Vol. 1

heavy weight paper - perfect for felt tip pens!

50 SHEETS 11x8½ 4x4 QUADRILLE 33-688

NATIONAL

NATIONAL BLANK BOOK COMPANY, INC. • HOLYOKE, MASS. 01040

MADE IN U. S. A.

CHEMICAL SYMBOLS & VALENCE CHART

Brian R. Page

Chemical Elements, Symbols, Atomic Weights and Valence Chart

Atomic weights, based on the exact number 12 as the assigned atomic mass of the principal isotope of carbon, carbon 12, are provided through the courtesy of the International Union of Pure and Applied Chemistry and Butterworth Scientific Publications.

For the radioactive elements with the exception of uranium, thorium and radium, the mass number of either the isotope of longest half-life (marked with a star) or the better known isotope (marked with two stars) is given.

Chemical Element	Symbol	At. No.	Atomic Weight	Valence	Discoverer
Actinium	Ac	89	227*	.....	Debierne
Aluminum	Al	13	26.9815	3	Oersted
Americium	Am	95	243*	3, 4, 5, 6	Seaborg, et al.
Antimony	Sb	51	121.75	3, 5	Valentine
Argon	Ar	18	39.948	0	Rayleigh, Ramsay
Arsenic	As	33	74.9216	3, 5	Magnus
Astatine	At	85	210*	1, 3, 5, 7	Corson, et al.
Barium	Ba	56	137.34	2	Davy
Berkelium	Bk	97	247*	3, 4	Thompson, Ghiorso, Seaborg
Beryllium	Be	4	9.0122	2	Vauquelin
Bismuth	Bi	83	208.980	3, 5	Valentine
Boron	B	5	10.811a	3	Davy
Bromine	Br	35	79.909b	1, 3, 5, 7	Balard
Cadmium	Cd	48	112.40	2	.....
Calcium	Ca	20	40.08	2	Stromeyer
Californium	Cf	98	249*	.....	Davy
Carbon	C	6	12.01115a	2, 4	Thompson, et al.
Cerium	Ce	58	140.12	3, 4	.....
Cesium	Cs	55	132.905	1	Klaproth
Chlorine	Cl	17	35.453b	1, 3, 5, 7	Bunsen, Kirchoff
Chromium	Cr	24	51.996b	2, 3, 6	Scheele
Cobalt	Co	27	58.933	2, 3	Vanuquelin
Copper	Cu	29	63.54	1, 2	Brandt
Curium	Cm	96	247*	3	Seaborg, et al.
Dysprosium	Dy	66	162.50	3	Boisbaudran
Einsteinium	Es	99	254*	.....	Ghiorso, et al.
Erbium	Er	68	167.26	3	Mosander
Europium	Eu	63	151.96	2, 3	Demarcay
Fermium	Fm	100	253**	.....	Ghiorso, et al.
Fluorine	F	9	18.9984	1	Scheele
Francium	Fr	87	223*	1	Perey
Gadolinium	Gd	64	157.25	3	Marignac
Gallium	Ga	31	69.72	2, 3	Boisbaudran
Germanium	Ge	32	72.59	4	Winkler
Gold	Au	79	196.967	1, 3	.....
Hafnium	Hf	72	178.49	4	Coster, Hevesy
Helium	He	2	4.0026	0	Ramsay
Holmium	Ho	67	164.930	3	Cleve
Hydrogen	H	1	1.00797a	1	Cavendish
Indium	In	49	114.82	3	Reich, Richter
Iodine	I	53	126.9044	1, 3, 5, 7	Courtois
Iridium	Ir	77	192.2	3, 4	Tennant
Iron	Fe	26	55.847b	2, 3	.....
Krypton	Kr	36	83.80	0	Ramsay, Travers
Lanthanum	La	57	138.91	3	Mosander
Lawrencium	Lw	103	257*	2, 4	Ghiorso, T. Sikkeland, A. E. Larsh, and R. M. Latimer
