Vol 6

National® Brand

College Ruled White Paper

Single Subject

31-987

80 Sheets/11 x 8½

Dennison National Company, Holyoke, MA 01041

Brian Page (301) 992-0090

VM/SP Installation

11-17-85
Claudia Dewey

EastFSC - route prt destination

VM Overview

→ used to be standalone

1966 CMS (Cambridge) and CP/40 were merged.
1967 Released w/o support. (Rel. 6. - public domain)
1972 VM/370 - licensed
1980 VM/SP
1982 HPO

VM under VM cannot test AP/MP code.

Components:

CP - DMK prefix
CMS - DMS
IPCS - DMM (IPCS) or DTV (IPCS/E)
With ver. 4 only IPCS/E is available, and is no longer an extra cost item.
RSCS - DMT

Distribution

Starter System - 3 tapes:
1. Object
2. PUT
3. Actual system (default addresses)

Pregenerated SIPO or SIPO/E (custom).
Two-year backlog.
Merged Product Tape

Architecture

Jim Turner

CP runs disabled with translate off, in key Ø.
Only the HPO module which moves pages below the line can run with translate on.

CMS runs in a 360 environment using a BC mode PSW.
(360 machines only had 6 channels).
PSW actually held masks for each channel!
Also, the actual interrupt code!

> TOD clock synchronization - a continuous duty?
TOD - 64 bits
bit 51 incremented each microsecond
bit 31 changes each 1.0486 second
The TOD clock will role after 143 years.

CPU Timer decrements only while CPU is running.
used for timeslicing.

Interval Timer - decrements only while CPU is running.
used to time dispatch time slice end.

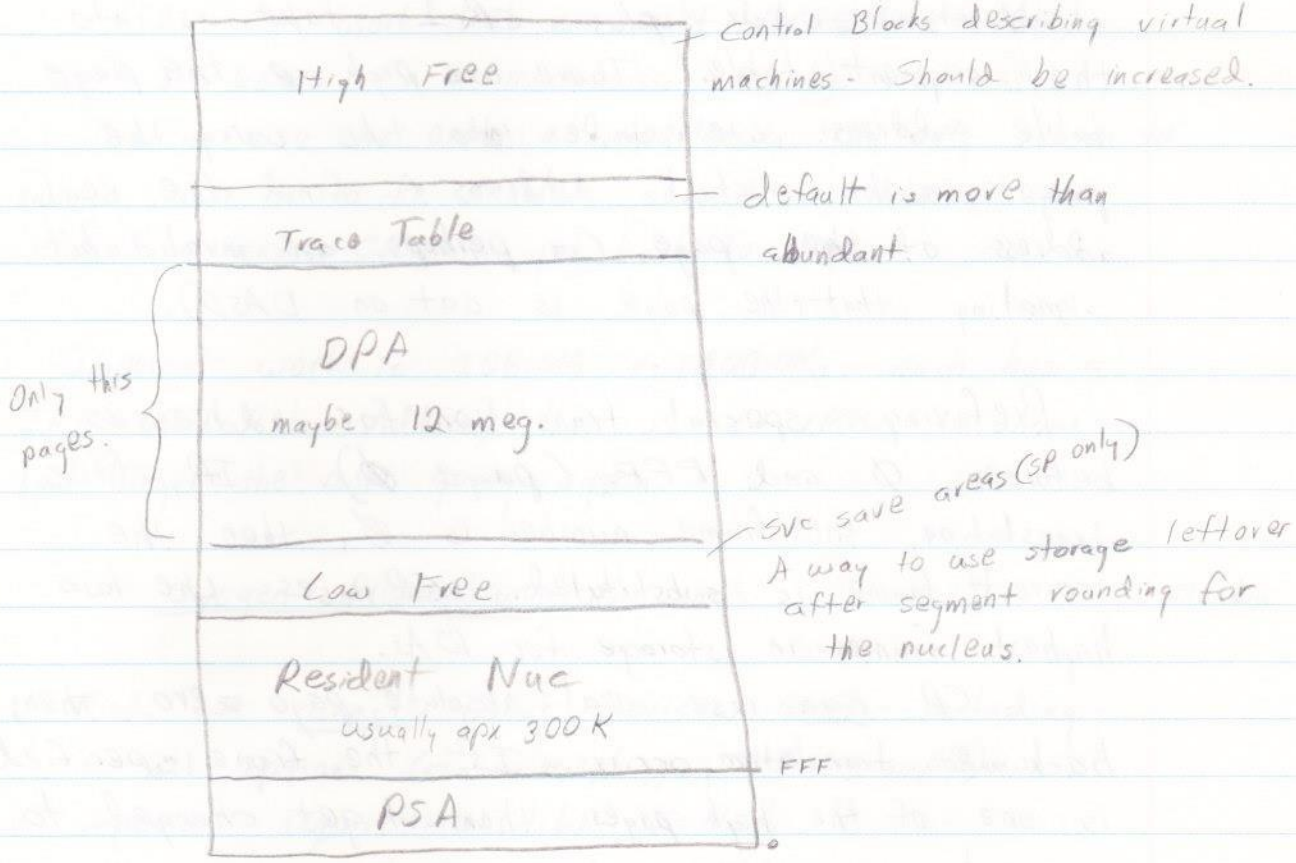
Restart interrupt causes CP to dump.

> Fail ARCØ12Ø - what would prevent an I/O interrupt or machine check from occurring with PSW bit 14 on?

VMCF & IUCV signal via External Interrupt.

VM/SP uses SIOF & deffered condition codes.

DAT takes an average of 64 machine cycles. TLB cuts this dramatically.



Each segment table entry maps 64K. Last 3 bits are Ø & 50 may be used as flags.

29 = seg. protect (HPO)
30 = common segment
31 = segment invalid

Page table has 16 entries (16 x 4K) = 64K (1 segment)
Bit 12 is page invalid - paged out (PIC 11).
Also holds Extended storage address - holds the two extra bits needed to go above 16 meg. (+ since CP itself does not use page tables it cannot go above 16 meg).

Virtual addr plus CR1 take us into the segment table. There we pick up the page table address. We index into this using the page in the virtual address + find the real address of the page (or perhaps an invalid bit signaling that the page is out on DASD).

Prefixing - special translation for addresses between Ø and FFF (page Ø). If, after translation, the frame number is Ø, then the correct frame is substituted. CP uses the two highest frames in storage for PSAs.
CP can use real absolute page zero. Then, backwards translation occurs. If the frame specified is one of the high pages, then it get changed to zero!

SIO is slow. The processor stops until the channel has a path to the device. SIOF immediately results in a CC of Ø.

CP runs disabled. Virtual Machines run enabled.
CP runs in supervisor state. Virtual Machines run in problem state.
CP runs DAT off. Virtual machines run DAT on.

Three units of work in CP:

1. Important CP (IOBLOK + TRQBLOK)
2. CP work that can wait (CPEXBLOK + CPEX)
3. users (+ the machines is enabled).

This is also the order of importance. The only reason to run a user is to be enabled so that an interrupt can come in and CP can get back to work.

Important work is IOBLOKs + TRQBLOKs. Each has a IRA (Int. Return Address) that is a LPSW to show how to get the user going again.

Sysgen Preparation

Claudia

1. Find out <u>exactly</u> what devices exist - including all options (controller plugging, for example).
2. Special requirements like V=R?
3. Microcode assists? PMA, VMPA
4. Processor configuration: UP, AP, MP
5. Mode of operation: SPM?
6. What software will run under VM?
   - VS1 or VSE handshaking?
   - OS Compilers (must be ordered separately).
   - 3705? VM only supports EP, no NCP or SNA.
   - OEM software - like GDDM (coded as saved systems in SNT)

7. Understand VM restrictions:
   Sys Planning Guide - Appendix D
   Alt Path or Res/Rel?
   Shared DASD between MVS & VM
8. Paging volumes adequate?
9. Will CMS have shared segments.

Plan DASD space allocations.

System definition files:
- Directory (two copies) - lots of work.
- DMKRIO
  - SYS - defines paging, nucleus, operator, etc.
  - SNT - " saved systems & EP
  - BOX - logo
  - FCB, UCS, DMLUCB, DMKUCC } Printer controls. Need only be changed for special printers.
  - PIA
  - PIB

Format disk space:
- IPL FMT

Service Programs

Service tape files:
1. ICKDSF - analyze & re-assign if neccessary.
2. CP Format/Allocate Pgm (defines in 4K blocks)
3. DASD Dump Restore Pgm

CP Disk cylinder Ø must be specially formatted.

IPL DSF out of the reader. DSF is not used to format under VM. It is only used to check for bad spots.

Analyze - hardware problems
Init - volser
Inspect - surface check
MAPALT - only for FBA devices
REFORMAT - replaces volume label info.

col 2
INIT UNITADDRESS(cuu) (150/160) NOVERIFY (don't check current label) VOLID(serial) (new volser) - NEWVM
OWNERID(owner) VTOC(cyl, trk, extent) - (0, 1, 1)
optional → MIMIC(MINI(cyls)) DEVICETYPE(type)
mimic 40 cyls, not a whole volume

INSPECT UNITADDRESS(cuu) VERIFY(serial, owner) -
NOCHECK ASSIGN TRACKS(cccc, hhhh) -
PRESERVE

On S disk is IPLable version of DSF:

XEDIT INIT JOB A
GET IPL DSF S2

Format/Allocate must be run against any CP owned volume - Spool, paging, directory, sysres, etc. Cyl Ø, Trk Ø, Rec 4 has bit map indicating use of device

Ø1 Permanent Space (Non-CP use, Mdisks)
The rest of 150 must be allocated as perm.

Ø4 - inactive Directory
ØC - active directory
Only the Directory can live at cyl Ø.
Ø2 TDSK - temporary disk.
Ø8 - page (preferred only)
ØØ - Temporary - spooling + non-preferred paging
1Ø - Dump - 200 cyls.
- OVRD - over-ride file. Permits the class of a command to be changed.

Put the nucleus at the tail end of the pack. (Only requires about 3 cyls).

Dasd Dump Restore

Copy CP NUC from tape to disk.
DDR files from release to release are not always compatable. Therefore, copy DDR onto the tape ahead of the DDRed file.
The CP + CMS DDRs are not necessarily the same.
RESTORE NUCLEUS to get starter system.

Second Level Sysgens

Planning Guide has the devices defined in the starter system.
Volser of second level disk must be the same as the CMS label of the minidisk.

EDVMIØ2 - userid
TURKEY password
CCCVME

VM screen - D PVM
TSO - EC

Directory

11-18-86
Claudia

The directory defines the virtual machines.
Always located on sysres - a CP owned volume + pointed to from VOL1 label (Cyl Ø, Trk Ø, Rec 3).
The Direct file may be written out with DMKDIR using CMS DIRECT command, or standalone - IPL DIR. (SP PUN CONT, Pun Pgm, Pun Directory, IPL C).
Use the CMS command.
The DMKDIR pgm only checks syntax, not minidisk overlaps.
* in column 1 is comment line.
No continuation characters required.
Maint must have access to sysres to write out.
Spool + Page space can be allocated to a nolog userid + then Maint can link + do work if required.

The options Realtimer + ECMODE are necessary to IPL second level.
ACCT optron - collect accounting info for machine.

DMKSYS, DMKRIO, DMKSNT

DMKSYS - describes sysres.

Describes all CP volumes, operator console, storage size, page + spool, monitor options, password suppression option, accounting parameters, system id (lower right of screen), + ...

SYSOWN - must be 1st. Specifies up to 155 CP-owned volumes by VOLSER.

SYSRES - 2nd macro.

volser, address (existance of alternate volume), devtype, starting loc of NUC in cyl - place at the end (3 cyls);

All allocated out of perm.

Syscor = real memory specification

sysmon - monitor spec:

designated userid to receive spool file

class

Auto start ?

Enable for classes (Perf, User, Dastap)

Start + Stop times

Record limit

Number of Buffers (3 or 4)

Syslocs - must be the last macro.

Put paging on fast devices, spool on slow. Separate on volumes & channels. Isolate TDISK from anything important.

Consider multple CMS system & Y disks.

HPO DMKSYS Differances:

Directory has PMA option (to follow V=R option).

DMKRIO - HPØ supports 32 channels ØØ-1F.

DMKSYS -

SYSPAG replaces SYSORD. Must go after SYSOWN & before SYSRES.

Swapping must be specified first.

SYSCOR - allows up to 64 meg.

Can have multiple syspage macros - But will round-robin through the first set first!

DMKRIO - Describes I/O farm. VM will do a TIO to each device.

For 308x processors an IOCP is also required.

All macros must be defined in the proper sequence.

Macros:

Cluster - defines control units for remote 3270s.

Can be a 3705, 3274

Terminal - define 3278s & 3279s before 3276s, 3277s & 3275s. This is required to use the new features of the new terminals.

Tony Chestang - friend of Phil, Mike, & Jeff

RCTCUNIT
RIOGEN macro must come last.

VMFASM is used to assemble these decks.

VMFASM filename ctlfile (options

Use the same ctlfile for RIO, SNT, & SYS.
Options -

DMKSNT -
To identify saved systems. Only has 3 macros. Defines what system & where on disk it is saved.
SAVESYS command provides name.
SNT must reside on a CP owned volume, usually sysres.

— skip to Next page —

CMS Nucleus Generation

CMS will go on the 290 disk in lab. CMS lives on a minidisk.
CP is simpler than CMS.

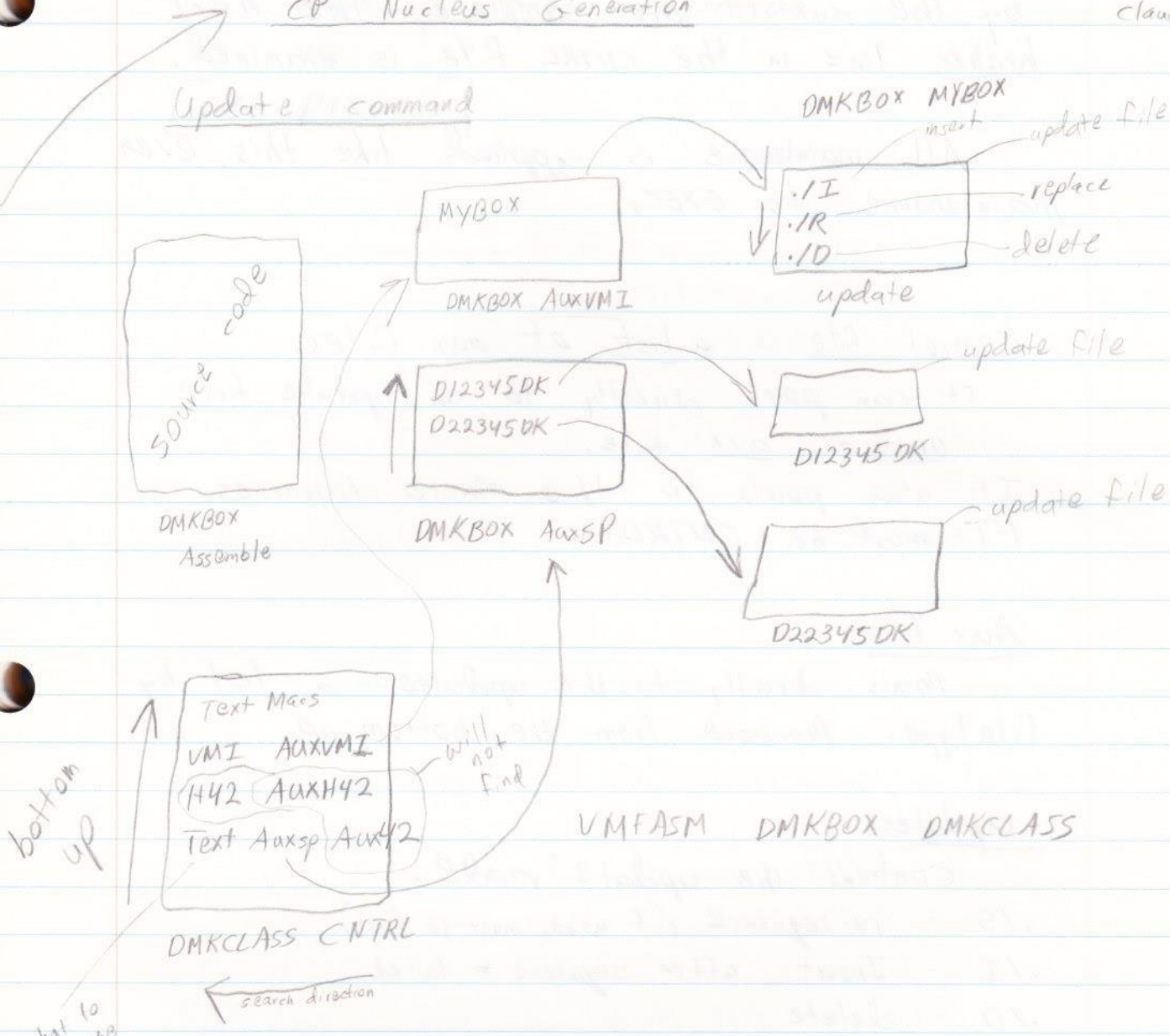
CMS on minidisk has:
1. system files
2. CMS NUCs

Format 290 B
Format 290 B 1 (Recomp
First cyl is CMS file system & rest is for NUCs.
Q V 290 will show 5 cyls; Q Disk shows 1 cyl!

11-19-86

claudia

CP Nucleus Generation



CNTRL file controls what updates go on the file. If an aux file is found, it is read. Each entry will point to a a file of IEBUPDTE-like commands which will insert, replace, & delete code in the Assemble file. Each update listed in the Aux file is read. Aux file is read bottom to top.

Each line in the CNTRL file is read right to left. Once any aux file listed on a line is found, the other entries on that line are ignored. When the updates provided

11:30

by the auxfile are complete, the next higher line in the CNTRL file is examined.

All maintenace is applied like this, even maintenance to EXECs.

Control file is a list of aux files.
It can point directly to an update file or an aux file.
It also points to the macro libraries.
FT must be CNTRL.

Aux file
Points directly to the updates. - a list by filetype. Processed from the bottom up.

Updates
Contains the update code.
./S - re-sequence (if used, must go first)
./I - Insert after sequence # listed
./D - delete
./R - replace
./* - comment

CMS Nucleus Generation

[see previous]

11/20/86
Claudia

Maintenance

PUT tape format:

File 1 - VMSERV
2 - Memo
3 - List of pgm prod maintenance on tape
The next file will have an EXEC to load down the maintenance for each product

CP Aux
CP update
CP Aux for macros
Updates for macros
Source
CP Text

Do a VMFPIC2 SCAN of tape file to see what's there.

All PUTs are cumulative from the base for a particular release (except for level set - which basically causes a new base tape to be released).

Order the PUT Bucket when ready to apply PUT. It may be wise to skip a PUT level occasionally.

To remove a PE comment out the Aux file entry & then re-run. (Check regen section of documentation).

L * fixname · to see which modules are effected by the bad fix. Copy over the IBM aux file & the comment the entry. Place the newly named aux file ahead of the IBM aux file in the search order.

Local Mods

Copy the IBM aux file + give it a new name:
AUXxxx

CMS

EREP

Classes C, E, & F are necessary to access the error cylinders. Only class F can clear the error cylinders.

IPCS

Used to move dump spool files to disk for diagnostics. IPCS/E comes with SP4.

If you add maintenance to IPCS then you must use UMFASM. Otherwise, treat it like a CMS command:

Load pgm
GENMOD module
Copy module to CMS system disk

DUMPSCAN n - to look at dump.
IPCS needs the NUCMAP available to find entry points.
PROB - create/update a problem report manually in the absence of a dump.
APAR - prints a form describing a problem.

Amdahl Analyze is free to Amdahl customers. Its has better control block formatting features than IPCS.

CMSBATCH

Mdisk 195 is A disk. Some shops will create to CMSBATCHs - one w/ 30 cyls & one w/ 150 cyls. 195 disk is erased between every job!

Control cards:

/JOB userid account ← no validity checking!

/SET time - seconds #lines #cards - limits

/* - end of job

and any CP & CMS command.

CMSBATCH should be class G.

Clear out anybody else's LINKs prior to starting your job.

RSCS

RSCS Direct - defines network.

Installation:

1. tape to disk 195
2. Xedit RSCS DIRECT
3. " Profile
4. IPL 195

RSCS Direct - do not put additional info in this file. Connections must exist.

RSCS is BC mode:

chan. Ø - Byte

1 - 5 - selectors

Can only have six channels.

RSCS maintenance files are kept on Maint's 294, 394, & 194

11-21-86

claudia

Backup / Restore

DDR may be used to preserve the CP owned areas on disk volumes. The DDR virtual machine must have MDISK statements for the entire pack. Remember to develop standardized restore procedures.

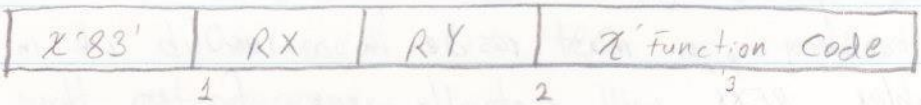
DDR may run under VM or standalone. Test the standalone before you need it. DDR exists in two versions, standalone + subsystem - thus the need to test.

For user areas, dump the whole volume via a full pack minidisk. To restore a minidisk, specify restore with only the cylinder addresses of the effected user. There are several products available to assist with VM backups.

