame
+18 ↑ last PCB used to manipulate (page in) this frame.

PFTEs are not ordinarily formatted by RSMDATA.

PFTE counts are reported and should be sufficient.

PCB Paging Control Block

When RSM needs a ASM service, it get a PCB from available PCB queue, fills it in, and passes it to ASM.

+0 ↑ forward PCB
+4 ↑ backward "
+8 Queue id
+C FID - Function Id - what the request is for.
+18 ↑ Page RAB
+20 ↑ TCB/SSRB
+28 Virtual Address of data
+2C ↑ PFTE for this frame
+34 ↑ XPTE - location on DASD for data.
+40 AIA - used by ASM.

If we are low on frames, why?
ASCBFMCT - allocated frame count for an address space.
ASCBRSM - ↑RAB

RAB
+4C PFTE Queues - Pageable
- Fixed (not for fixed frames)
- Deferred

SRM control:

OUCB - 1 per ASCB (+90) - kept by SRM.

Good to find why an addr space is swapped out.

'05' Real storage shortage detected.

+29 SRC - Swap Out reason code - why originally swapped out

+6A TWSS - Target Working Set Size. Won't swap in until this number is available.

RSM Health indicators:

Available Frame Counts in RCE:

AFC - available

PBAFC - perferred below

DFC - double

POOL - all Available frames

BELPL - available below 16 MB

RITSRMNT - SRM notification of shortage.

RCE

AFCLO - low threshold

AFCOK - ok "

DPQ - Deferred PCBQ (frames unavailable)

Fixed frame counters.

RIT

PFTE Queue anchors.

Address Space info

PFTE + PCB Queue anchors

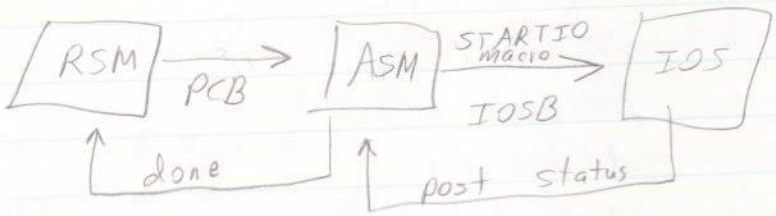
Fix control queue anchors

ASCB frame count.

RSM was re-written for XA. Prefix IAR. Now has eyecatchers at +0 in CBs.
For error, check VRA in SDWA for useful info. See debugging section in Logic Manuals. Look for reason code as well as abend code.

ASM — Aux. Storage Manager

Sits between RSM and IOS. RSM makes requests. IOS does the I/O.



ASM also controls all VIO.

ASM doesn't often fail but provides info on outboard paging errors.

ASMVT (Formatted by ASMDATA).

+4 SART ↑
+8 PART ↑
+14 EREC ↑ - Bad slot record address.
- No hardware Logrec entry for page pack - a software entry and update to this table.
+18 MSGBF ↑ - Message buffer ASM would like to display.
+28 IORQR - page requests received count > should be equal
+2C IORQC - page requests completed
+30 - Swap req received >
- " " completed

+74 VSC - Allocated VIO slots
+78 NVSC - " non-VIO slots
+7C ERRS - total bad slots on all local page data sets.

Bad Slot Record Table -
+Ø Current ↑
+4 First ↑
+8 Last ↑

PART Paging Activity Ref. Table

+8 EUSE - Parte's in use
+22 FLG1 - Has flag indicating no more room for VIO. Why?

PARTE - 1 per page dataset possibly allocated.
+8 Type - PLPA, Common, Local
FLG1 - in use or not.
+A NN - PARTE number
+C DEIB - start/end of dataset (needed to convert slot to CCHH).
+14 SLTA - avail slots on this ds.
+18 ERRCT - count of permanent I/O errors. At 'AF' system drops dataset. Probably lose system to Ø28.
+20 PATP - ↑ Pat map - bit map of page slots in use. Good for eyeballing quantity in use.
+24 PCTP - ↑ Performance Char Table. ASM uses to find the fastest device. Indicates device type of volume.
+2C UCBP - ↑ UCB of device.
+3E REQS - Count of outstanding req. for this ds. For Req/Comp inequality, will identify where.

IORB

+4 ↑ next IORB
+8 ↑ Paging channel program
+C ↑ IOSB in use. Created by ASM and passed to IOS.
+20 ↑ PARTE.

Swapping is similar with PART replaced by SART. Swap sets are the quantity instead of page frames.

ASCB +78 - # VIO slots for this addr space
+7A - # Non-" " " " "
+14C - ↑ ASM Header (not much data there)

LOGREC recording is the responsibility of the IOS driver. This is why we get only what ASM wants us to know - a software record of Abend Ø89 or ØC8. Most ASM errors are device related.

Lab - Sample dump

Worksheet
Prob. Description
Device(s)

WQE buffer count = buffer limit
UCME @ FDCFØØ in output pending
RDCM @ FEØD68
TDCM @ 9B7FØØ

Roll Time is 46 seconds - IPL default value.