Lead	Pb	82	.....	1	.....
Lithium	Li	3	6.939	3	.....
Lutetium	Lu	71	174.97	2	Arfvedson
Magnesium	Mg	12	24.312	2, 3, 4, 6, 7	Welsbach, Urbain
Manganese	Mn	25	54.9380	.....	Liebig, Bussy
Mendelevium	Md	101	256*	1, 2	Gahn
Mercury	Hg	80	200.59	3, 4, 6	Ghiorso, et al.
Molybdenum	Mo	42	95.94	3	Hjelm
Neodymium	Nd	60	144.24	.....	Welsbach
Neon	Ne	10	20.183	0	Ramsay, Travers
Neptunium	Np	93	237*	4, 5, 6	McMillan and Abelson
Nickel	Ni	28	58.71	2, 3	Cronstedt
Niobium (Form. Columbium)	Nb	41	92.906	3, 5	Hatchett
Nitrogen	N	7	14.0067	3, 5	Rutherford
Nobelium (c)	No	102	256*	.....	Ghiorso, et al.
Osmium	Os	76	190.2	2, 3, 4, 8	Tennant
Oxygen	O	8	15.9994a	2	Priestly, Scheele
Palladium	Pd	46	106.4	2, 4, 6	Wollaston
Phosphorus	P	15	30.9738	3, 5	Brandt
Platinum	Pt	78	195.09	2, 4	Ulloa
Plutonium	Pu	94	242*	3, 4, 5, 6	Seaborg, et al.
Polonium	Po	84	210**	.....	P. and M. Curie
Potassium	K	19	39.102	1	Davy
Praseodymium	Pr	59	140.907	3	Welsbach
Promethium	Pm	61	147**	3	Glendenin and Marinsky
Protactinium	Pa	91	231*	.....	Hahn and Meitner
Radium	Ra	88	226*	2	P. & M. Curie, Bemont
Radon	Rn	86	222*	0	Dorn
Rhenium	Re	75	186.2	.....	Noddack and Tacke
Rhodium	Rh	45	102.905	3	Wollaston
Rubidium	Rb	37	85.47	1	Bunsen, Kirchoff
Ruthenium	Ru	44	101.07	3, 4, 6, 8	Claus
Samarium	Sm	62	150.35	2, 3	Boisbaudran
Scandium	Sc	21	44.956	3	Nilson
Selenium	Se	34	78.96	2, 4, 6	Berzelius
Silicon	Si	14	28.086a	4	Berzelius
Silver	Ag	47	107.870b	1	.....
Sodium	Na	11	22.9898	1	Davy
Strontium	Sr	38	87.62	2	Crawford
Sulfur	S	16	32.064a	2, 4, 6	.....
Tantalum	Ta	73	180.948	5	Eckeberg
Technetium	Tc	43	99**	6, 7	Perrier and Segre
Tellurium	Te	52	127.60	2, 4, 6	Von Reichenstein
Terbium	Tb	65	158.924	3	Mosander
Thallium	Tl	81	204.37	1, 3	Crokes
Thorium	Th	90	232.038	4	Berzelius
Thulium	Tm	69	168.934	3	Cleve
Tin	Sn	50	118.69	2, 4	Gregor
Titanium	Ti	22	47.90	3, 4	d'Elhuiar
Tungsten (Alternate Wolfram)	W	74	183.85	6	Klaproth
Uranium	U	92	238.03	4, 6	Sefstrom
Vanadium	V	23	50.942	3, 5	Ramsay, Travers
Xenon	Xe	54	131.30	0	Marignac
Ytterbium	Yb	70	173.04	2,3	Gadolin
Yttrium	Y	39	88.905	3	.....
Zinc	Zn	30	65.37	2	.....
Zirconium	Zr	40	91.22	4	Klaproth

[a]Atomic weights so designated are known to be variable because of natural variations in isotopic composition. The observed ranges are: hydrogen ± 0.00001; boron ± 0.003; carbon ± 0.00005; oxygen ± 0.0001; silicon ± 0.001; sulfur ± 0.003.

[b]Atomic weights so designated are believed to have the following experimental uncertainties: chlorine ± 0.001; chromium ± 0.001; iron ± 0.003; bromine ± 0.002; silver ± 0.003.