Diagnose SVC 202

Opcode = X'83'

DMKHVC is a table of diagnose functions - a branch table. A user-written function would go into this table. 4 bytes:



0000 - 00FC Function codes are IBM pre-defined.
0100 - 01FC User definable function codes.

The ability to execute a given function depends on the command authorization of the virtual machine.

Use of diagnose is documented in the System Programmers Guide.

SVC 202

Used to call CMS modules. Can issue a CMS command from within a program.

Programmable Operator Facility

Jim Turner

First fully implemented in SP3. The purpose is to compensate for the limited console support in VM. The heart of POF is a routing table which can intercept messages and let a specific program handle the function - which can be written in assembler or REXX. Messages may also be ignored - like logons and logoffs. IBM supplies several default options. The user can write pgms to, for example, att a tape drive to a user. This is a function which otherwise would be handled by an operator.

The messages may also be routed by POF pgms to another VM via RSCS. Thus, a remote site may be unattended. POF uses IUCV. The handling pgm must reside in a loadlib if in assembler. REXX will actually run <u>faster</u> than an assembler pgm.

Monitor

Sysmon Macro in OMKSYS. Requires a gen to change. MONITOR command can over-ride defaults.

To enable RESP requires a mod + re-gen. It comes disabled.

Performance

Use INDicate and QUERY. The results of these commands vary with the authorization class of the issuing virtual machine.

SET QDROP OFF Q1 - forces user to stay in Q1. Use for VTAM virtual machine.

For CMS, never use a blksize greater than 12000. Otherwise, CMS splits it + retrieves two blocks.

With SP, keep page + spool close to the center of the volume. Put 3/4 on one side and 1/4 on the other. Then, when half fail, it is balanced. HPO is smarter and this is not necessary.

Get CMS nuc off of sysres. Put 19D + 19E on separate channels.

In an application development environment, separate the 191 disks of group members since they may tend to all be busy at the same time.

Canned Procedures

Many EXECs exist to aid in the installation of VM. User must access the right disks at the right places & then an EXEC will format, download, update, assemble, ask a few questions, punch the deck, & IPL.

MDF

12-17-87
Gloria

4K - storage can be dynamically shifted.
↳ Also, we can have dynamic partition & join.
↳ requires Parmlib change to prevent long-term page fixing. May be useful for raiding a test domain for its storage during times of need in the production domain.

CPU Allocation

Domain	# CP * Target %	= % allocated
1		=
2		=
3		=
4		=
		sum < 100 * #physical CPUs

Make a chart to schedule multiple proc / mult domain

	CPU 0	CPU 2
Dom 1		
2		
3		
4		
Total	100%	100%

← this must be 100% (or less) for each CPU.

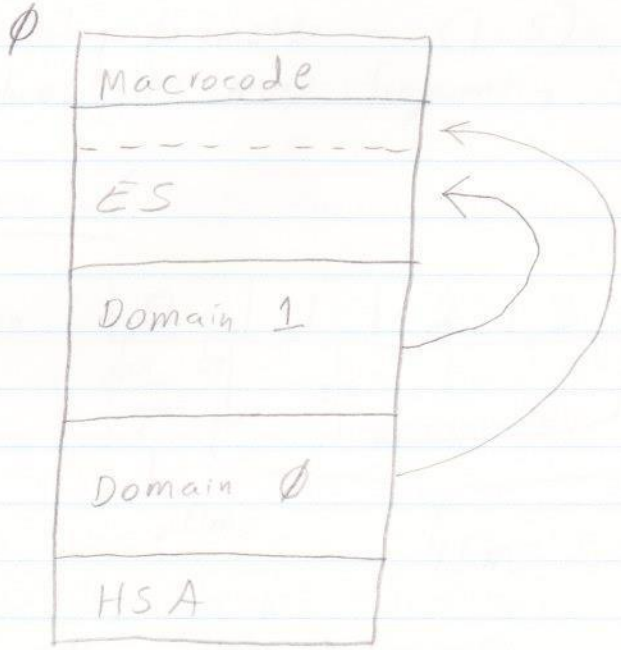
The scheduling parameter on 55 is global & parmanent (written to ISYS).

Have all pages face the same way.
Get rid of Asyac msg on fail examples.
Sheet of equations.
Format of MRSD data - DSECT

Steve guarantees that a 5870 console can run on a block chan.

A one CP system without CPU affinity will still come down in event of a MCH check.

Storage



Ø came up first.

There is no AMDIOCP release 6.ØC! This must be EREP.

AMDIOCP 6.ØC now supports classic 580s.

5890 standalone now supports 3480s.

A user determined CPID (on CX frame) has replaced the old CP#

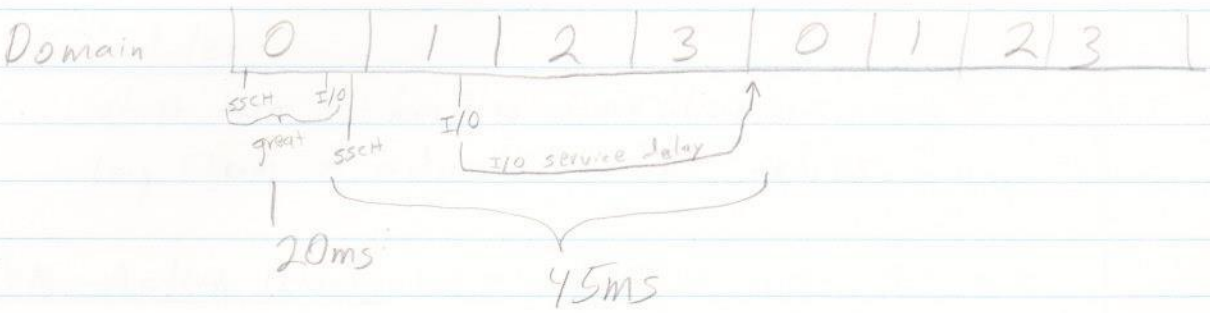
Performance

CPU overhead has 2 components:

1. Domain switching.
2. Buffer pollution.

Scheduling parameter - (1-5) - the higher the value, the more frequently it is dispatched.

Low SP:



High SP will reduce I/O service delay:

0	1	2	3	0	1	2	3

Domain comes active more frequently & picks up these queued I/O rupts.

ROT → use higher SP for interactive domains.

Due to I/O service delay, MDF may nullify the value of EDAS or cache.

Shared DASD may become dominated by a non-MDF system. Also, reserve/release becomes elongated.

A logically partitioned system has a 500ms time slice (this is the same as for a non-MDF Amdahl machine).

Scheduler

short term - handles timeslices.
long term - adjusts target values.

VM Active Wait

PSW	TIE kkkkk	CMWP
	011 3	1000
	↑ bit 5	

If one domain is a CPU hog (batch...), protect the interactive domain by giving it a minimum. Else, time will migrate to the hog.

ROT: for beginners -
SP = 3
target = max
Change gradually.

MDF Measurement

clocks:

TOD - wall clock

Clock Comparator - causes Int when < TOD

CPU Timer - decremented only when CPU is in operating or load state.

Interval Timer (370 mode) - used by TPF (ACP)

When a domain is not dispatched, the effect is the same as if the stop button had been pressed.

TMB219 - Customer version of explaining RMF wait reports.

Wait time is accurate. The sums of waits for all domains is the complex wait. What we don't know is how to attribute the busy among domains. This works even with floating percentages. 100 - complex wait = complex busy.

With fixed % we can get domain busy:

Fixed Domain % - % domain wait = % domain busy of complex

1. How busy is entire 580?

$$580\ \text{wait}\ \% = \frac{\text{sum of measured waits}}{\text{interval} * \#\ \text{physical processors}} * 100$$

2. How Busy is each domain?

DORT = Domain Residency Time - time domain is dispatched on 580.

DORT = interval * % target (for static %)

Domain Wait % = [illegible]

3. What portion of the 580 is each domain using.

Total 580 Busy time =

(interval * # phy proc) - sum of waits

MOFWatch

Shows how busy the domain is relative to the complex.

MRSD can show busy relative to the domain - but not realtime. This must come in through an exit.

VM HPO CP Internals

2-29-88
Hollis Johnson

EDHPIxx - userid
SP PRT RSCS
Tag Dev pit complex2 EPSPRT1 } to print

191 - A disk
194 - CP text
19D - Help
294 - CP updates
321 - class
322 - class
394 - CP Source

CP

Manages Real Resources (Mem, I/O, CPU)
Schedules
Provides services (with a degree of device independance) DIAG

CP simulates a console:

Begin → start button
CP Read — stop
Ext — Int
Ind — Meter
sys clear — reset
IPL — Load
Adstop — address
Trace → single step

CP provides 16 meg. of storage. A V=R user can get more.

User storage is furnished via segment & page tables of 370 architecture. Guest virtual storage is supported via shadow tables.

Seg + Page tables are in high free. DAT utilizes these tables, they are a part of the hardware architecture.

Swap Tables are not part of the arch; these point to pages in aux storage.

Virtual Machine

BC mode - 360 mode / Virtual Storage
Uses Seg + Page tables. All storage is managed by CP

EC mode - 370 mode: the guest has seg + page table. Thus CP uses shadow tables.

BC mode addressing:
VMBLOK → seg table → Pag table → Swp table
↳ CR1 (STOR)

EC mode addressing: shadow tables are required. CP can handle up to 12 shadow tables at one time.

Substantially more overhead to run an EC virt machine.

Devices:

RDR PRT PUN - totally simulated

DASD - can be real; are shared

MDISK statement

other - must be attached as real; tape, for example cannot be shared. If a real prt is ATTached, it cannot be shared.

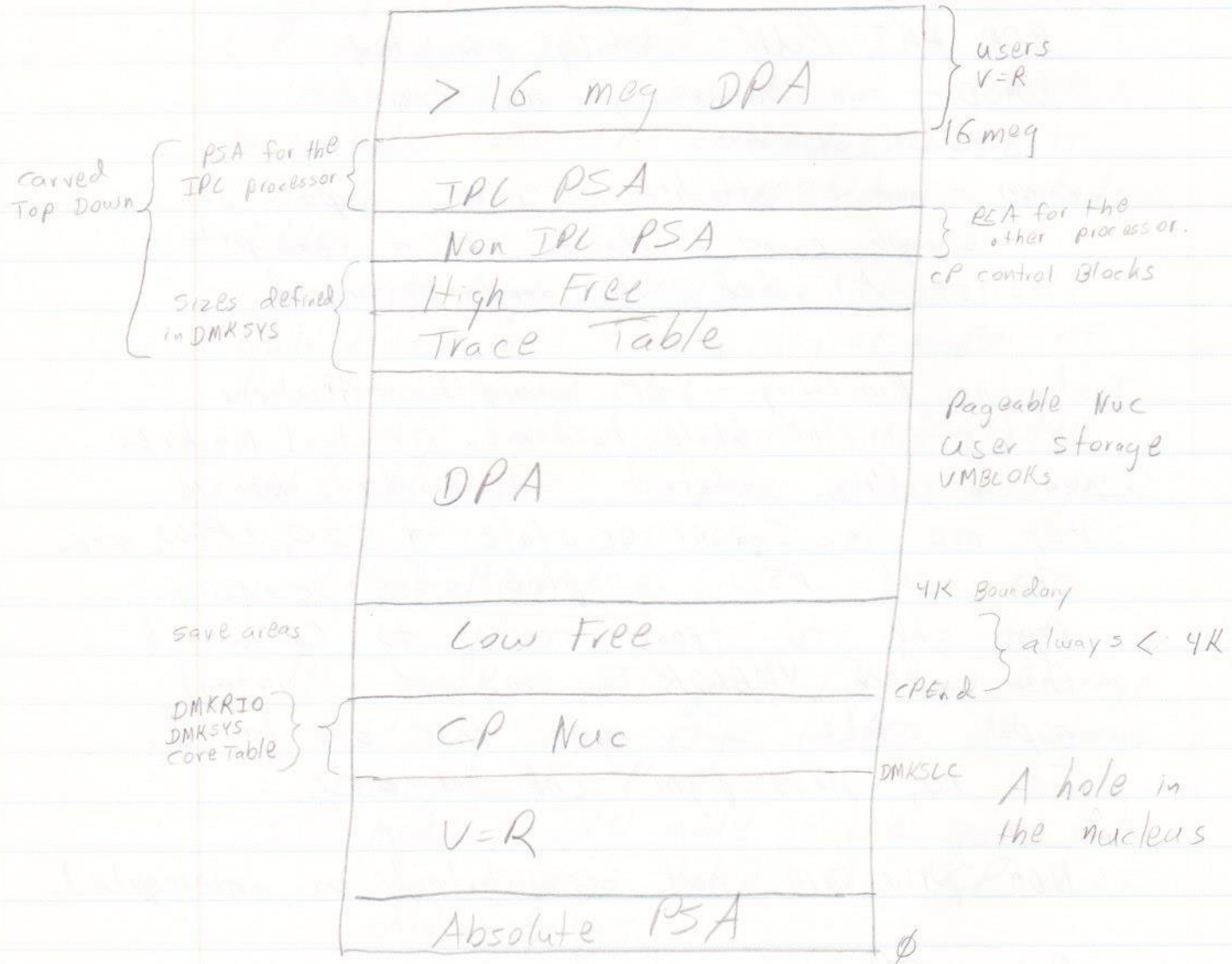
Instruction Processing - CP intercepts instructions which effect the whole hardware. Virtual Machines run in Prob. state! The machines believe they are in Supervisor state → SIO, CPSW, etc. The real PSW is prob. state, causing Pgm chk to trans control to CP which then checks VMBLOK to see if in virtual super state. If yes, then simulated. If no, pass Pgm chk to user.

Non-priv are not simulated or intercepted.

Only CP can run in supervisor state.

Timeslice is protected; user can never really disable for Ext interrupts.

Storage



If high free fills, CP will take a page from DPA (page Extends).

If the >16 meg area is used by V=R machine, then all storage between V=R area + upper area appears as <u>offline</u> to the guest.

Insure core table is <u>just</u> the right size.

CP names:

DMKxxxyy

↳ use of module — RIO — yy = CH, CU, or DV
SYS
SCH
DSP
VAT - builds shadow tables
SCN - scan; generalized search
yy = RU or VU

yy = entry point

Regs:

0 - length passing (Free/Fret, QCN, etc.)
1 - Free area adr; buffer pointer
2 - Cntl parameter; PTR, QCN
I/O { 6 - real/virt, chan pointer
I/O { 7 - " " CU "
I/O { 8 - " " Dev "
10 - stacktable entity - adds to Q
11 - VMBlok of cur user
12 - first base reg
13 - save area pointer (18 fullwords, Pointer is to last 16)
14 - return link pointer
15 - Goto pointer

CP SVCs: used by CP, not users.

0 - abend
4 - reserved (abend)
8 - link request } builds savearea & branches
C - return }
10 - release savearea
14 - obtain savearea
18 - AP/MP; SIGP

CP Linkage:

BALR - uses BALRSAVE in PSA (only one)

GOTO macro - go to dispatcher
implied exit; uses SVC 8

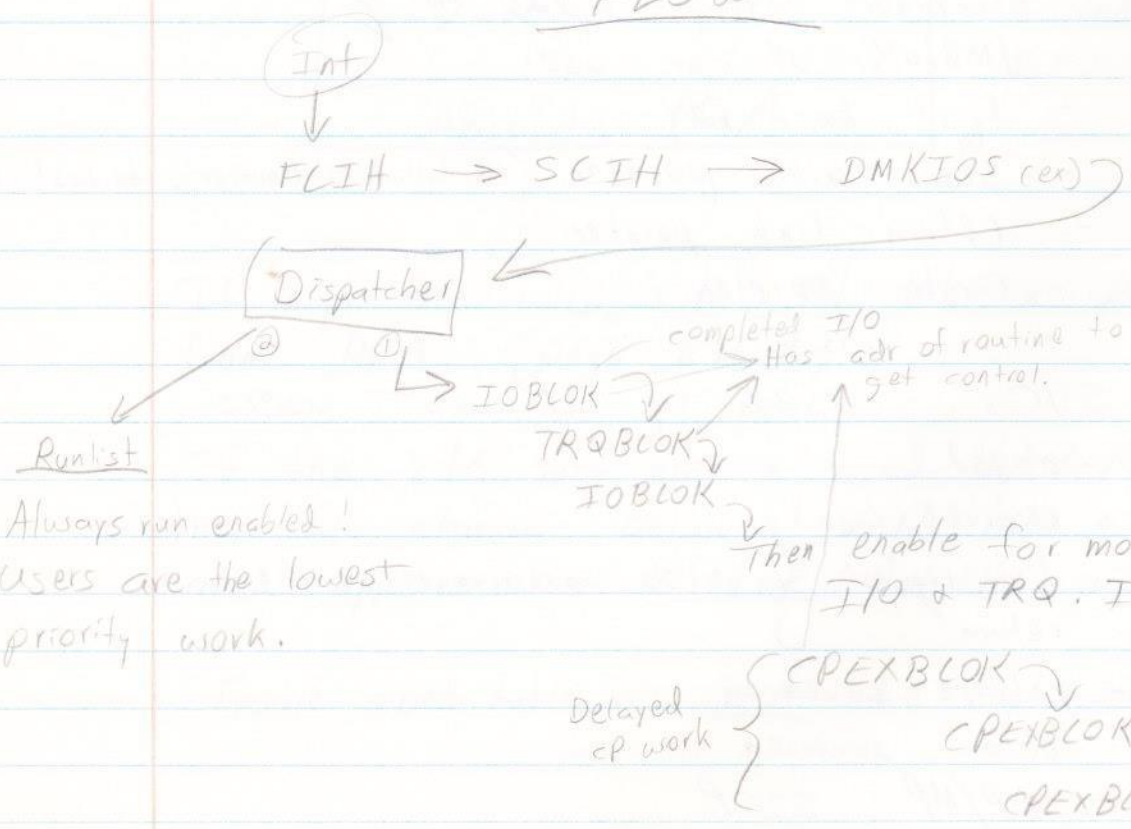
SVC Linkage - DMKSVC gets savearea

The CALL macro decides whether to BALR or SVC. DMKFree uses Freesave as a private holding area. Even routines BALRed to, can get free. CALL macro has a list of routines which can accept BALR. Most transfers are via SVC.

BALR modules must be CP resident.

All interrupts pass control to CP code.

FLOW



Architecture

EC PSW - status
next inst adr. (IS)

Bit.	Status
15	All users are in prob. state
12	EC/BC - CMS runs BC mode in virtual PSW. SETECMODE ON does not force EC mode unless guest loads an EC PSW. This allows an ECBLOK to be created. In CMS, permits use of chans over 5.
5	Trans/Non Trans - Trans means IS is a virtual adr. - used by all users. Non-tran - CP; Real adr
6+7	EN/DIS - I/O, Ext en for virt. machines; Disa for CP
	Key - 4 bits

Pgm Masks - Set in real PSW by user.
20 - Fixed point overflow
21 - Decimal
22 - Exponent underflow
23 - significant

070D - typical for user
0002 - Disa wait

BC PSW - bit 12 = Ø

No DAT bit

Channel masks are in PSW

Interrupts

Allow CPU to change its state via PSW swap.

Type code	Six kinds:	Old PSW
	Ext -	x'18'
	I/O -	x'38'
	Mch -	x'30'
	Pgm - cannot be masked	x'28'
	SVC - " " "	x'20'
	Restart -	x'8'

370 I/O

Build Chan Pgm ⟶ Seek

CAW points to CCWs — set sector

Put dev adr in Reg — Search ID (check keys w/in CKD)

SIO w/ Reg of Dev — TIC

Read adr

```
LA   R2, CCWs
ST   R2, CAW
L    R1Ø, Devaddr
SIO  Ø(R1Ø)
```

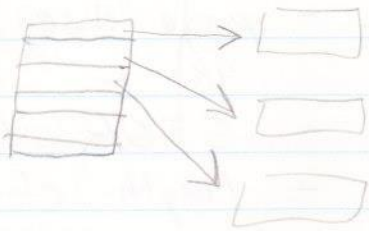
I/O Int.

CSW points to CCW after failing CCW

IDAW -

ccw addr area → table of addr → stor.

ccw addr. →



HPO uses SIOF.

Timing - 4 clocks:

TOD - beginning May 11, 1971
Doubleword
Bit 31 increments apx 1 sec.
Bit 51 is microsecond.
Runs always

Clock Comparator -
Doubleword
When = to TOD, ext int. 1004

CPU Timer
Doubleword
Decrements
Causes Ext Int 1005 when negative
Does not run in stopped state
Used when setting timeslice

Interval
Full word
Located in Mainstore
Old 360 Timer!
Bit 23 decrement 300/sec
Decrements
Causes Ext Int 0080 when negative
Does not run in stopped state
Used to set Dispatcher Timeslice

DAT - 5 parts:

PSW - EC mode bit (12) on
- Translation bit (5) on

CRØ - page + seg size
4K + 64K

CR1 - STOR for active user (VMSEG)
- length of seg table

Segment Table - 4 byte entry for each 64 K.

Page Table - available
- real page frame address

CRØ - bits 8-12 = 10000
to indicate 4K Pages + 64K segments

CR1 - bits 8-31: seg table origin.

bits Ø-8: add 1 + multiply by 64

	size of table	Document this much storage
Ø	= 64 bytes =	1 MB
3	= 256 =	4 MB
F	= 1024 =	16 MB

Seg table Entry 4 bytes

When calculating page table addr, consider bits 29-31 to be Ø.