MP Considerations

11-8-85

MPs share memory but have different locations zero. The Prefix register contains the absolute address of a processor's PSA. The first (high) five digits of every real address are checked for zeros. If they are zero, the contents of the prefixing register are added to the real address to form an absolute address. Absolute location zero is not really used as a PSA.

The processors communicate via SIGP instruction which causes an external interrupt in the target processor.

Serialization:

CS/CDS/TS - compare and set lock in one instruction.

Locks - Used via convention. - Main form of serialization on MPs.

ENQ/Reserve - Another convention.

Intersect - Dispatcher & SRM use to manipulate the dispatcher queues. Like private locking system. Check SVT.

Wait - one CPU forces the other CPU to go into a disabled wait. Used for error recovery.

Unique control Blocks:

LCCA/ PCCA/ PSA/ IOWA

MVS/XA can support up to 16 CPUs.

SIGP orders:

Two forms -

Remode Pendable (External Call) - generates an Ext Int of 1202. Check PCCA XC + EMS buffers. SIGP sets bit. Ext Int causes CPU to look at its PCCA buffer to find out what to do. RPSGNL is the macro.

traced as EMS

Remote Immediate - Emergency Signal generates an Ext Int of 1201. Traced as a CALL. Used to notify other CPU of a machine check in the originator. Macro = RISGNL.

Direct Signal (DSGNL) - directly invoke hardware functions like stop + store status, CPU reset, etc.

For MP problems, always check each processors' LCCA, PCCA, and PSA. Find out which processor is ailing. The one with the symptoms may not necessarily be the one in error.

Processor Failure -

ACR - (CSD flag, Trace entry, Logrec, PCCA)

Malfunction Alerts - dying gasp. Code 1200 Ext Int.

Resource Contention -

Locking - LCCA Spin bits

ENQ -

Parallel Processing w/o Serialization -

Corrupt data areas

unusual abends

impossible to recreate

Trace table is about the only documentation.

Lab 6 -

Title : error occurred while subchan status was being processed IOSVIRBA, comp SC1C3

DAE data:

5ØC4

Reg 9 possible base

ASID 4D

JBB 2110

Abend PSW = 1ØC49CC

Logdata -

Jobname = JRT8ØAØ6

Virt Addr of translation exception = 23Ø88681

Instruction previous to PSW: p.73

585ØCØØ8 L 5, C+ØØ8

C = FBA5ØØ
008
FBA5Ø8

47FØ9CBA B R9+CBA

R9 = 10C4856
CBA
10C5510 - okay

7
12
21

PSW inst = 58CØ517C

R5 = 2300650E
17C
2300668A

14
12
26

Reg 5 is bad.

IOSVIRBA @ 10C49CC
10C4850
+17C = offset in module.

Moral:

Backing up one instruction from PSW is not always right. Is it reasonable?

— SLIP —

SLIP command can come from console, or TSO
cmmdxr with OPER privilge. SCIP sets PER,
PER is a hardware function:

Successful Branch (SB)
Instruction Fetch (IF)
Storage Alteration (SA)
Reg Alteration

When detected, a Pgm Int 80 is generated.
PER causes performance degradation.
Stor + Reg alterations have very little
overhead.

PSW bit 1 enables/disables PER. This
permits PER for 1 Addr space
CRs 9-12 are used to control PER.
Only one PER trap may be on at once.

At interrupt:

PSA + 8E PICOD
PSA + 96 PER Int Code
PSA + 98 beginning of instruction causing
PER interrupt.

SCIP commands:

SET -
MOD -
DEL -
Display -

Try to make the slip trap as narrow as
possible.

SLIP may specify that tracing of the situation occur to the active GTF.

Multiple SLIP traps may be in use at a time. SLIP interfaces with RTM or PER.

Addresses may be specified indirectly.

CVT +250 ↑ Slip Header (SHDR)
- +4 PFC — If Ø no enabled traps
- +C FWD — ↑ First SCE
- +10 BKWD — ↑ Last SCE

SCE
- +4 SCVA
- +C FWD
- +10 BKWD

SCVA
- +4 LN

A store status after Actron wait stores RTMs PSW + Regs. So, look at PSA +40C for Slip Work Area for original error info.
- +Ø Flags for determining type of data at +C
- +1 Logical CPUID
- +4 ↑Regs @ error
- +8 ↑PSW at error
- +C ↑Variable work area.

For Action = SVCDump

CVTSDBUF (CVT + 24C) points to SDUMP Buffer:

+0 = Type
+4 = Environment. Determines contents at +5C
+14 Regs
+58 PSW
+64 PASID
+6A SASID

Documentation:

MVS Diagnostic Techniques
MVS System Commands
TSO: SPL

Dumps 7 + 8
Addr of regs + PSW:

#7 SADMP

PSA+40C = Slip Work Area = 9FE3A4 (p. 141)
+4 = 9FE1C8 = ↑ Regs
+8 9FE218 = ↑ PSW

#8 SVCDump

CVT + 24C = SDUMP Buffer = 801BC2000 (p. 201)
+14 Regs BC2014
+58 PSW BC2058