[c]The name for element 102 (Nobelium) is still in doubt because its discovery is in dispute.

PERIODIC CHART

1a	2a	3b	4b	5b	6b	7b	8	1b	2b	3a	4a	5a	6a	7a	0	Orbit
1 +1 H −1 1.00797 1															2 0 He 4.0026 2	K
3 +1 Li 6.939 2-1	4 +2 Be 9.0122 2-2									5 +3 B 10.811 2-3	6 +2 +4 −4 C 12.01115 2-4	7 +1 +2 +3 +4 +5 −1 N 14.0067 2-5 −2 −3	8 −2 O 15.9994 2-6	9 −1 F 18.9984 2-7	10 0 Ne 20.183 2-8	K-L
11 +1 Na 22.9898 2-8-1	12 +2 Mg 24.312 2-8-2									13 +3 Al 26.9815 2-8-3	14 +2 +4 −4 Si 28.086 2-8-4	15 +3 +5 −3 P 30.9738 2-8-5	16 +4 +6 −2 S 32.064 2-8-6	17 +1 +5 +7 −1 Cl 35.453 2-8-7	18 0 Ar 39.948 2-8-8	K-L-M
19 +1 K 39.102 -8-8-1	20 +2 Ca 40.08 -8-8-2	21 +3 Sc 44.956 -8-9-2	22 +2 +3 +4 Ti 47.90 -8-10-2	23 +2 +3 +4 +5 V 50.942 -8-11-2	24 +2 +3 +6 Cr 51.996 -8-13-1	25 +2 +3 +4 +6 +7 Mn 54.9380 -8-13-2	26 +2 +3 Fe 55.847 -8-14-2	27 +2 +3 Co 58.9332 -8-15-2	28 +2 +3 Ni 58.71 -8-16-2	29 +1 +2 Cu 63.54 -8-18-1	30 +2 Zn 65.37 -8-18-2	31 +3 Ga 69.72 -8-18-3	32 +2 +4 Ge 72.59 -8-18-4	33 +3 +5 −3 As 74.9216 -8-18-5	34 +4 +6 −2 Se 78.96 -8-18-6	35 +1 +5 −1 Br 79.909 -8-18-7	36 0 Kr 83.80 -8-18-8	-L-M-N
37 +1 Rb 85.47 -18-8-1	38 +2 Sr 87.62 -18-8-2	39 +3 Y 88.905 -18-9-2	40 +4 Zr 91.22 -18-10-2	41 +3 +5 Nb 92.906 -18-12-1	42 +6 Mo 95.94 -18-13-1	43 +4 +6 +7 Tc (99) -18-13-2	44 +3 Ru 101.07 -18-15-1	45 +3 Rh 102.905 -18-16-1	46 +2 +4 Pd 106.4 -18-18-0	47 +1 Ag 107.870 -18-18-1	48 +2 Cd 112.40 -18-18-2	49 +3 In 114.82 -18-18-3	50 +2 +4 Sn 118.69 -18-18-4	51 +3 +5 −3 Sb 121.75 -18-18-5	52 +4 +6 −2 Te 127.60 -18-18-6	53 +1 +5 +7 −1 I 126.9044 -18-18-7	54 0 Xe 131.30 -18-18-8	-M-N-O
55 +1 Cs 132.905 -18-8-1	56 +2 Ba 137.34 -18-8-2	57*+3 La 138.91 -18-9-2	72 +4 Hf 178.49 -32-10-2	73 +5 Ta 180.948 -32-11-2	74 +6 W 183.85 -32-12-2	75 +4 +6 +7 Re 186.2 -32-13-2	76 +3 +4 Os 190.2 -32-14-2	77 +3 +4 Ir 192.2 -32-15-2	78 +2 +4 Pt 195.09 -32-16-2	79 +1 +3 Au 196.967 -32-18-1	80 +1 +2 Hg 200.59 -32-18-1	81 +1 +3 Tl 204.37 -32-18-3	82 +2 +4 Pb 207.19 -32-18-4	83 +3 +5 Bi 208.980 -32-18-5	84 +2 +4 Po (210) -32-18-6	85 At (210) -32-18-7	86 0 Rn (222) -32-18-8	-N-O-P
87 +1 Fr (223) -18-8-1	88 +2 Ra (226) -18-8-2	89** Ac +3 (227) -18-9-2														-O-P-Q