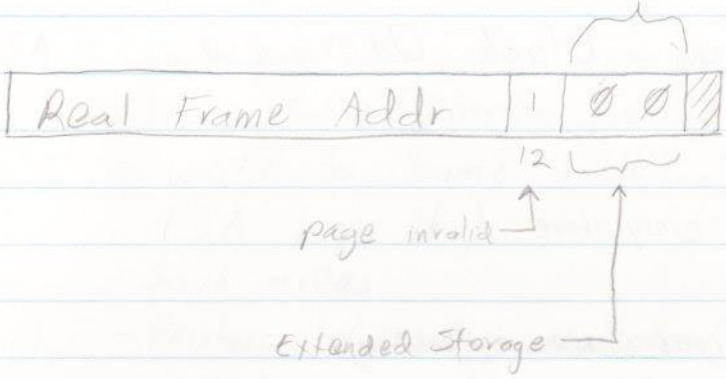
Page Table Entry - 2 bytes

Maps 4K. Hold Real Frame Address.

Seg # is first 2 nibbles

3rd digit is page

01 = 1
10 = 2
11 = 3



virt addr = 555764
go x'55' into segment table & find
FØ764762 → PT
→ Get 5th entry in PT = 444 page number
Displacement into page is 764, therefore,
real addr = 444764

If Extended Storage bits are on, move to front of address. If 11_2, then,
real addr = 3444764

Highest addr is 3FFFFFF = 64 meg.
This is the 64 meg addressing limit.

If AP or MP, prefix Reg specifies page which holds PSA. Format = XX FFF xxx (FFF = page #)
After real address is obtained, it is compared to prefix reg. If equal, real adr is changed to ØØØ. If real adr page = Ø, then change it to prefix reg page #.

Control Block Overview

PSA
- Anchors everywhere
- counters
- save areas (BALRSave, FreeSave)
- CP info
- CP status (AP, MP, UP gen & running)

→ Coretable
- 1 entry per page frame.
  who owns, etc.

→ PSA Extension (1 per processor)
  Dispatch List → IOBLOKs, TRQBLOKs, CPEXBLOKs
  Run List → VMBLOKs (started to run & want to continue running)
  True Run List → Immediately ready to run. Not waiting on I/O, Page ...

→ Real I/O Blocks
  Built from DMKRIO; CH, CU, Dev
  → Device Dependent Blocks

→ VM Bloks - chain of all user VMBLOKs
  → Segment, Page, & Swap tables
  → Virt Arch Blocks
  → Virt I/O Blocks
  → Virt Spooling Blocks

→ Real Spooling Block

PSA - hold pool of useful constants → 0, 1, -1
- trace table pointer (since both CPUs write to same table, the absolute PSA keeps this).
- Save areas
- Pointers to resident routines (for dang BALRs)

PXA - came with 3.4
1 / CPU
contains Dispatcher Queues.
Subpool info -
CP storage pools of various sizes

Real Machine
Cortable
SCBLOK - Swap C.B. Built when users are logically swapped. Anchors pages that will be swaped when user is physically out
SSBLOK - Swap Set Blok - where the swap set went to aux.
PCHBLOK
RCUBLOK
RDEVBLOK
} I/O

DASD
DASDALLOC - cyl 0, rec 0, hd 4
Reads allocation map of volume
AREABLOK - defines use of each rec on DASD. 3350 - 120 pages/cyl
3380 - 150 pages/cyl
Is page in use ...
syslist - Syspage macro info. specifies hierarchy: swapset + preferred paging

SFBLOK - Spool File Block
4.2 + below - high free (9900 limit)
5 - separate address space.

Real I/O Blocks
Built contiguously in memory.
PSA - ARIOCT - chan index table
32 halfwords containing index into chan table.
FFFF = undefined
ARIOCH - start of chan table.
Add index to this to get entry of particular chan.

RCHBLOK -
RCHCUTBL - cu table.
32 entries / # CUs on Chan.
Get index from ARIOCU in PSA.

RCUBLOK
RCUDVTBL - Dev table
can specify two chans (RCUCHA, +B) - backward pointers.
RDEVBLOK -
can specify two control units (RDEVCUA+B), backward pointers.

IOBLOK - chained off RDEVBLOK (@ RDEVFIOB. They may be chained off the highest busy component.

Locate I/O Block:

ARIOCT - chan index table ↑
ARIOCH - start of RCHBLOKS
CU - RCUBloks
DV - RDEV

Chan index table - halfwords index
ARIOCH + index = RCHBLOK

CU index table - 32 halfwords index
ARIOcu + index = RCUBLOK

DEV index table: 16 halfwords index
index compressed
ARIODV + (index * compression) = RDEVBLOK

Compression
4 bit shift

Device Dependant
UR - RSPLCTL
RSPXBLOK

Term - CONTASK

Transmission CUs - NICBLOK (Network interface - defines 3705s)
BSCBLOK - bisync devices

VMBLOK - one per logged on user.
- alloc in DPA on 128 byte boundary
- chained in cyclic chain
(QN starts with your VM BLOK)
- Virt Machine Info
- runnable
- anchors
- virt regs
- machine stats
- virt PSW

Virtual PSW - save old PSW into virtual page Ø + interrupt code. New psw is placed in VMPSW in VMBLOK to prep for next dispatch of user.

Storage

Segtable
Pagtable
SWPTable - paging
SCBLOK } swapping
SSBLOK }

Arch

MICBLOK - what microcode enabled for user
ECBLOK -
STOBLOK - multiple STORs indicator
XINTBLOK - ext int block for saving ext int for user; will be presented when user enables

EL
LEDHPIØx LEARDAY

Susan & BP - team 4

Virt I/O

Same arrangement as real, but anchored in VMBLOK.

Scheduling & Dispatching

1. IOBLOKs + TRQBLOKs, intermixed
2. CPEXBLOK
3. VMBLOKs - backward/forward chained.
   - Run List
   - True Run List

IOBLOK

addr
forward/backward pointer
CSW (to give to user)
IOBIRA - routine to get control.
Built at I/O request time & posted to a dev. At I/O interrupt, CP places it on a dispatcher queue.

TRQBLOK

- bit Ø is 1 to distinguish it from an IOBLOK. This is why May 11, 1971 is the oldest date!
- CP & user TRQs are on same queue.
- Address of routine to get control - TRQBIRA
- Pointer to VMBLOK of requestor

CPEXBLOK

- delayed work
- contains Regs
- forward/Backward pointer
- address to get control
- can be chained from IOBLOK. When IOBLOK gets dispatched, it adds this to the queue.

Real Storage Management

3-1-88
Jim Turner

STOR must be on 1 meg boundary. Therefore if DEFSTOR is 512K, VMSEG will still be 1 meg, but VMSTOR will describe exactly how much can be used.

Page - 4096 bytes of data
Frame - in mem (last 12 bits are Ø)
Slot - on aux storage

True Working Set - pages ref while in queue. These will be logically swapped out.
Trim Pages - all pages not referenced while in.
Swap Set - group of ref. page - True Working Set subdivision. ~~Trim pages will be added to achieve the~~ Excess over multiple SS size are placed on trim.
Logical Swap - still in store, on swap q.

Phy Swap - swap sets moved out to aux.

Swap area - SW on syspag macro. System will swap even if no sw defined
Interactive Buffer - Portion of DPA reserved for Q1 users. This is really an algorithm that effects elligible list candidacy. This # pages is subtracted from available pages when CP decides whether to run a non-Q1 user - more likely to go to E list. IB is set too high by default - 5% SET SRM IBUF to 10%.

Q2, Q3, & slow Q1 users (SET SRM IBUFF BUFTIME xx) appear to be entering a smaller DPA than busy Q1 users.

DPA is what's left after NUC, V=R, & low free are defined bottom up, an PSAs, High Free, Prime Free, & Trace are define 16 meg line downward. In V=R paging is controlled by the guest. PMA & VMPA permit MSS/SP to use a V=R area above 16 meg. Also, CP does not have to do CCW translation of the CCW addresses (V=R page zero addresses must still be translated).

Size of CP nuc may be adjusted with some limits. The Nuc resident coretable is rather large and cannot be moved out. Nuc ends at DMKCPEND. Above this, to the next page boundary, is low free. In SP, this is an SVC save area. In HPO it is treated as high free. SP benefits by having non-paging low free. Some sites artificially increase DMKCPEND to push it above the page boundary.

CP runs DAT off & therefore cannot use the 2 bits in the page table for addressing above 16 meg. Only the DAT on users can use this area.

High Free is defined by SYSCOR macro, FREE operand. Default is 3 page frames for 1st 255, 1 page for each 64K to 16 meg, and 1 page for each 256K above 16 meg. (Excludes V=R area)

If AP/MP, then add 25%. The default is usually too low. Try 2 page frames per logged on user. If high free is short, it will EXTEND into DPA. This is bad. Tune for virtually no extends.
Acct records live in high free until enough accumulate (80 bytes each) to cut a spool file.

In HPO, prime free storage is used for IOBLOKs, TRQ, SVC save areas. Default is 10%; should be 2%. This holds CNTL blks that are performance oriented. Check "Prime Misses" + reduce until some appear, then raise some.

Trace is usually too high! 4K is minimum + may be fine. Only SMART cares. Excess here goes to DPA. Trace cannot be change on SP w/o a mod.

VMBLOKs live in High Free in SP, and are locked in DPA in HPO. These are placed on a 16 byte boundary.

Storage Protection

The 2K key instructions (SSK, ISK, + RRB) can be used on post-3081 machines if CR0 bit 7 is set. CP checks this before dispatching. If not set, then Pgm chk.
DMKSTA checks at initialization to see if the processor is 2K or 4K keys. This also checks for VMA μcode support.

PSA +3CC points to CORETABLE (ACORETBL).
16 byte entry per page frame - two formats:
if on a list (free, flush, swap) the entry
will have forward/backward pointers on the list.
Else, adr of VMBLOK (owner) + lock count
(is CP using, is I/O using, then page is fixed).
Coretable describes all storage. CP, V-R,
etc., is permanently non-pageable. Lock count
will always be > Ø.

Coretable is in DMKSYS.

In DMKSYS,
1. DMKSYSRV - real store size not including PMA.
2. DMKSYSRM - actual real storage size.
3. DMKSYSRC - real store size including PMA.
1 + 3 are set at SYSCOR Macro. This determines
size of coretable

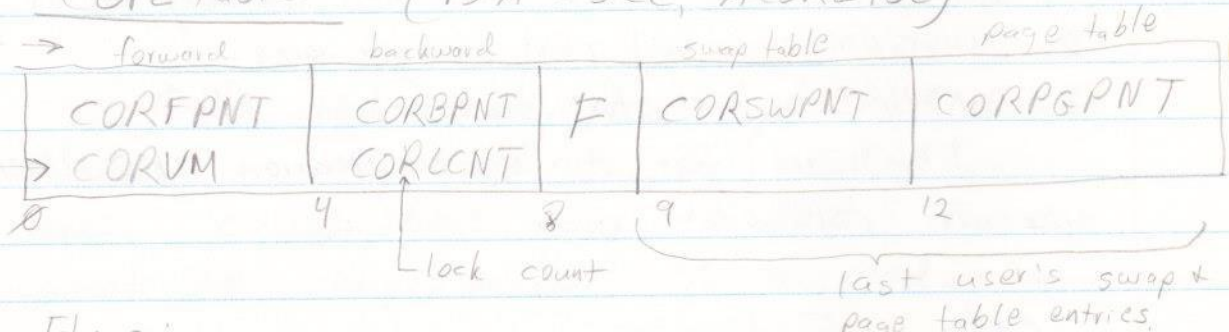
Entry: 5 types.
1. Freelist: CP wants users above the line.
- Below 16 meg - DMKPTRF1
- Above 16 meg - " 2
The user page Ø is kept below 16 meg line
because CP references this for virtual I/O.
I/O pages must also be below.
Freelist pages are immediately available.
CP maintains some pages on the freelist.
It will anticipate page outs to keep
this up.

2. Flushlist - DMKPTRU1
This is a list of trim pages.
Marked as invalid + maybe reclaimed.

3. Swaplist - logically swapped out pages.
These have been referenced.
One swap list / user. Chained
from VMBLOK → SCBLOK → page.

4. Not on a list -
User page -
System -

Core table (PSA + 3CC, ACORETBL)



Flags:
80 - locked for I/O
40 - " " console
20 - flushlist
10 - free list
08 - shared
04 - swaplist (HPO only)
02 - CP
01 - Disabled or not available

If in transit, byte 8 holds CCW opcode.
(06 read, 05 write).

Position in coretable determines real address of page frame. They are sequential.

$(RPN * 10_{16}) + ACORETBL =$ coretable entry

CP deals with Real Page Number + uses this to find the right entry.

Swap list + page tables are anchored in the user's VMBLOK. CORETABLE points to the exact entry in each. Swaptable entry holds disk location of page.

Swaptable has 16 entries:

Flag	V	key, key	DASD adr

↳ virtual page number

If coretable entry describes high Free, the first word in EDCDIC "FREE."

Or → "OL" if offline

or → "FREA" if cache aligned DPA (VMBLOKs)

One 16 byte entry for each 4K. This can take a lot of space.

Lists: DMKPTR points to 1st entries.

Flushlist - trimset goes here (not referenced during Last run. May be reclaimed, or used to replenish the freelist).

DMKPTSRS - HPO - only CP module that runs DAT on to move pages above 16 meg.

Freelist - when user logs off, or IPL clear, then pages go immediately here (also DIAG 1Ø). Else, pages stop @ Flushlist. These pages get here voluntarily.
Involuntary - from Flushlist, swaplist, or stolen.

Swaplist - working set. 1/user. These are now logically swapped out. This is purgatory.

Q-Drop
At end of timeslice the user looses working set. Runs through pagetable (to limit of VMSTOR) & invalidates (bit 12) every entry. CP picks up the real frame number & finds the coretable entry for each entry & places each page on a list. They go to either working set or trim set & then subsequently are placed on the flushlist or swaptable.

VMBLOK
↳ SCBLOK ⇄ SCBLOK ⇄ SCBLOK ⇄ ...
↳ Trim set
↳ Working set

SCBLOKs have counts of pages in each set.

The SET Min Working Set Size commands will be applied after winnowing. Pages are borrowed from Trim to working to fill out. This size should be a multiple of the swap set size! Because swap set size adjustment comes next & the non-multiple set would be paged, not swapped - undoing the intent.

If the user is SMALLV then it gets to keep all pages in working set. SMALLV is user who used < 1/16 Quantum. Quantum is timeslice allowed. This is indirectly based on the proc. speed. 5890 type is about 5ms/Quantum. User can't hit many pages in 1/16 Quantum.

Logical Swap - add user SCBLOK to chain of SCBLOK. DMKPTSIF + IB anchor interactive & non-interactive queues.

Page Flush - Trim set pages are moved to Flush list. Interactive user pages go to bottom; non-inter go to top. Pages are taken from the top to fill the freelist. DMKPTRU1 points to top.

Freelist - replenished by DMKSELCT in HPO. Its goal is to keep the freelist full. Page + Swap outs occur only to replenish the freelist. Freelist size = 1 page per user in runlist + 2.

CP prepages 2 swap sets (default) prior to dispatching an interactive user. Be sure not to exhaust freelist by pre-paging too many swap sets. SET SRM Minnumss & prepage are closely related.

Swap Out/Page Out

Garbage Collection - time dependent. Since CP doesn't Q drop CP pages in DPA would never get to free list. Garbage collection scans 1/8 of DPA beginning around every 4 sec looking for unchg, unref pages. Entire DPA is scanned after 8 cycles. Uses RRBE to reset ref bit. If already off, the page is placed on freelist.

Average life of garbage page is 48 secs.

HPO 5 Garbage collector is enhanced. Rel. 5 does less at Q-Drop. It will catch more user pages.

DMKPTRF1 = 1st CTE on < 16mb Freelist
2 = " " " > " " "
FN = count of # pages on < 16 mb
N2 = " " " " " > " "

DMKPTSIF → Interactive logical swap queue anchor
BF → Non " " " " "

CP tries to save I/O. It puts swap sets out before draining flushlist. Pages of users logically swapped for 100 seconds are moved out with one SIO & lots of pages are freed up. This beats 1 sio/page from flush list. Making the time shorter will increase swapping. This can be desirable.

Once old non-interactive users are gone, then drain flushlist. A VFault (virt fault) will look for a page above 16 meg. Else, page fault is for CP use & should be below 16 meg.

Then, swap out non-interactive users from logical swap users.

Then swap out interactive.

phase 1 Finally, <u>core table scan</u>! Not locked, not CP, not on a queue, not changed, then invalidate & steal.

If a page below is needed & a page is available above 16, then simply move a page above & free the frame below.

DMKPGT - find the best spot on disk.

PAG - schedule page write.

<u>Core tabe phase 2</u> - quiese other proc & take everything including reserved pages.

Replenishment Summary

1. Disposable Page Collector (Garbage Collection)
2. Old non-inter working sets
3. Flush list
4. Swap queues
5. Core Table scan

DMKPTRFN - # pages avail (Freelist) < 16 m
N2 - >16
SS - # page steals systemwide
FF - # pages moved from flushlist
PØ - # times freelist empty
PR - # pages reclaimed
CT - # calls to DMKPTR
SW - # pages being swapped < 16 meg
S2 - " " " <
ES - " " " stolen from in-z users
NP - " " " avail in DPA

Some of the counters are reset at Monitor interval.

Free Storage Management

This is management of high Free - DMKFRE & FRT.
FRE still serves as the entry point for FRT. (DMKFRET)
Types of request:

Normal -
- Big - Frels list
- Little - from subpool (of appropriate size). separate for each CPU.

Prime - set size blocks for performance oriented control blocks. Separate for each CPU.

Cache Aligned - build VMBLOKs at logon.

DMKFRE - contains code to allocate storage & manage it.
DMKFRT - logic to deallocate.

PXA - subtable → subpool table chain anchor. 2, 4, 6 doubleword chains. These are dedicated to a CPU. Alloc is fast.
DMKFRE holds FRECS list. Holds table of large free areas.
FREENUM - 8 bytes past FRECS, a count of # blocks on FRECS.
PXA - anchor for prime storage blocks.
DMKFRE - anchors cache aligned blocks which are available.
DMKFREE - entry point:

```
LA   R0, IOBSIZE     # of doublewords
CALL DMKFREE
+    L    R15, AFREE
+    BALR R14, R15
```

On return, R1 has adr of storage.

Requested size will be rounded up to match a subpool. Area returned is not cleared. Storage must be explicitely release when finished.

Small subpool request: (less than 1024 doublewords, 8K!) (SP- 33 doublewords) - Round up request, check subpool pointer in PXA. If subpool avail, move pointer to R1.

If none available, go to FRELS list. Pointer is Ø if none available.

92 Subpool sizes! (SP has 11).
2 to 128 doublewords in increments of 2 bytes.
160 to 1024 " " " " 32 "

This table has two entries for each pool size; one non-DPA followed by one DPA, etc.

DPA & non-DPA subpools are chained separately. Also, each CPU has its own set. Non-DPA subpool blocks are timestamped.

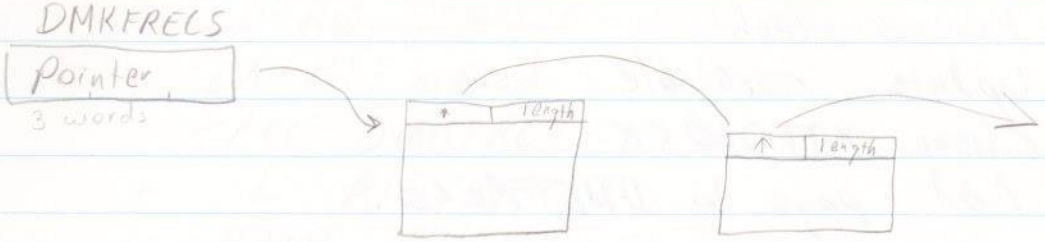
Once/hour subpool merge tries to combined blocks to place on FRELS. This also happens whenever a user logs off. If a pool block is contiguous with FRELS then combine.

Subpool Entry -
Doubleword in length:
Address of Free Queue Element
# of elements on chain

FQE adr	# FQEs	Mon. Seg #
4	6	

Large Block Request - DMKFRECS -
entries are in ascending address sequence.
1st word = pointer to next
2nd - unused
3rd word - # of blocks

Each block has a length header. in 2nd word.
First word is pointer to next block.



FRECS list is run, looking for first or last equal request.
If request is for greater than 28 doublewords, look for last equal entry. If no fit, take last larger size fit.
If no storage on FRECS, then EXTEND. Call DMKPTR to get DPA block & search again.
During Extend processing, no useful work is being done. Extend occurs one page at a time & if the request is for more than 4K, pages will be picked up until enough contiguous storage happens to appear! It could collect (page out) 60 page in order to deliver 5K for CP use.

Extend Processing

Bump Extend count.
Set XTNDLOCK (x'349' in absolute PSA) so dispatcher won't start anything new.
Decrement # avail pages
Get lock (DMKLOK)
Call DMKPTR for a page frame - if a page out is required this is a performance killer.
Release lock
Update coretable
Clear XTNDLOCK
Add page to DMKFREELS.

And think what happens if 4K is not enough.

Prime

DMKFREEP - entry point. Prime is a permanently alloc pool of 16 Doublewords. 10% Default is too high. Change to 2% of high free.

If no prime is avail, use normal free storage subpools. No great loss.