Atomic Number → 50 +2 ← Oxidation States
Symbol → Sn +4 KEY TO CHART
Atomic Weight → 118.69 -18-18-4 ← Electron Configuration

Transition Elements — Group 8 — Transition Elements

*Lanthanides	58 +3 Ce +4 140.12 -19-9-2	59 +3 PR 140.907 -20-9-2	60 +3 Nd 144.24 -22-8-2	61 +3 Pm (145) -23-8-2	62 +2 Sm +3 150.35 -24-8-2	63 +2 Eu +3 151.96 -25-8-2	64 +3 Gd 157.25 -25-9-2	65 +3 Tb 158.924 -26-9-2	66 +3 Dy 162.50 -28-8-2	67 +3 Ho 164.930 -29-8-2	68 +3 Er 167.26 -30-8-2	69 +3 Tm 168.934 -31-8-2	70 +2 +3 Yb 173.04 -32-8-2	71 +3 Lu 174.97 -32-9-2		-N-O-P
**Actinides	90 +4 Th 232.038 -19-9-2	91 +5 Pa +4 (231) -20-9-2	92 +3 +4 +5 +6 U 238.03 -21-9-2	93 +3 +4 +5 +6 Np (237) -22-9-2	94 +3 +4 +5 +6 Pu (242) -23-9-2	95 +3 +4 +5 +6 Am (243) -24-9-2	96 +3 Cm (247) -25-9-2	97 +3 +4 Bk (249) -26-9-2	98 +3 Cf (251) -28-8-2	99 Es (254) -29-8-2	100 Fm (252) -30-8-2	101 Md (256) -31-8-2	102 (254) -32-8-2	103 Lw		-O-P-Q

Numbers in parentheses are mass numbers of most stable isotope of that element.

6-29-83
Don Stein

Gene Amdahl left IBM and between 1971 and 1974 designed the 470-6. Just prior to production, IBM announced 5370 with virtual storage. The 470-6 production was cancelled and the machine redesigned into the 470-V6. The first 470-V6 was shipped in 1975. The machines were created in this order:

470-V6
470-V7
470-V5
-V7A
-V7B
-V7C
-V8

Performance Order:

V5 } comparable 2-3 mips.
V7C } - this one is upgradable.
V6 } comparable to 370/168
V7B } - upgradeable
V7
V7A - Has an accellerator switch; customer is charged more in high gear.
V8 - Comparable to 303X machines.

Amdahl goals are:
1. Quick installation
2. Field upgradable

The 580 series was announced in 1981. First unit (5880) was shipped in Aug. 1982. The 580 series machines are microcode driven. This is an innoration over the 470 series. The 580 machines are comparable to the 308X IBM series.

The 580 family:
5840
5850
5860
5870
5880

Maximum storage on IBM CPU is 48 meg. The 580 will handle 64 meg.

4705

The 4705 is a front-end communications processor. It is twice as fast as the 3705. The 4705E competes with the 3725.

The 4705 is a control unit which is attached to a channel. 16 CUs can go on one channel. 16 devices can go on a CU.

256 devices, or 16 other control units may be attached to a 4705.

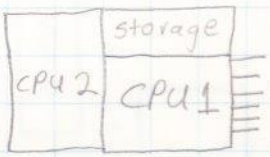
DASD

2314 - 350 generation disk
3330 - Removable disk (still in production)
3350 - permanent HDA (competition for Amdahl 6280)
3340, 3370, 3375 } 43ØØ series machines; ½ to 3 or 4 MIPS
3380 - latest IBM

Configurations

Every channel has two cables, tag (control) and bus (data transfer). Max length is 200 ft. Control units on a channel are daisy-chained. All CUs "hear" every request on a channel. Only the appropriate CU responds to a query.