Each proc. has own prime. Once/sec the pools are balanced between processors.

Cache Aligned Pool Request - aligned for HSB access. Six pool sizes but only 96 DW is being used (for VMBLOKs). DMKFREC is entry point. Pointer is DMKFREA. If none avail, another page is taken from DPA + carved up. Like a permanent extend.

With 4.2, storage is released via DMKFRET, but entry is via FRE.

```
LA   RØ, IOBSIZE
LA   R1, areaadr
CALL DMKFRET
+    L   R15, AFRET
+    BALR R14, R15
```

Subpool blocks are pushed on top of regular subpool chain. Prime go on top of PRIMHDR anchor.

If a whole block belongs to DPA, an unextend occurs.

Unextend occurs unless the routines are DMKQCO or DMKQCN which are known abusers. These guys loop on acquire + free. DMKFRET does not unextend, it holds block assuming that another request will be along shortly.

Storage Pool Merge - Once/hr or at every logoff. DMKFRET checks every subpool to see if it is adjacent to a FRECS area. If so they are merged.

Non-DPA blocks are returned to free storage if at least 60 seconds old, (for small blocks), or 15 secs for large blocks.

DMKFRERC - I want storage but do not extend the system to get it! Time driven sampling routines use this. They simply reschedule & try later.

DMKFRETR - returned storage goes directly to FRECS. Keeps subpools from growing

DMKFRETE - logoff cleanup. Branch to FRETR. Unextend if possible. Called by DMKUSP.

Virtual Storage Management

Control Blocks:

Hardware Defined {
VMBLOK anchors segment table + SCBLOK.
segtable points to page tables.
Pagetable contains RFN of each page.
Status
Abo points to swap table.

SWPTABLE - DASD location of slots → direct pointer if paged out. Indirect pointer points to SSBLOK (swap Set Block).

SCBLOK points to swap list of SSBLOKs for logical & phy swapping.
SSBLOK - DASD location.

Demand Paging - two swap sets are prepage. The rest of the working set is built one page fault at a time. Needs a PIC 11, or 10.
CP can explicitly demand a page since it runs DAT off.

If CP ever pgm chks, it abends.

Pre-paging is initiated by the scheduler. Number of swap sets should be number of paths to swap sets for interactive users. For non-inter, prepage 1 swap set. The lowest virt store adresses are brought in. No guarantee this is what is first need.

Page fault on a page in a swap set brings in the entire swap set.

Page status:
- Resident (see Page Table; bit 12 invalid bit)
  - Active → bit 12 is Ø
  - Reclaimable → bit 12 is 1 + RPT is ≠ Ø
- In Transit → flag in swap table
- Not Resident
  - Paged out - Page table entry ØØØ8
  - Never used -
  - Part of a swapped out set

SEGTABLE - CR1

has flag byte in last byte of adr. This is ok since a page table must begin on a DW boundary (last 3 bits Ø).

X'4Ø' - seg ENQ
1Ø - seg mig
Ø4 - seg prot
Ø2 - seg common (MVS)
Ø1 - segment invalid. PIC 1Ø

Examples:

FØ1234ØØ - in queue
adr

FØ1234Ø1 - invalid bit on; user inqueue.

ØØØØØØ01 - Never used. Only invalid bit is on.

Page Table Header - 16 bytes ahead of table. Has timestamp to see how often the segment is referenced. This is used for page migration.
For shared seg, has a count of # users. Also, pointer to share table (SHRTABLE). (-8)
Has pointer to swap table. (-4)

Entries are half words. Bits Ø-11 are RFN. Bit 12 is invalid bit. Bits 13 + 14 are extended adressing. Bit 15 is used by page migrator. This means Migrator has been here if Ø8.

Swaptable - 16 byte header. Points to VMBLOK + Pagtable.

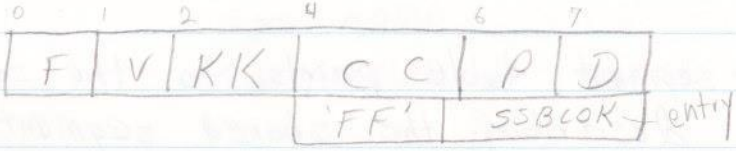
Entries:

Flags → has double ref + chg bits to support 2K keys.
In-transit bit.
Shared sys
Permanently assigned. Migrator cannot move this. The CP NUC set this for himself + leaves NUC on best paging device.

Virt page # (0-F)

Key - 2 bytes (for 2K pages!)

DW
- CC - cyl #; If FF then pointer to SSBLOK
- P - formatted 4K page slot
- D - dev code - position in sysown list. Begins with 0.



SC BLOK

Next Pointer
Previous ↑
} if on a list

Owners VMBLOK
SSBLOKs
↑ 1st SSBLOK
↑ last "
} if phy swapped

↑ CORE Table entry of first page on swap list (Log. call/swap
↑ last
↑ first trim
↑ last "
pages in WS
Trim

see VSM 11-4

SSBLOK

Flag byte for in transit
Has SSBLOK entries to link virtual page number to each page slot (since they are not necessarily contiguous in the swap set).

CP Virtual

CP does use seg & page tables - but manages it itself. Up to pageable Nuc, virt adr = real adr. Everything in CP has a constant virt stor address. It treats itself as a 16 meg virt. machine. Spool file buffer, etc. are created at init & then paged out.

3-2-88
Hollis Johnson

Shared Segment Management

Each user's segment table points to the same page tables describing the shared segment. The SHRTABLE lists the page table locations to be shared.

Shared segments are defined in DMKSNT.
Attributes:

shared / Noshared
↳ protected / unprotected

Modules

DMKCFG - Entered via logon, IPC CMS, or Diag 64. Calls DMKCFF.

DMKVMA - ensure seg protection when VMA seg prot not avail.

DMKATS - "unshares" changed pages & notifies users.

Control Blocks

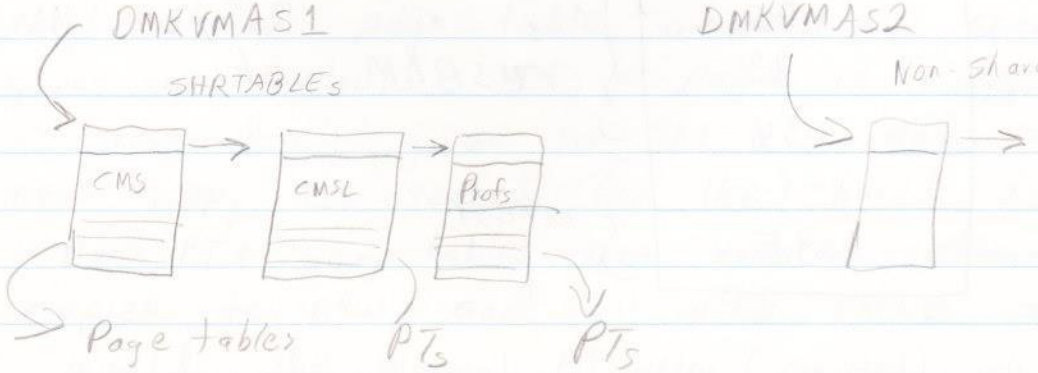
VMABLOK - 1/ saved sys / user - chained off VMBLOK (usually multiple).

SHRTABLE - 1/ shared sys or DCSS

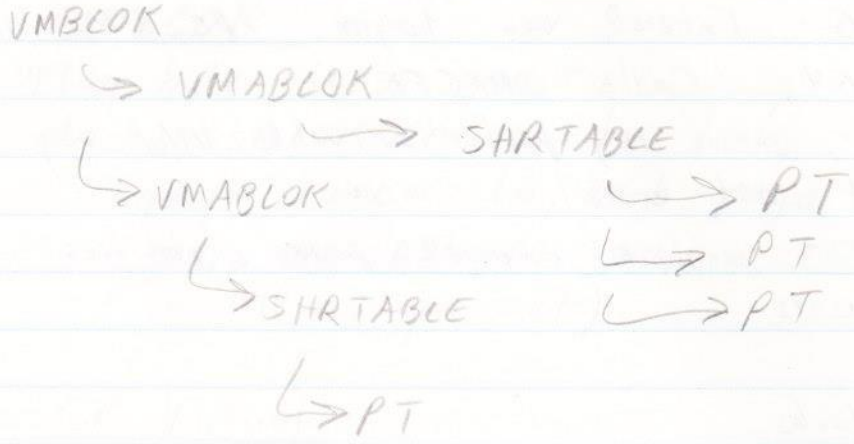
VMABLOK - 16 bytes: ↑ next VMABLOK
↑ SHRTABLE
Sys Name
↑ Non-Shared sys

SHRTABLE: Flag byte - indicates protection.
sys name
↑↓ pointers to shared & unshared tables on chain
size in DWs
shared segs
shared seg numbers
Pointer to page table for each shared seg listed above.

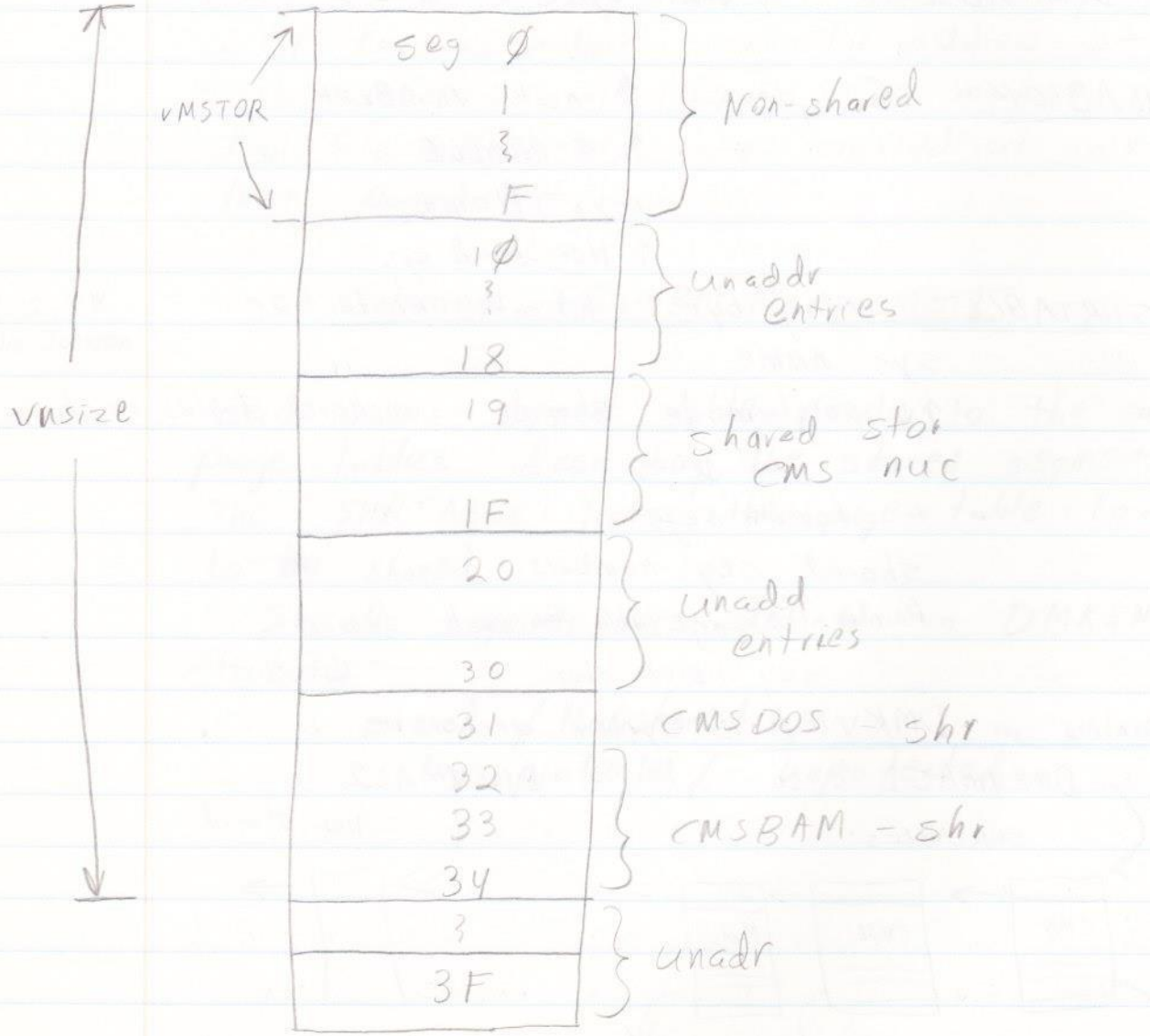
Pointers in DMKVMA: System anchors -



Virtual Anchors



Segtable



VMABLOK is not required when VMA seg prot is avail. PT is found through DMKVMA chains.

SHRTABLE is built when 1st user requests system + storage is freed when last user logs off.

DMKCFF - does setup + clean. Will build a SHRTABLE if required. Updates PSW + regs for user to begin exec.

DMKCFG - reset user on IPL cond.
call DMKVMI if IPL by adr.
If IPL by name, access DMKSNT
construct tables
DMKCFF to prep user.

DMKCFF - get PT + swap tables, updates them
- reads DMKSNT
- Determines shr / non-shr + anchors in VMABLOK
- Builds new shrtable if req.
- modify PSW + Regs
- exit to disp.

W/O VMA protection, CP must check every shared page table whenever a user gives up control! If a page is chg, it is unshared + given to the user and a new copy is created for the shared users. New PT + swap tables are created, changed entries copied to new, and all other entries are invalid. The original PT entry (corrupted) will be invalidated + the corresponding swaptable says

to read it in from DASD. Thus a new copy will be brought in whenever a user tries to reference the page. The new SWPTable entry is invalid, indicating a copy of the page does not yet exist on Aux. As the changing user runs, the other entries (now blank) in the page table will be built.

AP/MP - each CPU has a set of page & swap tables.

Hardware Seg Prot - bit 29 in segment table entry (x'04'). Causes seg prot if change attempted. PSA flag indicates if seg prot is active. VMABLOKs are not used.

Sharing reduces paging because it reduces the number of pages trying to reside in memory.

Hollis

Paging / Swapping

At Q-Drop the PT entries are invalidated & separated into Trim & working set. These are anchored in an SCBLOK. Trim is placed on flushlist (DMKPTRU1). WS is on swaplist. PTEs of both will be in the form xxx8. Pages are invalid, but still in & therefore RPN is ok. Coretable entries will not have a VMBLOK pointer; instead it will have pointers to flush or swap list, plus flags.

When user comes in-queue & references a page w/ invalid bit a PIC 11 occurs. DMKPRGIN detects PIC 11 + calls DMKPTRAN which returns CC ∅ or 2. ∅ = page is now avail. CC 2 = means virt adr was out of range & causes PIC 5, addressing exception.

To reclaim, turn off bit 12 in PTE, update coretable w/ VMBLOK, & repair chain to remove PTE.

DMKPTRAN exec LRA - load real address.

CC ∅ = Pg in storage ⇒ then exit
1 = seg. invalid ⇒
2 = PTE invalid
3 = addr exception

This test governs what PRGIN needs to do in order to make the page available.

Two types of paging: 1) Demand (PIC 11) which is either page or swap fault. 2) Prepaging. CP doesn't PIC 11. It uses the TRAN macro which causes DMKPTRAN to run.

Pages will be either on flush, swap, or free list, or else out on aux.
Free list pages don't need to be paged out.
If pages on swap list (log swap) all will be made available when user moves in-queue.

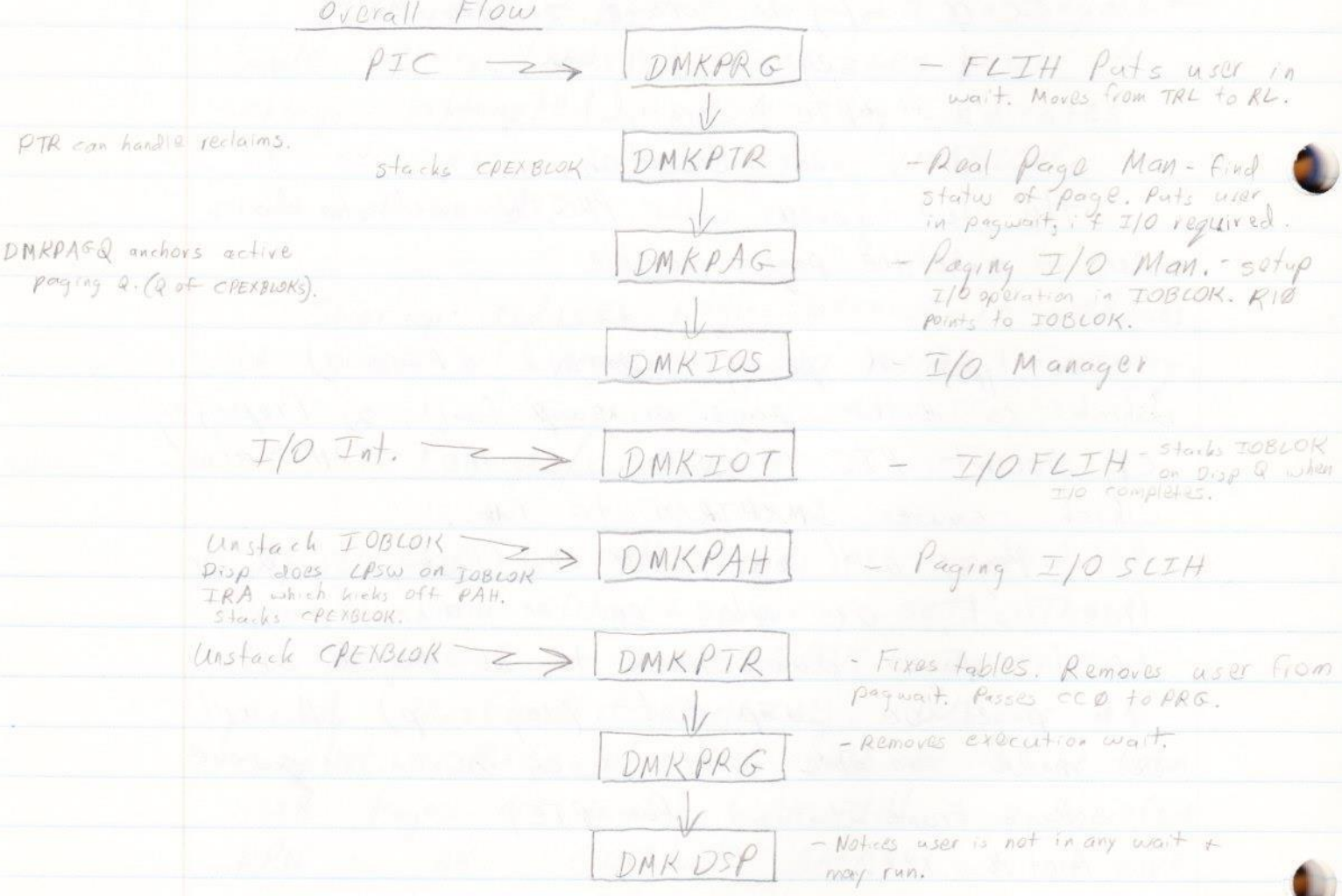
Page Frame Status: (on PTE)
Active - XXX2
Reclaimable - XXX8 (& coretable tells which list)

In Transit is a special case. coretable flag has CCW op code. PTE = XXX8
Swap table shows in transit.
If frame is paged out PTE = 0008.
If never ref. PTE = 0008 + swap table is all zeros.

CP Explicit

R2 has parameters + DMKPTRAN is called. On return, real address will be in R2.

Overall Flow



DMKPTR does a load real addr. (LRA).
This causes addr trans to occur & the real addr is returned in a reg if available & CC=Ø.
CC=2 = PTE invalid - most complicated. → Reclaimable, In Transit, Out.
1 = STE " - then go to DMKSTR (seg Tab Rebuild)
3 = Virt address is out of range.
LRA R2, Ø(R1) lets CP to do a translation without DAT being on.

DMKPRGIN - entered via PSW swap. If not a PIC 10 or 11 goto DMKPRV or FPS for simulation. Put user in Exec wait. SVC to DMKPTRAN. If user is EC mode (VMBLOK virt PSW) then goto DMKVAT for use of shadow tables.

DMKPTRAN - Bring page in. This is invoked with options (Bring, Defer, VFault usually).
R11 = VMBLOK.
Reclaimable: (CC=2 from LRA)
If page is on freelist:
increment reclaim count (DMKPTRPR + ER)
decrement # free pages (DMKPTRFN or N2)
Set storage keys
Reset ref & chg bit (leave chg on if on)
Update Coretable entry:
VMBLOK in 1st word
Lock Ct in 2nd "
Set flag
Leave PTE↑ & SWPTE↑ alone in CTE
Put real adr in R2
Return to caller w/ CC=Ø

common even if page is on flushlist

In Transit : (CC=2 from LRA)
check swaptable + see page is in transit.
Defer is not specified : (only if CP invokes TRAN + can't wait)
Build CPEXBLOK
Exit CC=1

your R3 will then get control when page is back. It is the IRA for the CPEXBLOK.