UP - Uniprocessor
AP - Attached Processor (2 CPUs sharing channels + storage)



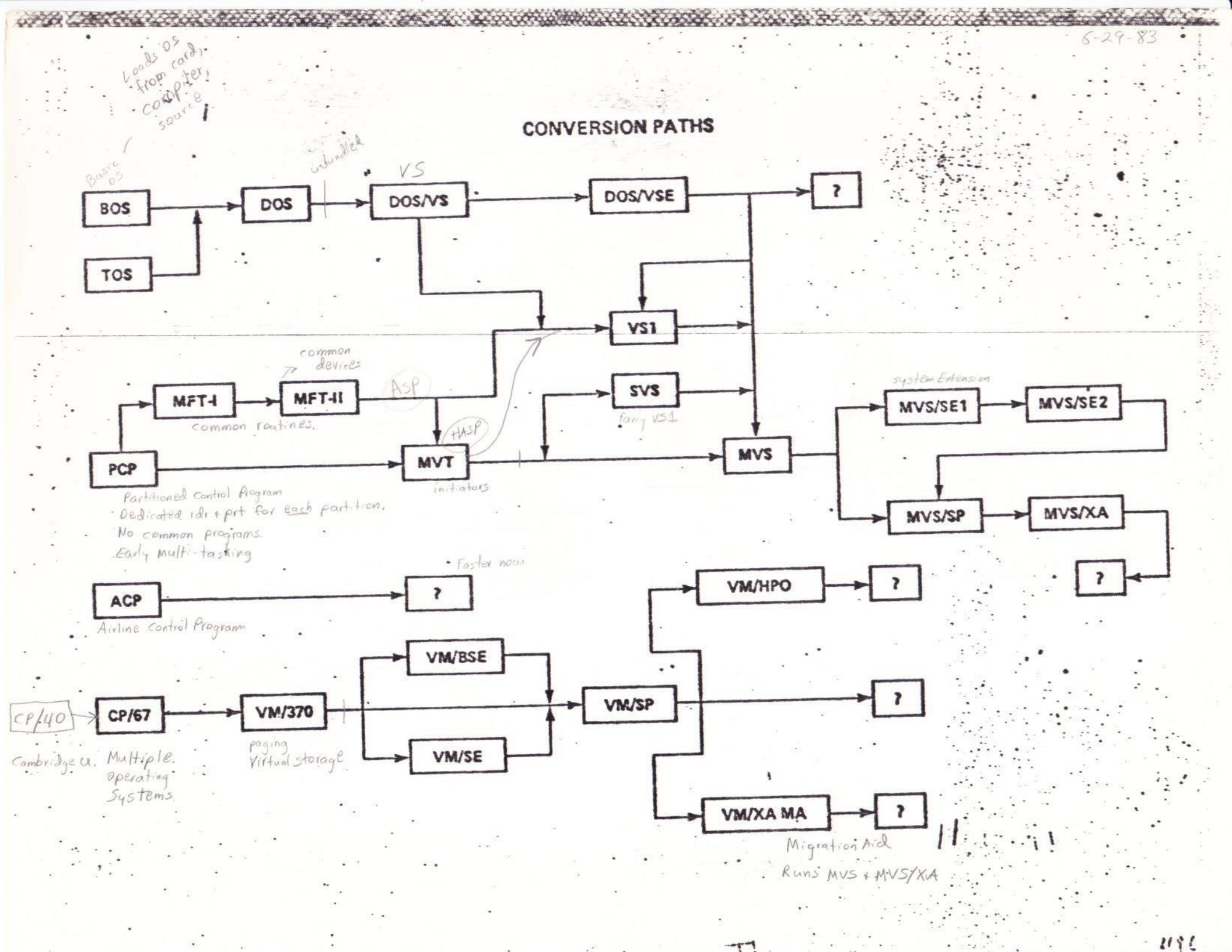
MP - Multi-Processor - two CPUs, each with storage and channels, but interfaced through something like CTCA. May be "loosely coupled" or "tightly coupled."

DP - diadic Processor (IBM 308X series): dual CPUs in one box, extremely tightly coupled, sharing storage and channels.

XA - Extended Architecture: Removes channels from direct CPU control. The SIO instruction is replaced with the SCH (sub-channel) instruction. The channels are "smart." The 3081 will run in either 370 or XA mode. The 580 will soon be able to run both modes simultaneously. Also, XA can be put on a 470 (the upgrade changes the machine from hard-wire control to microcode).

6-29-83

CONVERSION PATHS



Control units can have several addresses. This permits up to 32 devices on a control unit. On Amdahl DASD, each device has two paths (two controllers in each head of string).

Protocol

ASCII - byte transfer (start/stop), usually 300-1200 baud. (slow)
BSC - binary synchronous communications. Permits interleaving on multipoint line.
SDLC/SNA - advanced communications discipline.
1200, 2400, 4800, 9600, + 19,200 baud.
Channel speed is approximately 1.5 meg/second.

Operating Systems (see handout)

MVS

MVS (12,000,000 lines of code) communicates with problem programs through SVCs. MVS controls paging, I/O, storage allocation, and scheduling. JES controls spooling for MVS. BTAM, TCAM, or VTAM control communication access to MVS. JES can spool through VTAM to a 4705 to other CPUs. TSO handles timesharing through TAM.
Other components: RJE + VSAM.

DOS/VSE

Amdahl is not involved much in DOS since DOS is usually an operating system for smaller machines. Power handles spooling for DOS. TCAM and VTAM can run under DOS.

VM (2,000,000 lines)
VM was designed to support interactive processing.
CMS is the timesharing component.
RSCS handles external spooling (through 4705s or other machines).
VM can support a Batch facility.
PVM (pass-thru) allows CMS access to other CPUs.

ACP

Airline Control Program: does not have subsystems. It does have utilities. Designed to be fault tolerant.

Amdahl Software

MVS/SPA - SP Assist. The 3081 hardware has a few more instructions than the 470. Assist translates these instructions into 470-executable code. This is really a user-mod to the error recovery routines.

MVS/ECS - Extended Channel Support: in conjunction with a hardware mod will increase a 470 from 16 channels max to 32 channels.

VM/ECS - same.

VM/SA - improves privileged instruction simulation.

VM/PE - performance Enhancement. A special supervisor coordinates between VM and a subsystem (MVS). PE allows each to handle its own I/O. PE has three page zeros (containing PSWs), one for VM, one for MVS, and one for itself. PE increases the RBT (Relative Batch Throughput) of MVS over a real MVS under VM.

Types of Interrupts:

I/O
EXT - clock, External Interrupt button.
Machine Check
Program - Invalid Op code
SVC - Supervisor call
Restart - Recover from crash
Page 0 (the first 4K of core) holds all necessary PSWs).

VTOC is linked list of DSCBs. Different type DSCBs.
DSCB4 - point to VTOC. DSCB1s are for data sets.
DSCB3 is for secondary extents.
DSCB contains CCHH for data sets.

See "Information Structures" handout for notes.

Topics:
- Lists - Sequential
  - Linked
- Stacks - LIFO
- Queue - FIFO
- Deque - I/O from either end.

- Linked Nodes - trees
- Multi-Linked structures
- File Systems:
  - VTOC
  - Indexed VTOC
  - PDS
  - CMS Files
- Control Blocks
  - DSECT
- Access Methods
  - QSAM - Queued (in buffers) } All used with sequential files.
  - BSAM - Basic }
  - ISAM - Indexed on key }
  - BPAM - used to identify PDS member; access through QSAM.
  - BDAM - specify CCHHRR for direct retrieval.
  - VSAM - panacea: ESDS, KSDS, RRDS

DASD

Each track has a home address and record zero.

HA = F CCHH
02 0194 0001 ← address of track
↑ Flag byte showing track condition.

Record zero contains the count area.

Find out about IIS courses available.

Branches between subsystems in MVS are handled with SVCs. Each subsystem has its own set of SVCs.

IOS (I/O system) within MVS is one of the larger MVS programs, about 8K. This is heavily used.
RMS handles recovery.
FREE handles storage allocation. (Free Storage Manager)
A SIO requires:
Buffers to hold data.
CCW to instruct I/O device.

The PAGE program keeps track of where every page is. He talks to IOS to do the moving.
The Master Scheduler allocates CPU time to each task.