Defer specified : (CP is not demanding the page immediately) This is usual
Add pag wt ct
Place user in page wait
Build CPEX + chain to existing CPEXBLOK because this page is already on the way in.
Exit to DMKPAG

Not Resident : (CC=2 from LRA)
Add pag wait count
Place user in pagwait since I/O req.
acquire free frame from freelist
update CTE
Flag swap table in transit
If first ref, then return to caller CC=Ø
Else, page-in necessary.
Stack CPEXBLOK
Call DMKSEL

PTRAN Summary
LRA → Ø - resident → lock if requested, + exit CC=Ø
1 - Seg. Table Rebuild (STR) + try LRA again
2 - Page Table Exception
- Reclaimable
- In Transit
- Not Resident

3 - Exit CC=2

DMKPTR can also lock (DMKPTRCK) or unlock (UL) a page, or request an available page from freelist (FR). - this last emulates a first-ref page fault.

Swapping

Swap In - data is in a swap set on aux. DMKPTRAN still handles + determines if paged out or part of a swap set. In swpTable, field swpCYL will be

FF | ↑SSBLOK ('FF' in first byte is flag)

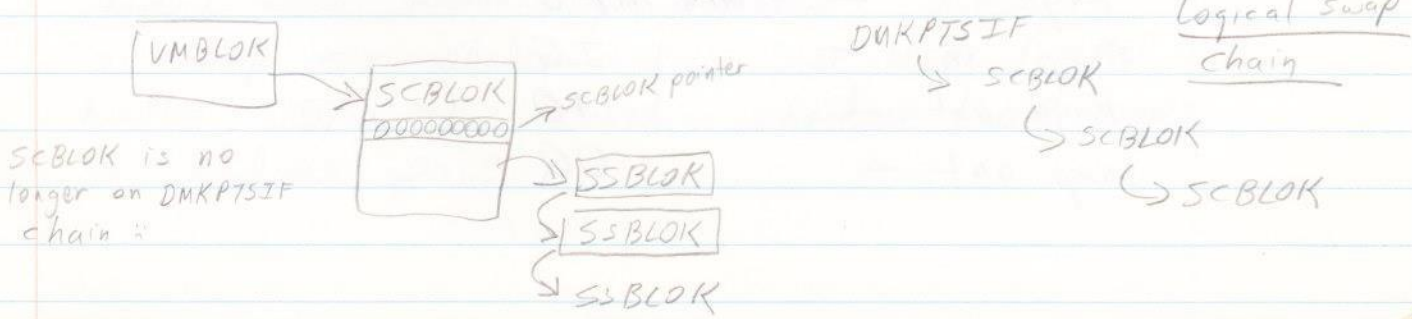
if swapped +

cccc pp dd

if paged.

Logical Swap-In - DMKSWA does pre-page or logical swap in. Check SCBLOK to see which. SCBLOK is no longer on chain when physically swapped. SCBLOK had been chained with other SCBLOK at Q-Drop. DMKPTSIF is anchor for these. Also, SCBLOK now points to SSBLOKs. ↑↓ SCBLOKs are zeroed when removed from chain.

Then remove CTE from swap list chain, put VMBLOK↑ back, reset CTE, + hardware keys.



Pre-Paging: do logical swap in of any remaining pages. Bring in # of swap sets specified using lowest addresses of sets actually referenced during last execution.

Swap Out - swap out occurs when DMKSELECT need pages for freelist:

1) Garbage
2) Old user swap
3) Flash
4) New users
5) Coretable scan

Get SCBLOK from non-inter queue. If intract user still logically swapped for DMKSWAITB (20 sec), then move him to non-inter queue.

Call DMKFREE for storage to build SSBLOK. Chain SSBLOK to SCBLOK.
Call DMKPGT to get slot on DASD.
Call DMKPGU to get rid of old slots.
Get CPEX to do swap out.
If more pages must move, build more SSBLOKs.

CPEXBLOKs are chained off:

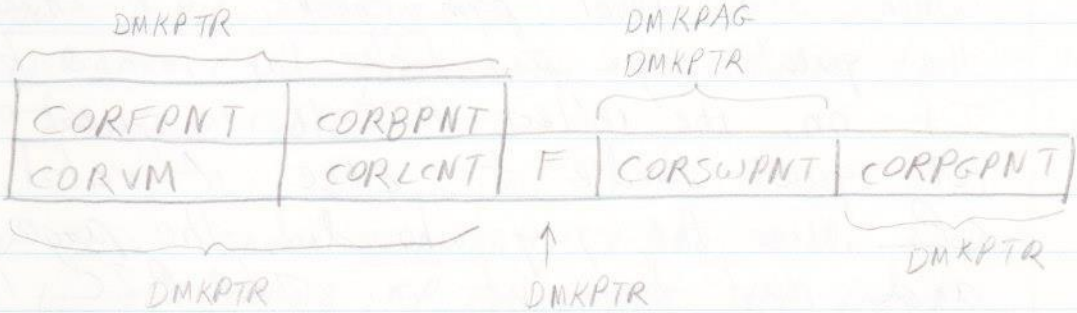
Page in - DMKPTRRQ
Swap in - IQ
Page out - WQ
Swap out - OQ

DMKSWA gets control from CPEXBLOK.
Set swap out flag in SSBLOK.
Call DMKRPIS to add page to freelist queue.

Real I/O

DMKPAG calls DMKIOS which queues IOBLOK on RDEVBLOK. All IOBLOKs are queued (& sorted) on RDEVFIOB. If multiple IOBLOKs required same cyl, then only one seek is required & will be executed together. The active IOBLOK is anchored @ RDEVAIOB.

Coretable Entry



3-3-88
Jim Turner

Shadow Tables

Shadow Tables are simply another set of segment and page tables. These are used for second-level V=V virtual storage machines. They can eliminate double paging. Shadow Tables map guest virt to CP real. The STOR gets loaded into CR1. Shadow Tables involve high overhead. CP SP can map six shadow tables per user (HPO handles 15/user). Each shadow table maps an address space.

These allow paging systems to run under CP. It is an arch simulation.

PIC 10 & 11 processing logic must determine who paged a page out, guest or CP. The right one must page it in.

Virtual PSW bits 5 & 12 indicate paging.

Guest Virt → Guest Real → CP virtual

level: 3 2 1

This ↑ is the problem.

Shadow tables map 3rd level to first level. This requires CP maintenance of the tables. When 3rd level pgm checks, CP checks the guest seg & pag tables for invalid bit. If on, the reflect PIC 11 to guest & let guest handle it. Else, if valid, then CP was the guy who did the page out and must bring it in. The PIC 11 is not reflected to the guest.

Modules: DMKVAT & DMKVAU

Control Blocks:
- VMBLOK -
  - ECBLOK - architecture stuff
  - STOBLOK -
    - Shadow segments -
    - Shadow Tables -

There are now shadow swap tables. CP doesn't care.

For non-paging guest, CP does everything. The guest never receives PIC 11. It believes memory is real.

Shadow Tables have a combination of info ('real' RFN) to map guest virtual to CP real.

Guest cannot modify segment + page tables w/o issuing privileged inst. (Load CR 1, SIOF, IPTE, (for paging) which is prohibited + CP intercepts the pgm chk + knows to check. The shadow tables are checked prior to the user being dispatched (via load CR1).

Shadow Table Creation:

Step 1: Copy guest Seg + Pag tables to shadow tables.

Step 2: Check each entry + do addr. trans to see where CP really put the page. Imagine the overhead.

Guest Virt		CP Virt Guest Real	CP Real	Shadow
020000	Guest Does This Trans	12000	455000	456000
021000		23000	N/A	N/A
022000		N/A		
023000		3400	789000	789000

CP will always build a shadow table for a EC, DAT V=V machine even if it has to give up one already built! and create a new one just to dispatch the user!

DMKVAT determines who did the page out on a PIC 11 + will go to either DMKPTR or PRG.

VMBLOK → ECBLOK (this should look like an adr).

Control status flag:

shadow tables present? valid?

Has CRØ changed (virtual CRØ means guest has changed architecture).

Is CRØ valid? (NO, pgm chk user) IF BC mode CRØ not chk.

Is user in EC mode?

ECBLOK

save areas for virt CRs.

" " " shadow control Regs Ø & 1

(These point to the tables running this address space. These will be loaded in real CRs).

↑ first STOBLOK (Default 3, SET STMODE xx)

count of STOBLOKs.

STOBLOK

pointers to next & last STOBLOK (last is in case one needs to be trashed).

Virt & Shadow CR1

size of VV=VR area (guest nuc)

last VV=VR page table.

Shadow Segment Table (not pointer).

↳ points to shadow page tables.

SET STBYPASS
tricky to use by
worth it.

Common Shadow Page Tables - various tables under a virt machine can share common page tables. - This is for the commonly addressed portions of MVS address space. The limits must be exactly correct in the CP command or MVS address spaces will corrupt MVS storage!

DMKVAT - Virt Adr Trans.

Does table maintenance -

Builds tables if bit 5 & 12 on.

Checks when CR0 or 1 is reloaded.

Any user page operation.

Guest PTLB, or IPTE instructions.

Allocates & initializes as soon as guest enters EC DAT mode.

When the user leaves EC mode, the storage is freed (ok to leave DAT mode).

VAT determines who (guest/CP) did a page out when a PIC 10 or 11 occurs. Entered from DMKPRG if bits 12 & 5 on. Call DMKPTR if CP invalidated entry. Else, give user the PIC.

DMKVAU - utility routine.

Does the translations of the guest tables to build the shadow tables. (3rd to 2nd level and 3rd to 1st).

Pseudo Page Faults - part of DOS & VS1 hand-shaking (SET PAGEX ON). Allows guest to suffer multiple page faults. If on, the user is not placed in page wait until the page arrives. The guest can run other work (VM=YES in guest gen). CP passes a PIC 14 (not in architecture!) & the guest dispatches other work. A second PIC 14 means the page in now in. PGBLOK ← ECBLOK holds the virtual address of the page to be brought in.

DASD Storage Management

CP tries to choose the best spot on disk to place a page.

Spooling is really non-preferred paging as far as management is concerned.

DMKPGT - Alloc DASD space (slot)

PGU - Dealloc " "

TDK - Alloc/Dealloc TDISK (using same control blocks)

VDG - @ init builds Alocbloks + Recblocks.

OWNDLIST - List of DW for each vol in the sysown list holding volser + pointer into RDEVBLOK. Resides in DMKSYS. Order of sysown list determines the order of volume use (CHPO over-rides with sys page)!

RDEVBLOK - describes device + if CP owned points to Alocblok. RDEVBLOK are anchored in PSA at ARIODV.

Alocblok - three types:

Full Volumes - describes CP space within pack. Used only to Dealloc slots. ↑ from RDEV.

Area Alocblok - describes use of CP area + is used to allocate. ↑ from sysplist

IDisk - describes TDISK space + use.

RECBLOK - represents a cylinder. In CCPD, CC points to RECBLOK & P points to entry therein. CP checks each RECBLOK when looking for a free slot.

CCPD
- → Alloc blok
- → entry in RECBLOK
- → RECBLOK

SYSPLIST - created from syspag macro. Defines use of each volume. Defines hierarchy for paging. Swapping → pref pag → gen pag → pag mig → Temp.
Be sure to round-robin among devices in same class.
Primary point of control is SYSPAG & SYSOWN macros. 4% of CPU cycles should be for paging. If more, then the dispatcher is effected. Take care here. IND command lists paging as a % of cycles; also SMART ← VM MAP.

ALOCBLOK -
- type flag (SW, PP, PG) may be combined for full-pack Allocblok.
- ↑ next deallocation Alocblok (PGU follows this)
- Cyl count in use / in area (how many empty cylinders. cyls in use may have only one page. A used cyl has a RECBLOK).
- # pages alloc
- # pages in RECBLOKS
- ↑ RDEV Blok

anchors for paging & spooling RECBLOK chains

↑ next allocation ALOCBLOK
cyl pointers
↑ to current RECBLOK (where the arm is).
lots of count fields
cyl allocation bit map (says RECBLOK exists)
Ø = avail
1 = assigned

RECBLOK - ↑ from ALOCBLOK. Only exist if needed.
↑↓ Pointers
cyl #
Pages in use
Pages in cyl
Page alloc bit map
Ø = avail
1 = assigned
Flag - type of device + use

SYSPCIST - created from SYSPAG macro. DMKSYSSW PP PG
Flag
↑ next lower level in hierarchy
↑ previous sysplist
↑ allocblok currently being used
count fields
↑ sysplist extension
Then for each volume:
volser
Dev number
starting cyl
ending cyl
area type

Pointers: Allocation chain

DMKSYSSW → ALOCBLOK → RECBLOK → swap space
↓
RECBLOK → swap space

DMKSYSPP → ALOCBLOK → RECBLOK → Page Space
↓ ↓
ALOCBLOK RECBLOK → Page Space

DMKSYSPS → ALOCBLOK → RECBLOK → spool space
↓
RECBLOK → spool space

Deallocation Chain:

RDEVBLOK → AlocBlok → RECBLOK → DASD Space

Area Usage:

FMT	Syspag	Anchor	Type
Page	SW	DMKSYSSW	- swap
Page	PP	PP	- Pref paging
Page	PG	PG	- gen paging
Page	PM	PM	- migration
Temp	PS	PS	- spool
Dump	DU	DP	- Dump

At CP Init:

DMKACO (from DMKCPI) builds full pack ALOCBLOK.
DMKVDG creates area ALOCBLOKs.
Chains in hierarchy order. RECBLOKs are pre-allocated for all space but spool + migration.

DMKPST creates ALOCBLOKs + RECBLOKs for paging storage + chains to SYSPLIST

Slot selection

Based on -

level
volume
slot

1. Syspag hierarchy (type determines where to try first).
2. N-Select - what vol. on level
3. Moving cursor - where on volume.
   Start at lowest cyl + move to highest vol., then reverse + sweep back. This favors interactive users.
   Modified Moving Cursor - beginning to end + back to beginning. Used for spool, migration + dump.

N-select - determines how long to use a single ALOCBLOK. When count (ALOCTMI) reaches Ø, go on to next ALOCBLOK.

Dev + Type	ALOCTMI
SW, PM, PS, DU	1
PP, PG	1
3380	10
3350	8

Page Out Blocking (HPO Only)

HPO tries to conserve I/O operations. Multiple IOBLOKs are stacked on an RDEVBLOK + CP can group some together. When a page is ready to go out, CP will queue it on a dummy RDEVBLOK until reaching N-Select number. Then they are turned over to the I/O manager. Now a better chance of

anchored in DMKPAG

chaining these together (+ going on the same cyl.). This process is not done for cached devices.

TDISK Space

DMKTDK does Alloc/dealloc. If doesn't use RECBLOKS. Called from DMKVDA and VDR. In SP, tdisk is by device addr in ascending order. In HPO, it is in the sysown list bottom to top. To get better use, have CP search the smallest space volume first. That way, small requests for tdisk won't fragment the large spaces.

No hierarchy of devices. The user request is specific.

DMKZTD clears dealloced space if CLEAR=YES is specified in DMKSYS. CMSBAT machines should have mini-disks carved out of TDISK.

TDISKs AIOCBLOKs are anchored @

DMKPGTA5	- 2305
8A	- 338Ø
AØ	- 333Ø
5Ø	- 335Ø
A4	- 2314

4-7108 - 16 meg

184

Jim

Page + Swptable Migration

Page Mig - purpose: to free pref. space

Swptable Mig - purpose: to recover high free.
Once swptables are out, pseudo page+ swptables are created in order to page in the swptables.
Only 184 bytes of high free are needed to map a 1 64K segment.

Page Migration uses regular paging routines: simply page in from perf + page out to non-perf. (Same for swap).

DMKSTP - performance monitor, checks flags + when discovers migration flag on, it stacks CPEXBLOK to run DMKPGM to mig a page or DMKSWM to mig a swap slot. Threshold for mig is set by SRM PGFULL. When exceeded, user is selected for migration, along w/ # of pages.

To select: loop through user's seg + pag table. Mig if seg tab invalid, not part of shared seg, + not part of swap set.

Mig runs constantly in HPO 4.2 + mig a little at a time.

Save Mig Ø140

Swptable Migration:

Criteria: Paging > 4% of cycles,
Must be short of high free (Extends taking place).
User not in runlist.

DMKDSP unstacks CPEXBLOK & calls DMKSTR which move swaptables to page buffer.

At PIC 10, DMKPTR call DMKSTR which checks to see if the swaptable is out. If so, bring it back.

Swptable migration is indicated in the segment table entry:

FØ ØØ Ø Ø11
FØ ØØ ØØ41
FØ ØØ ØØ19 - Priv space Mig
FØ ØØ ØØ13 - Gen space Mig

Pseudo pagetable has entries for each segment in sequential order.

VMBLOK, VMSWPMIG → pseudopage table (in in valid bit)
↳ pseudo swptable

has 16 entries

seg Ø	seg 1
seg 2	seg 3
seg 4	...

0	CCPD
1	CCPD

3-4-88
Jim

Timer Management

TOD Clock - came with 370. Incrementing DW register. bits 52-63 are not used. Bit 51 increment 1 μsec. Bit 31 increments 1.0486 sec. Set by SCK instruction. Obtained via STCK. Always runs.

Clock Comparator - DW "system alarm clock". Same format as TOD. Set by SCKC + read by STCKC. Does not increment. Continuously compared w/ TOD + when < causes Ext Int 1004. Int are not kept pending. Be sure to enable for Ext Int before reseting. Also, be sure to reset while handling an Ext 1004.

CPU Timer - same format as TOD. Used as a stopwatch. Positive value is loaded + decrements at bit 51. Int is generated when it goes negative (Ext 1005). set by SPT, read by STPT. Int are not kept pending. Does Not decrement when CPU is stopped.

Interval Timer - 360 Arch. Fullword in PSA. Bit 23 is decremented 300/sec. Not as accurate as CPU Timer. Stops when CPU is stopped. This decrements & generates Ext Int 0080 when negative. Int are kept pending.

TOD is based on 1 Jan 1900 GMT.
Difference between local + GMT is in DMKSYSTZ.
the previous local midnight is stored at PSA TODATE.
Latest STCK is kept DMKSYSCK. CP uses the TOD to schedule deadline priority calculation + controls placement in the runlist. Uses bits 7-38.

Clock Comp. is used to schedule delayed execution. DMKSTP, the performance monitor uses this. The TRQBLOKs are chained in order off DMKSCHTQ (in the scheduler). Top request is current. User requests are also stacked here. ↳ the one currently loaded in the register. At pop, the TRQ gets moved to another queue for execution. The chain will never be empty. The SCHTQ is itself a TRQ with a TOD value of 19 Sept. 2043.

CPU Timer is used to control access to the runlist.

Int. Timer is used only for Dispatcher timeslice + is used to control access to the processor. Timeslice is based on this + since it decrements 300/sec., the timeslice will not be shorter than 1/300 sec.

SET DSPSlice nnnn controls the dispatcher timeslice. Lower this to help interactive users. Lower as long as TVRation is not effected.

PSA values: Pagewait, IOwait, Idlewait. Initially set to 7F — F. CP loads this when waiting, enables for interupts, + when int comes stores value back in PSA. CPSTAT3 tells which field is in CPU Timer.

Determining whether I/O or Pag wait is real hookie. If more than 4% of CPU is for paging, then wait must have been for paging!

PSA

Real Interval Timer X'50'
TOD as MM/DD/YY
last midnight TOD DW
Idle
Page
I/O
} Total System
Problem State
CP Running Status

VMBLOKs

ECBLOK ↑ (permits use of clock comp & CPU Timer)
Running Status flags
Pending interrupt flag
Interval Timer Value
Virt CPU Time used - Problem State Time
Time left in queue
Total time in supervisor - Is really Total Time
↑ TRQBLOK holding real CPU Timer Value in Virt CPU Timer is in real
CR6 savearea
↑ chain of XINTBLOKs (stored while user is disa for EXT INT)
↑ TRQ for delayed sleep of logoff
↑ TRQ for Queue Drop Delay Timer

ECBLOK

Virt CPU Time

↑ TRQ for CPU Time } only one ↑ for each
↑ TRQ for Clock Comp } because that is all that is architecturally available.

IRQBLOK

Clock Comp Value for Int.
TOD that TRQ was queued
↑ VMBLOK
IRA - must be non-pageable.
Real CPU Time Value if the Virt CPU Timer is loaded in real CPU Timer

XINTBLOK

# of Int
Ext Int code
CR 0 mask (bits 16-31)
Ext Int parameter word.

DMKTMR manages timing functions - simulates timing & interrupts. To simulate inst, stack a TRQ - see if user already has one, yes then change it; no, then stack one.
To simulate int, go into VMBLOK & add XINTBLOK.
Also calculates users' V & T Time.

DMKPRG receives int. Try Fast Path Simulation (DMKFPS). If not able, go to DMKTMR.

Inst. Simulation -
If user not EC mode, pgm chk.
If address not Dw aligned, spec exception.
Bad addr - addr. exception
Check storage key
Simulate
DMKDSP

Set TOD Clock - returns CC=0 & does nothing.

Set CPU Timer -

DMKDSP updates VMVTIME & VMTTIME, Virtual Interval Timer, reflect interval time int. to user, & accl. for system wait.

Virtual Int Timer - is in page 0 of the user's virt. storage. SET Timer ON or SET Timer Real causes updating. Int Timer does not run if machine is stopped. When user not running virt. machine is stopped.
Int Timer is decremented with value of timeslice (less any CP READ time). Update occurs prior to dispatch. SET TIMER REAL will cause non-running time to be decremented (MDIWatch!).
With VMA, user's page 0 gets locked in storage & ucode handles decrementing.