All subsystems are standard assembler programs.

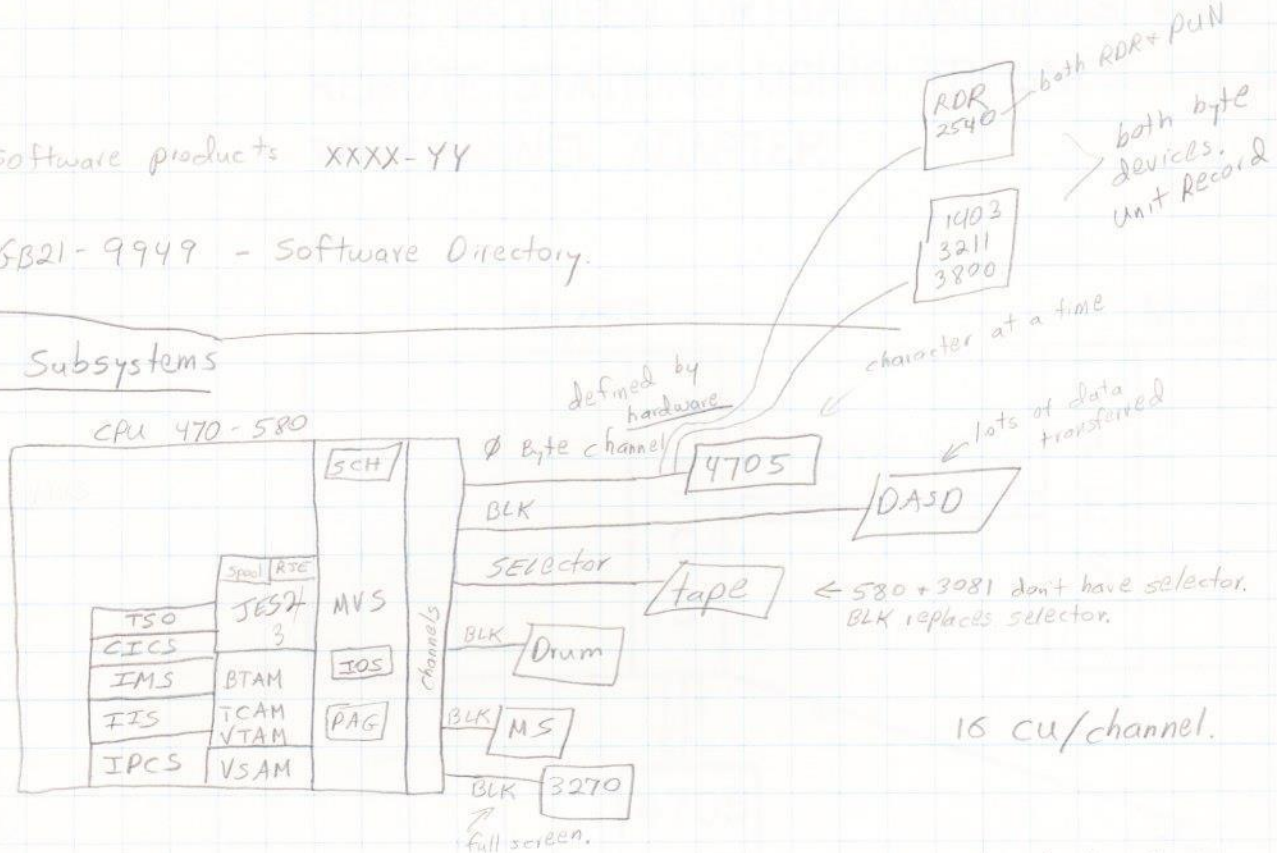
6-30-83
Don Stern

SLSS - System Library Subscription service → Provides Manual updates.
TNL = the amendment.
Document numbers : XXYY-YYYY-Z
↳ level of manual
↳ LY = logic manual (expensive <$100)

Software products XXXX-YY

GB21-9949 - Software Directory.