With SET TIMER ON you cannot set a Int Timer Value & then go into an enabled wait. You won't be dispatched until an int comes in & the interval timer field won't be update until you are dispatched & thus will never issue an interrupt.

Scheduler maintains TRQ chain. To stack a TRQ, put address of TRQ in R1Ø & call routine. Scheduler does NO validity checking of TRQ. It will stack an invalid TRQ & crash the system.

New TRQs are inserted into the chain in chronologic order. If the same TRQ is stacked twice, scheduler will loop trying to insert it in the chain.

Only one chain even on a MP. (ACTIVERQ)

Pseudo Timers

A device may be defined as a timer and a SIO will bring back time values. Also, DIAG ØC brings back DW CPU timer. Both give MM/DD/YY & HH:MM:SEC & V & T CPU Time.

DIAG 70

Defines field for Virt Time & current TOD value. CP locks the page & these fields are updated prior to dispatch. NO interrupt is required to access the values since they live in user virtual.

Spooling

Lots of control blocks + modules.

DMKSPL - spool file manager

DMKRSP - real spool manager

Finding Devices:

PSA

↑ Punch Table

↑ Prt "

↑ RDR Table

All point to DMKRIO for # entries, Displacement into RDEVBLOK chain, + CUA.

Spool File attr.

class - Determines device + priority

Name - 12 char

type - 12 char

spool id - 1 - 9900

Rec Count - 1 - 16,000,000 (cc is a rec)

Copy Count - 1-255

Real Stor Man.

Allocates frame at SIO.

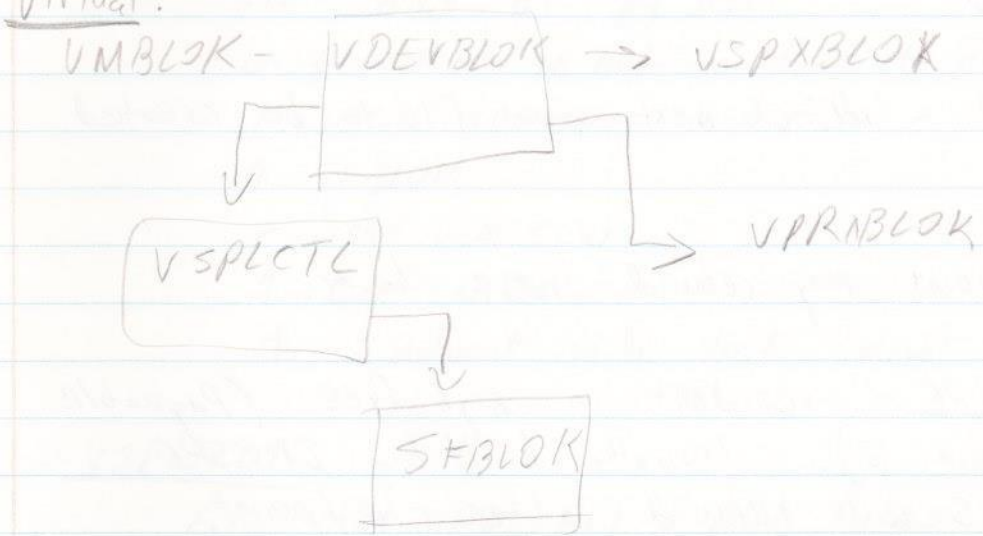
Locked during data transfer. - only the frames in the current SIO are locked.

CP can only have a max of 1280 spool file buffers. Thus only 1280 spool operations can be active at once. Else, PGU005.

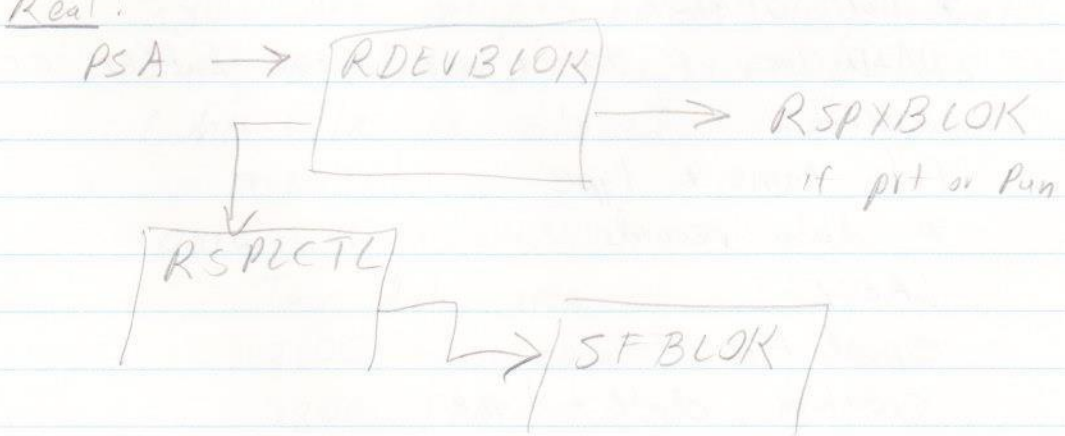
All spool I/O is handled by DMKPAG and uses separate RECBLOK chains exclusively for spooling.

SFBLOK - Spo File Block
VSPLCTL - Virt Sp CB - only around during spooling
VSPXBLOK - ext. to VDEVBLOK
VPRXBLOK - virt prt extension
RSPLCTL - Real Sp CB
RSPXBLOK - extension - 1/real Prt or Pun
SHQBLOK - Sp hold Q Block
SPLINK - Spool Buffers.

Virtual:



Real:



DMKRSP - chains of RDR, prt, & pun files. Has execatchers. Also has a chain of files to be deleted. Chains point to SFBLOKs. PSA points to these anchors in DMKRSP.

DMKRSP

PRT
PUN
RDR
DEL - to be deleted
ACNT -
DUMP - dump spool files
HOLD -
SPT -
ID - id of next spool file to be created.
:

Checkpoint may rebuild these chains.

SFBLOK - resident in high free (Pageable in release 5). Usually built by DMKSPL. DMKCKS will rebuild after checkpoint.

↑ next SFBLOK
DASD loc. of 1st + last page buffer. CCPD
userid
file name + type
data records
LRECL
Spool File #
creation date + time
copies
spo file class

Distribution code
Chkpt map # - what slot on what cyl.
form #
Control flags

VSPLCTL

Exist for open files (temporary)
Used to chain SFBLOKs to VDEVBLOK.
Built by DMKVSP (or DRD for dumps).
Contains info to need to process file:
Virt addr of user's CCW
DASD loc of current page buffer
Virt add of pg buf
Displacement in buf to start of next rec
recs remaining in buf
↑ SFBLOK
current user CCW
↑ Buf area
↑ Indirect adr list (might cross pg boundary)

VSPXBLOK

Exists for open prt + pun files;
3800 prt setup; holds tag info.
Built by DMKCST (tag) + DMKVDS (3800).
When file is closed, the specifications are set.
Contains:
Tag data area
3800 flash count
3800 FCB
3800 char arrangement Table
3800 copy mod
Name of flash overlay.

RSPLCTL - exists for open real files.
Restart info (pt runs out of paper).
Contains:
Restart CAW - CCW adr
DASD loc of ccw pg
Real Adr of pag but
virt Adr " "
& SFBLOK

RSPXBLOK - ext to RDEVBLOK for every real Prt & Pun. Has form info, etc.

SHQBLOK - exist for each user w/ files on HOLD. Permits future files to be held for a userid.

SPLINK - this is the spool data:
header + (optional) tail.
Chained from SFBLOK.
Active SPLINK lives in DPA +
while there chained from RSPLCTL +
VSPLCTL.
Built by DMKSPL.
Contains CCW to drive real device
and data.
Only first SPLINK has Tag, FCB, +
trailer w/ date/time stamp +
3800 info.
↑ + ↓ SPLINK

Note: cannot backspace a 3800 printer.

Data area of SPLINK is 4080 bytes & is not compressed. Data is actually interspersed with the CCWs. First CCW is no-op with bit 35 (skip) = Ø. RSCS looks for this & sees that 135 bytes of TAG exist. Next is TIC CCW & is FCB CCW, & then write, tic, data ... At end of CCW chain is another NO-OP so CE/DE can come back simultaneously. (This is 370 arch).

The CCW addresses are virtual. Write CCW will be 00000C. Tic will be C + length of data.

Spool File Recovery

Cold - nothing recovered

Warm - SFBLOKs
RECBLOK
SHQBLOCKs
} all recovered

CP simply does chaining to high free.

CKPT - Read every rec of every spool file to build RECBLOKs. Does everything sequentially. No overlapping I/O.
If invalid chain found, recovery stops.
Open Acct records lost.

Force - if an invalid file is found it is thrown out & the chain is repaired.
No indication of which files lost.
Really no advantage to using CKPT except you know that no files were lost.

On shutdown, all info is taken out of high free to build RECBLOKs & written to warm start area.

Chkpt - each entry is 128 bytes. (32 rec/page)
1-6 Chkpt maps. Each entry in the map describes the respective chkpt 128 byte entry.
Table: X'FFFF' = empty
AAAA = SHQBLOK
E01E = status of dev 01E
FEFE = End of Table

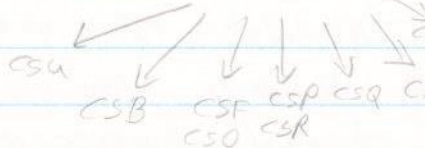
DMKREPSTR ND point to the starting & ending cylinders.

To delete, find map containing entry, read into storage, then read the record in, & invalidate chkpt entry.

DMKRSPM1 - M6 points to virtual address of the chkpt maps. This is in CP's virtual storage.

console CMD → DMKCFM

DMKCFC - has table of commands, abbreviation, & pointer to routine.



DMKSPL - Build the SFBLOKs & do the chaining. DMKSPK handles deletes. Two routines: 1 stacks SFBLOK on del chain & another that deletes.

DMKSPL - flow:

To open virtual spool file:

Build VSPLCTL
Get a page (DMKPGU)
Get DASD Buffer (DMKPGT)
Get Spool # (DMKCKT)
Lock page (DMKPTR)
Init SPLINK
If 3800 get large VPRXBLOK

To Close:

Update SFBLOK
Read 1st SPLINK (DMKRPA)
chkpt (DMKCKS)
Validate & send msg.

DMKSPK - Delete routine:

To delete:

run chain of del SFBLOKs.
Reclaim DASD space (DMKPGU or DRD) single-threaded
Delete SFBLOK
Give back spoolid.

User is placed in I/O wait while spool file management takes place - even if no physical I/O occurs.

DMKRSP - Real Spool Manager
Holds spool anchors & counters:
↑ RDR, PRT, & PUN Chains
↑ Acct card Buffers
↑ Delete chain
↑ Spool Tape Delete Chain
Spool ID counter
↑ SHQBLOK
↑ active Monitor SFBLOK
Virt addr of Chkpoint Maps
⋮

RSP reads the SPLINK in & adjusts the write CCWs. It knows where it put the SPLINK & thus changes the CCW addresses. It maintains an address counter as it works the way down the SPLINK. Then it alters the CAW to point X'10' into the SPLINK.

Nucleus Creation

3-7-88
Gary Weinhold
914-347-2390

VMFLOAD Loadlist CNTRLFile - exec the builds a nuc in the reader. The loadlist is a list of modules. the CNTRLFile determines which version of various modules to INCLUDE. Output is an IPLed card-image file. Movefile to tape from RDR.
Loadlist determines the order of modules in the nucleus.

Building CP

DMKCD00E - DMKCD00E LOADER - stand alone loader for deck (c/ 4 bootstrap cards)

DMKRSA - first Nuc mod

DMKSLC - if V=R present (1 card: SCC adr of where to set the location counter as we build this nucleus).

DMKACR - order of resident nuc modules does not really matter, except DMKCPE which must be the last.

DMK CCH -

CPE - end of nuc

TAP - 1st pageable mod. Aligned on page boundary. Must be first.

VMD } Pageable modules
⋮ }

DMKSAV marks the upper limit of pageable modules that will be available before initialization of the paging subsystem.

CKP - now split:

CKD } Don't move, change, or add to these.
CKN }

LDT - Loader Terminator. Must be last card.

IPL 00C loads and executes the loader: 4 card bootstrap loads DMKCD00E reads CP text decks, links all modules (resolves addresses), creates an executable CP nuc in storage, + transfers control to DMKSAVNC. Any errors are printed. Also, entry points are printed. Comment cards are considered errors. The result is the NUCMAP. The LDT card has an entry point which must have been resolved. This is DMKSAVNC.

Loader wait states are documented separately from standard WSCs.

DMKSAVNC writes the Nuc to disk (using rol + loc. found in DMKSYS). The current TOD is placed in DMKCPI. Ends in disa wait state '12'.

DMKSAV saves the data needed to IPL in order to create the IPL text. DMKCKP becomes record 2. CKP must be able to find the rest of the nucleus. The bootstrap gets DMKCKP running.

Nuc load termination:

CSC = 12 - normal
10 - wrong volser
11 - fatal I/O error
ØA - Mch Chk
14 - Storage failure

Gary

Initialization

IPL text:

Rec 1. 24 bytes:

IPL PSW = ØØØc ØØØØ 0000 1000

CCWs = Ø6ØØ1ØØØ ØØØØ1ØØØ Read

Ø——Ø Ø——Ø

At end of chn pgm, the psw at loc Ø is executed. The Read CCW puts DMKCKP (1 page) into storage loc '1000'. The second CCW is not used. Chain bit is not on. PSW begins execution at DMKCKP.

DMKCKP checks PSA CPID to see if the value is CPCP or WARM to decide if warm start data should be saved. If either then it will try to save warm state data.

Warmstart may also be written at shutdown or abend. At abend, after dump is taken DMKDMPRS will do an I/O to bring in DMKCKP and start it running.

If CPCP then save warmstart data and end wsc = 8. If anything else, keep coming up. When coming up DMKSAVRS uses the warm start data.

PSA area checked by DMKSAV:

CPCP - save warmstart + shutdown

Warm - " " + restart

xxxx - normal IPL on cold machine
No warmstart data saved.

This fields are then changed to:

WRM - for restart

COLD - for normal IPL.

Init Sequence:

DMKCKP reads in CKD, CKF, CKH, CKM, + CKN right after CKP. DMKSAV gets loaded to its correct location in NUC. Throws aways all I/O interrupts (for users pounding their keyboards). Then check to see if warm start.

Warmstart data: save -

- avail console
- addresses of en terms
- Acct info
- log msgs
- SFBLOKs (from DMKRSP ↑)
- status of UR devices (from PSA ↑).
- RECBLOKs (from SYSPCIST + ALOCBLOK)
- Spool Hold Queue Blocks. (SHQBLOKs)

An IPL over a running system 4/0 clear option will save warmstart & shutdown.

DMKSAV - loads nuc up to itself but no further. Throws out all I/O ints except from sysres. Exits to DMKCPI. WSCs =

ØA - Mch Chk

11 - Fatal I/O error

11 - Storage error (no attempt to recover from double-bit).

Real Initialization Overview

All we have so far is a Nuc in storage. Init storage, work areas, control blocks, allocate save areas, mount & verify DASD, init directory, logon operator (now send msgs), start DMKDIO MIH, test for VMA & extended storage, allocate dump file, init error recovery, & wait for work.

DMKCPI & J control init.

↓

DMKSTA - init stor

PMA - init for a possible future PMA guest

API - AP/MP support if required (2 BAs)

MNT - Mount Devices - Search every RDEVBLOCK

Record states, volser, & if CP-owned.

SEG - init sys Virt store (SPLINKs, etc.)

Pageable virtual store. Maps to seg & pag tables.

OPE - init Operator; looks for alt if req.

TOD - init clocks via prompt

Both proc. will synch on this

UDR - Build index into directory (pageable)

Build override file to chg class of CMDs.

CPJ - finish init

DMKCPJ
↓
WRM - determine type of CP start
IDU - alloc dump
OPE - logon Operator
CQN - SPool File Msg } simulate commands
CSO - start UR dev. }
SEG - init I/O to bring in the rest of pageable nucleus from sysres so it can be later paged out again to page space.
IOE - I/O err recovery
IOG - Mch Chk recovery
STP - Sys Perf Module: set disp timeslice & init monitor gathering.
OPE - fake STA MON cmd + AUTOLOG1 as if the operator issued these commands.

DMKCPI - first thing it does is to set up the PSWs in PSA:
Restart - DMKPSADU
External - DMKEXTIN
SVC - DMKSVCIN
Program - DMKPRGIN - all lead to PRGnnn abend. { Is briefly overlaid during exec. of STAP instruction.
Machine - Disable wait 'B' at first, later will point to a valid entry point.
I/O - DMKIOTIN

Init PSA fields, CRs (0,1,2,6,14,15) Clock comparator, CPU Timer, CPUID, & TOD clock. Next issue STAP instr. to see if the proc. is an MP (Store Address of Processor - Not installed unless machine is AP/MP capable). Init CPSTATUS fields in PSA

Find processor id of other processor (IPUADDRX). SIGP requires processor id; so each is tried. These run from ØØ-3F. It stops after finding one. All this goes into the absolute PSA. APSTAT1 records this.

Determine VMA availability with E612 instr. that returns ucode level. This is not Ø0Ø. Intercepts pgm chk while executing.

Shift RDEVBLOK to 3 bit shift if CP Assist is avail. & if less than 256 K. CPSTAT5 tells 3 or 4 bit shift.
Call DMKSTA.

DMKSTA - find storage size. Use SSK to each 2048 byte area to set up keys up to 16 meg and watch for pgm check. Then, use TB instr to keep looking above 16 meg. Store in DMKSYSRM.

Find if real size is larger than gen size to free unneeded core table entries.

Reserve two frames for PSAs. DMKPSA PREFIXA & B record ↑. Not yet filled in.

Test for 2K/4K storage keys. VM/SP won't run w/ 4K keys. (CPSTAT4)

Allocate high free. Use DMKSYS if entered, or calculate default: Remember to omit V=R area.

pages/meg

<16m 16pp/m - 1

>16m 4pp/m of hi free of an MP.

Multiply final result by 1.25

PSA PRIMEHI CO: In each PSA they each have their own.

{ Allocate prime storage or default (10%). alloc out of top end of high free.

Alloc trace table. <16 4pp/meg (w/o V=R)

>16 1pp/meg

Set trace start, current, & end. ('C', '14', '10') (Also DMKPASAST)

Set Tracefly (+'400') to 'FF' to indicate what events to trace. There are no commands to selectively trace. Just set the bits.

Discover what's left for DPA and init coretable. Freelist is built. Update coretable for CP pages as resident & CP owned. Call DMKTRET: will take end of DMKCPE to page boundary to add to avail free storage.

Freelist is built & ready (paging not yet happening).

DMKSTA returns to DMKCPI after preparing storage for execution.

DMKCPINT

Alloc areas in High Free. Preallocates CPEXBLOKs. Point RUNUSER & LASTUSER to system & set up prefix registers.

DMKAPI:

SIGP xx STOP/START is used to control the other processor. A SIGP xx RESTART gets the other processor going after initializing the NIPL PSA loc 0 to an entry point in DMKAPI.

API copies absolute PSA to the other two locations and initializes the appropriate fields (for each (mirror image) IPUADDR, CPUADDR, PREFIXA, IPUADDRX, CPUADDRX, and PREFIXB. Also, alloc save areas for 2nd processor. Init PXA, PXB for PSA extensions. Restart new PSW points to DMKAPIPR for AP. While API is running on the other side, the IPL processor loops (they are both using absolute PSA at this time). When API completes, start using prefixing and system lock for a double guarantee. Return to DMKCPI.

DMKAPIPR - fixes restart new PSW back to dump routine. Check for CP assist on this processor (if not, neither can use it). Update APSTAT1. Find if system is AP or MP. If MP, change ARIOCT to point to appropriate chan table (since this processor has separate chan. set). A dev could be offline to one processor. Turn off spin of other processor & wait (DMKWAI).

DMKMNT - mount routine inits I/O subsystem:
get I/O lock since we have to run init routine on both processors. Touch all RDEVBLOKs in order of chan table. Turn of disabled bit in RDEVBLOK when dev answers. If never answered, then it stays disabled.
(RDEVALT) Flag alternate paths where discovered. (RDEVPTHS)
Alloc RSPXBLOK for UR devices.

Find if Receive/Release hardware exists.
If so, flag.
For DASD, read volser + detect duplicates.
Check if CP formatted (ownerid field in volume lable = "CP370").

For CP owned volumes: find directory or override pointer in vol. label. If the volser is same as sysres, then store as primary. If not, these are alternates.

DMKACO store ALOCBLOK pointer in RDEVBLOK.
Also, set CPOwned in RDEVBLOK.

DMKSEG init CP segment, page, and suptables and core table. Check for VMA, EF (test protect...), + DAS (pgm call, pgm xfer) and record status. Check for extended key operation support and hardware segment protect. (Temporarily overlay pgm chk new PSW).

DMKOPE - locate operator console. Call DMKTOD if not auto-starting. DMKTOD will synch multiple clocks if present.

DMKUDR - allocate CP vs for all entries in the director → userid, ACI group (RACF), DASD location of directory entry. Anchored at virtual address DMKSYSPL. (169 entries per 4096).

DMKCPJ - check to see if interval timer is running - set a value & loop while waiting for int. If no int, issue msg to turn on interval timer and loop again.
Determine type of start ⇒ warm. Get type from operator. If response is shutdown, stop w/ WSC 6 & save warmstart data. Call DMKWRM.

DMKWRM -
Warmauto - read warmstart data, en termsx lines. Put operator back on same console. Then join normal warmstart code.
Warm - read w/s data, & continue as if cold. DMRRSP is recreated as it was at shutdown. Also, find ACOCBLOKs and point to the saved RECBLOKs.
[Never warm start after changing the sysown list].
Init status of UR devices (classes, etc).
COLD - init chkpt cylinders. Chkpt spool & hold queues using DMKRSP anchors. If true cold, these will be empty. Re-init w/s cylinders.
CKPT/Force - Restore spool files. Go through every spool file & recreate RECBLOKs that can be hung off of ALOCBLOK. If the SFBLOK chains are broken, then a FORCE is required.
Re-init w/s cylinders (trash them).

DMKCPJ - set CPID to "CPCP" to indicate system is running. If it crashes warmstart may now be saved. Call DMKIDU.

DMKIDU - calculated needed dump space, build SFBLOK, RECBLOK (DMKDMPSF), & allocate it. DMKDMPDV lists the device - dump, temp, or printer. Write symbol table (DMKSYM) to dump area - a primative Nucmap. call VDG.

DMKVDG - build paging ACOCBLOKs. TDSK (clear TDSK w/DMKCPX). RECBLOKs for PP & SW. (map shows all area avail. For spooling RECBLOKs are not pre-built. Call DMKPST to init paging storage. Also calls Vector Facility init.

DMKOPE - logon Operator & issue messages
DMKCQN - issue msg on sp files
DMKCSO - start CR
DMKHVD - init DMKKCPP bit map of VM/SP release level.
DMKSEG - bring in all pageable modules not in storage & page them out. - Go onto flushlist.
DMKIOE - init RMS. DMKMCHIN becomes mch chk new PSW.
DMKNLD - network load for 3705.
DMKSTPX - init scheduler info & start monitor.

RMS Init - Call DMKIOGF1. Issue STIDC to find avail channels. Determine if a chan exists w/o a RCHBLOK. Build chan config table in DMKCCH. Set CR15 to extended logout area. IOEL at X'AC' in PSA.
See if error recording cylinders are formatted (DMKSYS). They are formatted if not valid. Should always be valid.
Machines since 3081 do not do extended log outs & thus set a bit in the CPUID. CR15 will be Ø.

Termination

DMKCPS - handles shutdown. If AP switch to IPL processor & quiesce. If MP, set CPSHRCK in CPSTAT2 in both PSAs. Stop prob state exec via CPINITD in APSTAT1 in both PSAs.
Zero CR8 to stop MON. Close console file for operator & shutdown tube.
Record errors by chaining through RDEVBLOKs (OBR records). Record DASD stats w/ DMKDSB.
Stop prefixing & goto dump routine

DMKDMP - enter at DMKDMPRS & CPCP will be set in CPID field.

Abnormal - just about any pgm chk after initialization. The PIC is set in the PRG msg.
Abend macro - chain broken, DPA exhausted, etc.

SVC Ø → goto SVCDIE
DC CL3'ABC', AL1(N)
↑ abend code

PSW Restart: saves GPRs at X'500' & pass CNTL to DMKRSA X'0' Abend code = PSA002. Goto DMKDMP.

enter at DK

DMKDMP - detect if recursive. Save clock & regs to store status (software STSTS). Find if AP/MP. If yes, stop & store status on other processor. SIGP other proc. so it loads a WSC. Store status goes to absolute PSA, so move it to prefixed.

DMKDMPSF is the SEBLOK

Find dump device w/ DMKDMPDV. Default is printer.

CPID = 'CPCP'

If dump is to spool area, then things are a bit different. DMKSYM is already out there.

DASD format of Dump

1 - DMKSYM

2-6 - reserved for now.

7-12 - storage pages 0-5

3-6 - storage keys built in storage pages 2-5

3-8-88 Gary

Interrupt Handling

CP code always runs w/ I/O & Ext int disabled, in sup state, DAT off, EC on, & key 0. Subclasses of Ext Ints may be masked by bits in CR0.

FCIHs

MCH - DMKMCHIN	I/O - DMKIOTIN
SVC - DMKSVCIN	Ext. - DMKEXTIN
Pgm - DMKPRGIN	Restart - DMKRSADU

Both CP & virtual machines can issue SVCs & have program checks. Therefore, to handle, CP checks the old PSW. If problem state then let the user handle it (except paging, of course). Since users run prob state, lots of privilged operation pgm checks occur for SCP guests. These get simulated (DMKPRV, DMKFPS) and not reflected to the guest.

The Dispatcher (DMKDSPWI) has an enabled window (3 ins) right before dispatching a user to cut the probability of hitting an I/O'rupt as soon as user PSW is loaded.

The CP module DMKWAI loads an enabled wait & CP doesn't care if this is interrupted.

Restart

Save regs @ DUMPSAVE
Set CPABend to PSA000
Get dump dev
RECBLOK address
SFBLOK
Take Dump
set CPID to 'CPCP' or 'WARM'
Simulate IPL

Machine Checks - 2 kinds: Recoverable, Unrecoverable. Storage errors are usually recoverable. All others are usually not. Single bit errors are corrected by hardware & reported simply to log on EREP. Multibit errors

are not recoverable. CP can recover if it can get rid of the user & take the page frame offline (CORETABLE). If CP, the system abends. The user may be crashed even if the guest could handle it (like MVS). If the page w/ the error has not yet changed the page, then he is in luck; the bad page is abandoned, coretable marked bad, & the page is retrieved from aux store.

MCIC begins at X'E8'

I/O Interrupt → DMKIOT (PMA interrupts go directly to guest).
3 possibilities: Wait (Active Wait), Enable window, or user running. So, determine state. Save stat only if user running.

Check int dev addr &
located dev, cu, & chan blocks for dev. (DMKSCN)
Switch timer to sys overhead
Check if dedicated chan & if so, pass directly to user. (DMKVIO)
RDEVBLOK points to IOBLOK. Requeue this to the dispatcher if I/O completed normally.
CE & PCI cause the IOBLOK to be duplicated: one to dispatch & one to remain hanging on RDEVBLOK.

- Unsolicited interrupts (no IOBLOK off RDEVBLOK)
- Deferred CC -
  if Ø do nothing, no need to pass up
  1 - DMKIOS - if Dev Busy wait & try again, if problem, reflect to user
  2 - error processing - probably crash system
  3 - DMKIOS

Unsolicited Int:

Create IOBLOK & figure who handles

Some are ok - like hitting ENTER @ Terminal.

IOBIRA module

TTY - DMKCNS

Remote 3270 - DMKRGA

Sp Dev - DMKRSP

DASD - DASD

Local 3270 - DMKGRF

3704, 05 - DMKRNH

Ded. Dev - DMKVIO

If an interrupt comes w/ unit check or from an unknown device CP will do a Sense. (May simply not be in DMKRIO).

CP has no hot I/O recovery. It may build IOBLOKs until mem is full or may thrash throwing the int away.

- Errors:

If unit check, CP does sense before talking to user. When user finally does sense, CP pass the sense data. Record error & retry to redrive I/O. Sense data goes into IOERBLOK until passed to user.

To free the controller.

↳ hangs off IOBLOK & ends up off VDEVBLOK.

CP does no recovery for virtual machine I/O. If it is to be done, the user must do it.

Chan check - DMKCCH handles hardware related checks. Chan pgm chks, chan prot checks, & chan chaining chk are not hardware-related.

For CCC, IFCC, CDC, try resetting the chan. + try again. A WSC Ø02 is loaded if the path is the last to a critical CP-owned dev + the reset fails.

DMKIOE handles CP I/O errors by examining the IOERBLOK and the I/O extended logout.

Ext Interrupts - similar to I/O at the beginning. Determine class of int (Ø (3Ø), 1-timer, 2-MSSF, AP/MP). Update count in PSA, cut trace entry, + put class + code in PSA (X'85' + '87')

	Types		Actions
Class Ø -	0040 -	Button	Disc System Oper.
	0080 -	Real Interval Timer	DMKDSP
	0100 -	Virt " " " Timer Assist	µcode → DMKDSP
Class 1 -	1003 -	TOD synch check	DMKCLK
	1004 -	Clock Comparator	DMKSTK
	1005 -	CPU Timer - User's Whole Timeslice Expired → DSP	
Class 2	2401 -	MSSF (Console) uses HCBLOK	DMKMHC
AP/MP	1200 -	Malfunction Alert (MFA)	DMKMCT
	1201 -	Emergency Signal SIGP	DMKEXTSL
	1202 -	SIGP (External Call)	DMKEXTSL

XINTBLOK are used to reflect an EXT Int to a virtual machine.

→ Try to keep running in UP mode. Cannot do if CP was running on the failed processor.

Single Processor Mode - CP has only one processor of a diatic. MVS will try SIGPs to other processor. Therefore, SIGPs are queued to the V=R user.

Other user has absolute PSA plus copies for each processors.

SVC Interrupts

If prob state user issued SVC, then all are reflected to user except SVC 76 (logrec DMKVER) and SVC 179 (DMKCFM adstop). Usually handled by VMA.

If CP:

SVC 0 - abend
4 - reserved (abend)
24 - switch AP/MP - execution streams switch processors.

used for linkage among CP modules.

- 8 - Link - expansion of CALL macro.
- 12 - Return
- 16 - Release Save Area - Return to caller's caller
- 20 - Acquire Save Area - gets save area to use.

When adstop entered, an ADSTBLOK is created + hung off VMBLOK. Check address of SVC179 and look in ADSTBLOK to see if it matched (+ code that was overlaid). If yes, stop execution. If no, reflect to user.

For SVC 76, verify that error needs recording + return control. MVS software recs appear invalid + therefore are passed back to the guest. Check GPRs 0 + 1 to see if DOS or (OS, VS1 or CP). VM second level, uses SVC 76.

SVC savareas: 12 DWs:
caller's return address R14
caller's base R12
R13
Save areas
Normally resident in prime storage.
Active savearea chaining: Savearea↑ chain together.
Exit macro expands to:
CM R0,R11, SAVCARA
SVC 12
Return address is placed in the PSW and the machine is permitted to run. Savearea is good for debugging. The runing pgm, however can alter the saved registers

SVC 24 - Put CPEXBLOK describing my current work on the other processor queue (marked w/ affinity) & go look at the Dispatcher. The switch macro has a SIGP

Program Check Interrupts

MC causes a pgm check + is used regularly - the only normal CP pgm check.
DMKPRV to simulate:
DMKPRG - FCIH
DMKPTR - page tran PIC 10 + 11 (TREXADD & 40' PSA)
DMKPAG - shadow tables
DMKPRV - PIC 2 (or 1 on occasion) DMKFPS is a front end. Simulates any priv operation.
DMKVSI - virt I/O
VCN - " console
VSP - "

VMINST instruction being worked on.

pageable

DMKDIVC/D/E/F - Diagnose X'83xx'

TMR - Timer operations (store, set)

DMKMPS - Virtual MP Mode - if in Single Processor Mode (SPM).

DMKFPS - duplicates other modules' functions but does it faster. Does 11 instructions.

Conditions: PER not active
instr on single resident page
virtual supervisor
virtual ECMode

PIC 13 - Special Op Exception → for the semi-priv inst:

MVCP
MVCS
PC
SAC
SSAR
PT

result in a PIC reflected to the user (for guest simulation) to pretend the hardware doesn't support these.

PIC 80 - PEREvent → DMKPERIL if for CP. If for user, save PER code + addr + ECBLOK, and notify user.

When reflecting pgm check, try to detect PIL00P. Check pgm chk New psw + instr pointed to from it.

Privop processing must move the users page down below 16meg for simulation.

DMKPRV has counters for each type of privop. VMINST holds instr while being worked on (it may be residual).

3-9-88
Gary

Real I/O Processing

PSA pointers: — Entry Length

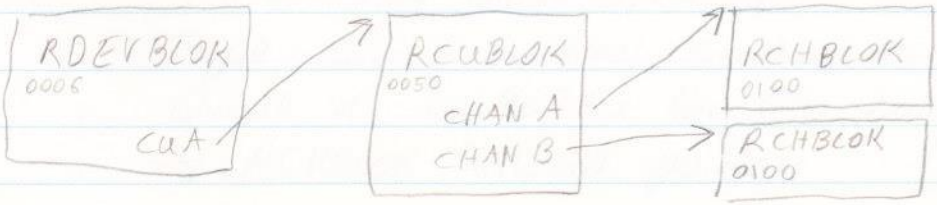
ARIOCT → Real Chan Index Table
Holds displacement into Chan Blok Table for each chan (since they need not be coded in order). X'FFFF' means chan does not exist.

ARIOCH → Real Chan Block Table — X'60'
Holds Table of all possible control units with displacement.

ARIOCU → Real CU Table — X'50'
Compressed displacement (4 bits) into DEV Table.

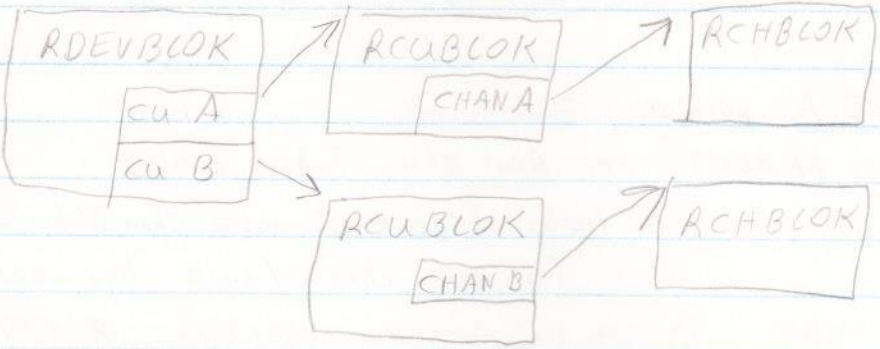
ARIODV → RDEVBLOK Table — X'70'

For MPs, two sets of channels may exist. Since only 32 chans will fit on each processor, CP will build a second chan table. The other PSA will have a unique Chan Index Table. The two Real Chan Block Tables are contiguous. The same CU & RDEV blocks are used. Each proc. has discovered during init. which CUs & DEVs it can access.



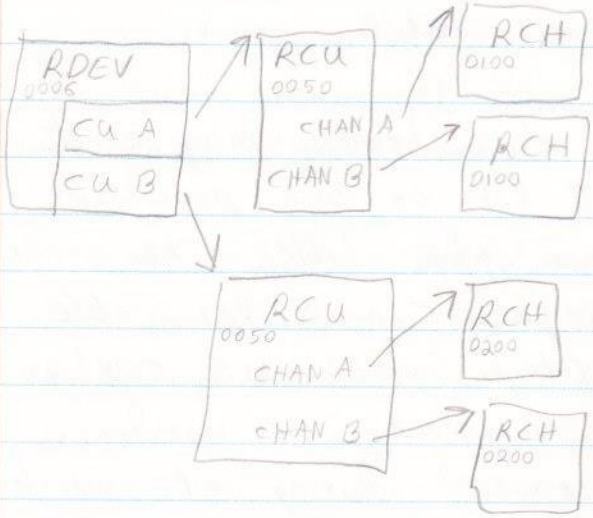
The two paths must lead to the same device.

On an MP gen, an ALT CU may also be specified with the same address:



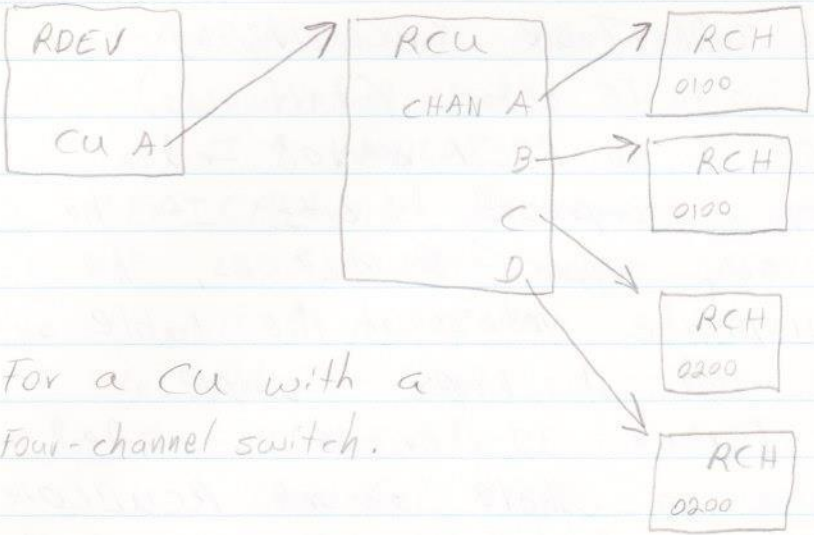
Note that the CHAN ↑ from the ALT CUBLOK is the 'B' field. This lets CP know that I/O may continue in event of a CU failure.

The ALT CU can be at a different address. This works on a UP too.



Dev = 156

Dev = 256



For a CU with a four-channel switch.

All pointer are fullword pointers & are set to Ø-Ø when null.

RCHBLOK - the result of RCHAN macro.

Holds:

Address in form xxØØ

Owning Processor - ØØ4Ø, ØØ43 for example.

Status - chan busy, offline, dedicated (if ded has VMBIGN

Type - selector, Byte, Blk mux

↑s to IOBLOKs queued on chan busy (RCHFIOB, LIOB)

CU index Table (32 entries, halfword each

00 08 10 18

20 28 . . .

If dev is greater than 18 & the 18 entry is 'FFFF' back up on entry - to the 10 entry.

RCUBLOK - anchore in PSA ARIOCU

Address in form ØØxØ or ØØx8

Status - Busy, Online, Scheduled

Type - Shared Sub-channels, Subordinate RCUBLOK

Queue of IOBLOKs for CU busy

4 RCHBLOK fullword pointers

Device Index Table (RCUDVTBL) -
up to 16 entries (halfwords).
FFFF if no RDEV macro. Index
entry is compressed 4 bits. If the
CU only supports 8 devices, the
appropriate entries in the table will
be used. This keeps it simple.
If feature = 32 devices is coded on
macro, a whole second RCUBLOK
is built with the address of the
second string in the address field.
This 2nd RCUBLOK has a RCUPRIME
pointer & the subordinate RCUBLOK
bit is on in the TYPE field.
(instead of RCUCHA field)
The busy field in the first is the
one that counts. The 2nd has
Ø pointers to channels. All IOBLOKs
will be queued on the first.
An RDEVBLOK will point to which-
ever RCUBLOK is needed.

RDEVBLOK - anchored by ARIODV
Address → ØØØx
Type, Class (Data Areas, Appendix A)
Device Name - 3380, etc.
Status: RDEVSTAT & STA2, 3, & 4
Busy, scheduled, offline, dedicated
↳ can mean that SIO came back busy.
That CU will then owe a DE when free.
↑ First & Last IOBLOK queued for device busy
↑ to active IOBLOK (RDEVAIOB)
↑s to RCUBLOKs (A & B)
Path Status (8 bits) Ø = avail xxxx xxxx
CU A CU B

For dedicated devices:
Virtual addr
↑ VMBLOK
For CP-owned devices (in same CB location): } overlay
sysowned index (RDEVCODE)
↑ ALOCBLOK
For DASD:
Volser
Last Cyl access
links to this device (a system)

Schedule Bit – when IOBLOKs arrive, they check first RDEV then RCU, then RCHAN. At whatever level a busy is found, the IOBLOK is queued. When queued, it sets the schedule bit on the lower levels. A second IOBLOK would spot the schedule bit & queue on that level. It knows that somebody is queued on need dev or CU & that a wait is required. When dev is free, I/Os are de-queued in Chan, CU, & Dev order.

IOBLOK (can't format w/ Analyze)
When active, ↑ from RDEVBLOK. Can queue behind Chan, CU, or Dev. When I/O done is queued to dispatcher w/ addr of routine to handle interrupt (SCIH). Holds: VMBLOK, virt dev addr, ↑ to real chan pgm (CAW), (IOBCYL) Cylinder for first seek, + if queued: chain pointers (F+B). After I/O interrupt, holds CSW + pointer to sense data if stored. (IOCRBLOK)

DMKDSPRQ

→ if ≠ than not for user

IOBADTO

Dispatcher IOBLOK queue is usually empty since these are high priority tasks.

I/O comes from either CP or user.

CP - Paging
Spooling
Console Functions
Dir Maint
error recording
SPTAPE
Monitor (if tape used)
Dump
} CP must create the channel pgms.

Virtual Machine - prepares own chan pgms.
SIO to dedicated dev or to minidisk
Diag I/O
SIO to virtual spooling dev
SIO to " console
SIO to " CTCA
CP must be able to convert these channel pgms (& for CTCA, simulate hardware).

Processing Flow

DMKIOS - the I/O Supervisor. Needs IOBLOK as input (& channel pgm).
DMKVSI - Virt SIO re-formatted & create IOBLOK
DMKDGD - Diagnose I/O
DMKGRF - Graphics (for local)
DMKPAG - Paging
DMKRSP - Real Spooling
DMKIOQ - Extension of DMKIOS to queue I/O
DMKIOT - Interrupt processor - updates IOBLOK w/CSW & passes to dispatcher.

The dispatcher then passes control to the appropriate I/O SLIH - RSP, PAG, etc.

DMKIOS has two entry points:

QV - virt machine I/O (real addr not yet in IOBLOK)

QR - CP I/O (IOBFLAG indicates CP request)

R10 → IOBLOK

R11 → VMBLOK (stored in IOBUSER)

R8 → RDEVBLOK or VDEVBLOK.

Queuing Rules:

Q IOBLOK from lowest Busy component

Mark any lower components as scheduled

Sched component treated as busy for subsequent requests.

For Alt paths:

Search primary path first

On MP's, search current processor first

Queue on all busy paths (copy IOBLOK + chg dev addr.; link IOBLOKs together - circular links)

When avail path is found, dequeue other IOBLOK.

The duplicated IOBLOK are smaller → mini-IOBLOK. X'18'

Has only dev addr (for whatever path), flags, * IOBMINI bit

IOBLINK to original IOBLOK, chain points for other queued IOBLOKs, & ↑ to I/O component BLOK (IOBMQDTO).

Mini-IOBLOKs are pre-allocated at Init + limit the number of ALT path queue requests. Max is something like 15. DMKIOSMQ is ↑ to high free area where avail mini-BLOKs live.

When a real I/O is necessary, the addr from the mini is merged into the real IOBLOK & the other mini BLOKs are discarded.

Find a path to a dev (DMKIOQ):

Check RDEVBLOK
Check CU A then CU B
Check chans A-D
No paths; then return to caller's caller. (SVC 15)
DMKIOS's

Queuing -
- IOBLOK queued off first unavailable path.
- Mini-BLOKs queued of other paths.
- Search always starts at primary address.

Queuing order: on Dev:
- Get IOBLOK Lock
- if not DASD, FIFO
- if fixed head paging - queue before moveable head requests
- if moveable head DASD - queue in ascending seek cyl adr order.

If CP, DIAG, or guest SIOF then issue an actual SIOF. If the guest issues a SIO then issue an actual SIO because guest would not be expecting a deferred CC.

If CC=3 to SIO(F) then mark path offline & try all paths (since only cu may be down). After all path tried unsuccessfully, mark Dev offline.

Moveable head DASD IOBLOKs are dequeued in alternating ascending & descending order (as per RDEVSKUP).

DMRDIO - delinquint Int. Detection. DIO sets a flag in RDEVBLOK, that is turned off by int processing. → if Dev found busy or Sch. Thus, when DIO comes back it can see if the device has stayed busy too long. If the dev is ok, then Busy will be on but DIO bit will be off. DIO then sets DIO bit back on + will check again later.

If both Busy + DIO bit on, then int. delinquint:

Attempt HDV (halt device)
Simulate IFCC (and software may be able to redrive)

If device is Sch + no active I/O, then simulate CU End.

If busy bit on + no active I/O, then simulate device end.

Channel End is not simulated.

Always set MIH time longer for second level than for first level.

Gary

Virtual I/O Processing

VMBLOK anchors: VMCHTBL → index into VCHBLOKs. 16 entries.
VMCHSTRT → Virtual Channel Blocks (VCHBLOKs) x'30'
In numeric order w/ gaps for undefined channels. (2nd level is limited to 16 chans).
VMCUSTRT → Virt CUBLOK for each 16 devices x'28'
No options for 8 or 32 devices.
VMDVSTRT → VDEVBLOK (x'30' length)

VCHBLOK

chan addr ØxØØ

Status - busy, int. pending, dedicated

Type - selector, Block mux (Only Ø can be byte & is assumed to be so). The type of devices on chan determine type.

↑ RCHBLOK if chan is dedicated.

Virtual CU Index Table (16 entries)

CU Int. Pending bit map (halfword), 1 = pending.

in VMBLOK
VMCHINT chan pending

VCUBLOK - Anchored off VMBLOK

addr ØØxØ

Dev 'rupt bit map (halfword), 1 = pending

Status - busy or int pending

Type - Shared subchannels - determined by type of devices

Virtual Device Index Table

VDEVBLOK - all alloc in free storage in a contiguous chunk. Each time a dev is defined all are copied into a larger space. Created by ATT, DEFine, & LINK. The request for free storage is conditional & if not available the request is denied. (ATT fails).

Dev addr = ØØØx

Dev Type

↑VMBLOK

I/O Count

↑ active IOBLOK (after SIOF) (VDEVIOB)

virt CSW (VDEVCSW)

Status - busy, 'rupt pending, not ready, Dedicated

↑ RDEVBLOK if ded, dialed, minidisk

Device Dependent Fields: console, spooled UR, & if minidisk → DASD VDEVRELN cyl relocation, DASD size (VDEVEND),

and link chain pointer if LINKed to somebody's VDEVBLOK. All linked VDEVBLOKs are chained. The real owner's VDEVBLOK may not even exist for the device.

I/O Processing

The virtual machine puts a CAW into his page 0 pointing to the chan pgm in a virtual page:

SEEK

SET SECTOR

SEARCH ID Equal cccc HHHH RR
00B3 0011 07 ← a label not necessarily a count

TIC

READ - will put data into a virtual page or pages.

Then the user issues a SIO or SIOF & does BC. Since these are Privops, CP finally knows what the virtual machine is trying to do. Pgm Chk occurs. DMKFRG → DMKPRIV passes control to DMKVSI. It checks to see if virtual device is valid & if so, tries to turn the request into a real I/O.

DMKVSI gets an IOBLOK, translates virt dev addr to real, changes cyl to point within minidisk (Chan pgm is copied into RCWTASK where corrections occur). SET SECTOR is ok. SEARCH ID Equal is also left alone (except that real addresses are used for all of these). The READ addr is changed & IDAWs used. IDAW bit is turned on. IDAW entries are incremented on halfword boundaries.

The RCWTASK block is a fixed size & if the chan pgm is too long DMKVSI has to go get another larger RCWTASK block and copy everything into it & then continue until it fills again. Also, CCW translation continues until done or an error occurs & nobody else runs. Some products benefit by generating huge channel pgms (FDR + Syncsort)

The IOBLOK IRA points to DMKVIO.

CCWs are not translated for V=R user with SET NOTRANS ON using ded dev, chan or ATT devices.

DIAG 98 does not checking at all.

For TIO or CLRIO, the virtual BLOKs are checked. No I/O takes place (except CTCA). The CC is faked. HIO & HDV is handled similarly unless the virt blks show a real I/O is in progress in which case CP will try to stop the real I/O.

Completion:

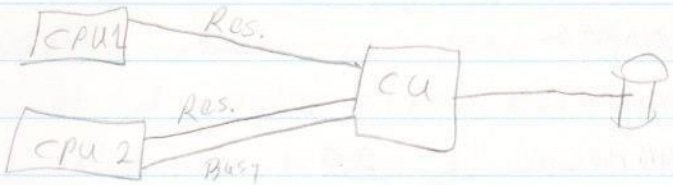
DMKVIO (SLIH) gets control from IOBLOK IRA, which checks CC (for possible deferred CC). If CC=0, store CSW in VDEVBLOK, & untranslate CSW address. If CC=1, post pending Int. in VM I/O Blocks. If IOERBLOK exists, store ↑ in VDEVBLOK, set unit check in CSW.

CC=3 not a path. Mark virt I/O blocks not busy.

DMKCCW is very large & complicated. (This is 0C0 in VM/SP XA).

Reserve/Release Support

Hardware support of this is available on Tape and DASD. The lock is for a device & is held in the str. cnt. & blocks all but one path regardless of the # of CPUs.



CP does not use it, but it checks for hardware availability at init. For ALT paths, CP will not permit a user to issue Reserve. changes it to Sense cc=0

Two MVSs under a single CPU are not protected from each other since both use the same path.

CP provides virtual Reserve/Release support. Based on linking to a minidisk (which may be a full-pack minidisk). In directory entry MDISK statement MWV ALL ALL
↑ indicates virt. Res/Rel.
User may issue Res & CP will set indicator in VRRBLOK (anchored off VDEVBLOK).
↳ Holds
↑ VMBLOK of reserver
Virt. addr of owner of VDEVBLOK
CPEVBLOK anchor for everybody who got a busy & will eventually get a device end.

Diagnose I/O

18 → Standard DASD I/O

20 → general I/O

Designed to run I/O synchronously. Wait until I/O is done & then deal w/ CC. Don't use virtual interrupt mechanism.

Chan pgm must still be built.

Diag 83xx causes pgm chk.

DMKPRG
↓
DMKPRV
↓
DMKSIHVC → 20

18 ←
DGD
↓
IOS
↓
DGD
↓
DSP

20
GIO
↓
CCW
↓
GIO
↓
IOS
↓
GIO
↓
DSP

DMKDGD - designed w/ CMS in mind. byte count must be 800/1024/2048/4096

CP will handle error recovery & user built chan pgm doesn't have to do it. User gets only a reg full of sense (rearranged).

Diag 98 - determine Real store addresses and avoid CCW translation. Subfuctions:

Ø - give adr & lock pg

8 - DMKIOS executes, no translation

4 - unlock page frames

User must have DIAG98 privilege specified in directory entry.

Dispatcher

3-10-88
Gary

Dispatcher does more than dispatch. It also reflects interrupts to virtual machines.

DMKDSPCH - main entry point, requires global system lock. If lock not avail, go to DMKDSPRU - the run user entry which cannot dispatch system work. Not used on UP.

DMKDSPA - fast entry for redispatch after simulation or interrupt handling. If same user can't run, go back to main.

DMKDSPB - checks new PSW if changed outside of dispatcher (like LPSW in virt machine) for validation & exit to main EPA.

DMKDSPWI - enablement window prior to dispatch.

R11 points to VMBLOK at all entries. This is to handle time accounting, reflect int., & validate virtual PSW. The scheduler may be called to modify the status of the virtual machine (it controls placement on the lists).

Dispatch order:

- IOBLOKs (IOBIRA)
- TRQBLOKs (TRQIRA)
- Enable for interrupts
- CPEXBLOKs
- Tiny Enablement window
- User
- Load enabled wait (or active wait)

If work is found at any level, it is processed & then the selection process begins again from the beginning.

Reflect Interrupt

Restart, PMA
Pgm
Ext ⟶
Restart - all others

If user is dispatchable, select highest priority interrupt, verify virtual PSW enabled for interrupts, and exit if not enable. If ok, reflect interrupt by performing virtual PSW swap, validate new psw, & store int code & related data in psw.

Validation checks for PI loops & External Int loops.

If ecMode set on for user, build shadow tables in case an EC mode PSW is loaded. Once new PSW is determined, see if any more interrupts can be reflected (the new PSW may be enabled for more).

XINTBLOKs are anchored in VMBLOK at VMPXINT. Ext crupts are chained in priority order.

I/O crupts are indicated at VMCHINT with a bit map (lowest # chan interrupt is presented first). Depends on CR2 or BC PSW mask.

IOPEND

VMPEND indicates Restart Int pending.

VMRSTAT has all the bits that make a machine eligible to be dropped from the ran list. This is usually idle wait for CMS.

The scheduler (DMKSCHDL) alters dispatching status due to a change in the virt machine.

If Extends exist, only do work to fix → Process only paging I/O Bloks (IBLOKs have real adr in first word. High order bit is off. TRQBLOKs

have the TOD is the first word ← the high order bit is on). Highest dev addr is 1FFF. Process paging CPEXBLOKs (identify by CPEXADD against table of known paging EPAs). NO users get dispatched during EXTENDS.

If no IOBLOK, TRQBLOKs, or interrupts, try CPEXBLOKs. These are CP work that can wait a bit: page-in, error recording, logoff... So, load R15 from CPEXADD + RØ-14 from CPEXBLOK ← then FRET the CPEXBLOK. No trace entry except FRET of CPEXBLOK.

Virtual Machine - check True Run List (TRL). Select first runnable user (TRL Lock & VMBLOK lock). Recreate environment:

CR1 Seg Tab or Shadow Table
μcode assists
PER if needed
FR Regs
GPRs
Create PSW (from old PSW or massaged VMPSW)

This code is complicated.

Insure CP stays in control of the hardware:
Load CRs with CP values.
Load interval timer with Dispatcher Time Slice.
Load CPU Timer with In-queue Time Slice.

CRØ - on all ext int. User can choose page size.
CR1 - seg Tab ↑ or shadow tables created by CP
CR2 - enable all chans
CR6 - set appropriate μcode assists
CR8 - if CPTrap - CP loads

CR9-11 - CP PER or
VM PER
CR14-15 - CPs Mch Chk setting.

Other CRs are not loaded. No DAS.

DMKWAIT(ST) - steal VMBLOKs from other processor. Get TRL lock & look for 2 or more dispatchable VMBLOKs. Move it back to our TRL & release other CPUs TRL Lock. Then exit to dsp & discover something on TRL.
Both TRLs are usually equal in length & little stealing usually occurs.

Dispatcher Timeslice (DMKDSPQS) is the interval timer value. If compute bound (Q3) multiply by 4.

If no users in TRL who are runnable, see if users in runlist. If no, set CP idle wait. If yes, check VMRSTAT of each. If:
the sum of their ws is > 1/2 the DPA
Then, set CP Pagewait!
Else, if one user is in I/O wait,
then set CP IOwait
Else, go back to PAGEWAIT anyway!

Load CP 0300 wait (on UP) or DMKWAITA for active wait. Switch CP PSW to key 3 to indicate active wait.

For HPO running second level under VM/XA in MP mode, a DIAG 44 (XA diag.) is issued to tell XA that HPO would be running active wait and don't run until an Interrupt comes in for HPO.

Gary

Scheduler

Scheduler orders & maintains TRL which the dispatcher uses blindly. Objective: don't overload DPA. The TRL machines are entitled to have pages in storage. Runlist machines are eligible to have pages in storage. Favor interactive users over non-interactive users. Maintains data on individual users, groups of users, + DPA.

Functions:

- Maintain lists
- Calc priority
- init Q-drop processing
- Determine if user is Q1
- Control E to runlist promotion
- Control Q3 users

Lists:

cyclic list - all VMBLOKs: Sch doesn't care.
Eligible list - users waiting for storage. anchored at DMKSCHE1.
Run List - DMKSCHRL - user who have storage allocated. User waiting on CP-simulation, etc.
True Run List - 1/CPU. Ready to run. Anchored in TRLANCHR in PXA & PXB.
TRLCT(+180) is count of VMBLOKs on TRL.

Runlist 2 big slice
Sch 3
Dispatcher - little
↳ DMKDSPQS

VMBLOKs are ordered by deadline on queues. Access is not necessarily round-robin. Queue Manager Units (QMUs) are bits 5-36 of TOD clock and is the deadline. Take TOD now + add average cyle time through eligible + run lists and make that the deadline (subject to coefficients).

Time Slicing

Quantum - determined by processor speed (determined by BC loop) in DMKSTP. This is stored as a word @ DMKDSPQS + is the length of a single stay on the processor. This value is loaded into the interval timer.

LITTLE TS

The dispatcher time slice is how long a machine can stay inqueue:

Q1 - 8 * Quantum (DMKSCHQ1) → VMQBLOK anchors
Q2 - 64 * Quantum (DMKSCHQ2) →

Quantum can be changed w/ SET SRM DSPSLICE

If a user has not entered wait by end of dispatcher ts, mark user compute bound + get a worse position in TRL. If CPU is not scarce, maybe increase timeslice value to cut number of trips through the scheduler.

If Runlist time expires w/o relinquishing control, you get Q2 (non-inter) + are dropped from TRL + run list. Logical swap out is initiated. Priority is recalculated (VMEPRIO) to do next deadline. Storage requirements are also recalculated → a guess at best. (VMPROJWS) Since a user is coming off TRL, see if another can go from E list to TRL.

When user goes idle, drop from TRL + set 300ms timer. At pop, drop from runlist + promote somebody else. For dropped user, recalc deadline priority + storage requirement. If user comes ready before 300ms put back in TRL + let him use runlist time slice (+ get downgraded to Q2).

A user stays Q1 by entering a wait longer than 300ms before the Runlist timeslice expires.

The deadline TOD starts at zero for moment of IPL. This gives 2 yrs of bits.

VMEPRIOR - E list priority

Equals VMTPRIOR except for compute bound machines.

VMEPRIOR = VMQPRIOR + TODCURR - TODIPL

this "normalizes" TOD

VMQPRIOR is the delay factor.

Directory priority value undergoes conversion:

$$\text{Factor} = \frac{64^{\text{Factor}/64}}{64}$$

A lower priority results in better service. See the CP logic Manual for a complete description.

CPU + Paging load factors also count. The individual user perf is * by average perf. If user is better than average then product is fractional + priority becomes a lower number.

If a user comes ready after a long wait, a new deadline is calculated which improves position in E list.

Projected WSS: depends on # pages & CPU time. How much work was done per page read. It favors locality of reference. This is compared to ideal. PWSS is adjusted if in bounds. PWSS < max pages used, and PWSS > 1/2 average # resident pages. Shared pages don't count in PWSS.

When moving from E list to runlist:
- Check 1st Q1 user on E list
  If PWSS fits, move to runlist
- Are enough Q1 users in runlist? If not add anyway.
- Can a Q3 on runlist be pre-empted?
- See in non-inter user can fit.

Criteria are kept in VMSBLOK.
- Min/Max % avail DPA
- Min # users in runlist = 1
- # users in runlist
- Min/Max avail DPA pages
- Sum of PWSSs of runlist users.

Priority for non-interactive:
When user used 6 consegutive Q2 timeslices - Q3 flag is set and will get a terrible deadline but will get 8 Q2 timeslices when it finally runs. The stay on the E list will be 8 times longer.

Commands —

Priority Calc

SET Priority
SET Favored %
SET SRM PB - set high for paging problems.
SET SRM IB - unnoticeable
SET SRM DSPSLICE - may be larger if no CPU bottleneck.
SET QDROP OFF - won't be swapped.
SET Paging %-

E list to Runlist Promotion (pwss calc)

V=R option - Pwss = Ø
LOCK } subtracted off WSS
SET Reserve }
SET SRM MAXWSS - can cause thrashing
SET SRM IBUFF - should be lower than the default 50%
Favored

To Monitor the Scheduler:

Ind Users
Ind Queues } Reports detailed scheduling info. → VMEPRIOR for users.
Ind Position }
Q - anything SET

DMKSCHQ1 - timeslice (DW) - 1st entry in VMQBLOK
DMKSCHQ2 -
Each has # users in runlist *
users in E list (Anchor of Elist is DMKSCHE1)

Only one E list

The VMSBLOK has slightly different categorizations:
- Interactive
- Batch

and the counts will not agree

Gary

VM Assists

These are extensions to the 370 POO and are dependent on processor models. These do not have to be available. VM can still run (except PMA). Assists include:

Virtual Machine Assist
ECPS: VM/370 - includes CP Assist
MVS Extensions - supported for MVS second-level
PMA

Virt Machine Assists: reduces privop simulation. Documented in 1980 manual: GA22-7074

ECPS - smaller processors only (4300s). Some CP code ran as microcode. Does not aid the user (i.e. Privop), helps CP - find freestorage, etc.

Expanded VMA - added more Privops to μcode.

Virt Timer Assist - replaces CP code on updating Virtual Interval Timer. - Not implemented on larger CPUs

MVS Extensions - DAS, LAP, & common segment facility. PC HPO implements with additional μcode, simulation, & reflection of non-availability. Documented in HPO 3.6 System Programmer's Guide.

PMA - avail only to V=R user. Designed to help MVS guest. Permits true supervisor state for MVS, & I/O intrpts handled by MVS through new interfaces between CP & guest. CP overhead is reduced.

VMA

Availability is determined by DMKCPI at init time. Used for all users ⇒ SVC may be reflected directly to guests. Also, improves shadow table maintenance. Default is to be used by all machines but may be turned off. CR6 controls VMA. Points to microcode block.

Flow:

User in virt supervisor & issues privop. Microcode detects before pgm chk & checks CR6 to see if supported. If simulation possible, do it & resume user execution. User is not interrupted, traced, or monitored. CP doesn't even know a privop was issued. User prob state time increases.

CR6:

bits 0-7 → 0 VMA enabled
4 → 1 SVC assist disabled
2 → Extend key operations
3 → ECMode Setting
8-28 → ↑ MICBLOK (in high free) 1/machine using any assists
↑ SEGTABLE
↑ ECBLOK
Flag | ↑ VMPSW
↑ work area
↑ virt interval timer
Flags
29 - 370 EF enabled
30 - PMA enabled
1 - Prob/Super State

Instructions

IPK, SPKA
ISK, SSK, RRB
SSM
LRA
LPSW
STCTL
STNSM, STOSM
SPT, STPT
ISKE, SSKE, RRBE

All SVCs except 76 are directly reflected to the user.

Shadow Table - prevents PIC 11 if the shadow table only need be validated with data from CP tables.

Assists will be disabled if user is tracing, using PER, or has an ADSTOP set. HPO running second level cannot offer assists to its users.

CP Assist

Certain CP functions are now in µcode. Initiated via E6 instruction:

Get/Ret Free Stor
Loc I/O CBs
lock/unlock page
DSP returns
translate CCWs
SV8 & 12 (Link & Ret)

Assists can be individually disabled. If successful, a trace entry is created with X'80' ORed on in first byte to alert that CP Assist did it. If function fails, CP performs through normal code.

At init, if not supported, the E6 instructions are all overlaid with no-ops. The assist may not exactly match the original CP function.

CP Assist is not implemented on the big CPUs because there is no speed gain.

Expanded VMA

Gets control is regular VMA can't do a function.

LPSW	SCKC
STNSM, STOSM, SSM	TCH
SIO (SIOF)	DIAG
SPT, STPT	