MVS Subsystems



16 CU/channel.

JES handles spooling with virtual assignments. In a multi CPU environment a JES (3) on a master will schedule all jobs on all CPUs. The master role is passed from one JES to another for reliability. If a CPU goes down, jobs for that machine are queued by the others. JES handles resource allocation.

The TAM talks to terminals on one side and application programs on the other through MVS.

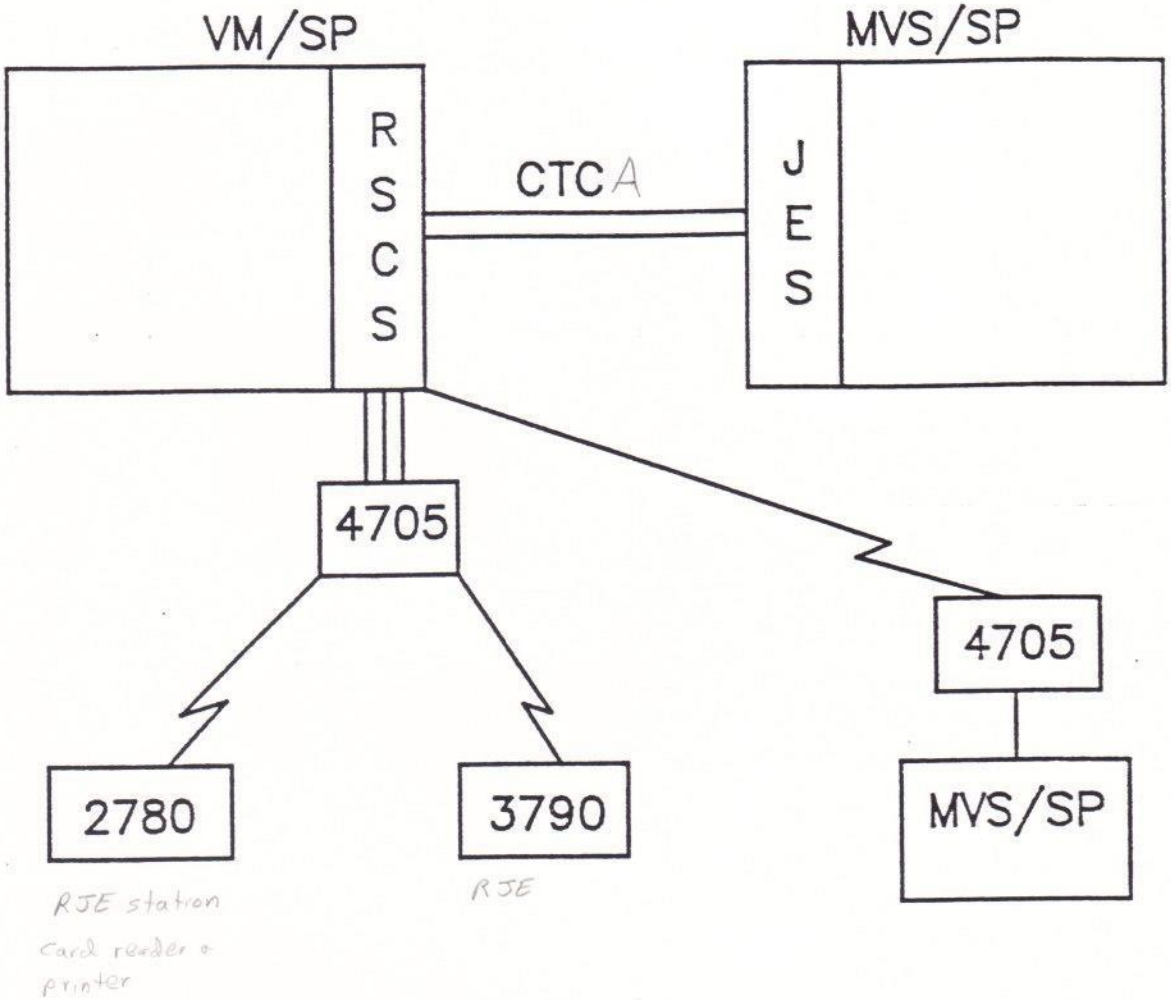
Drums - smaller than disks, faster access. IBM 2305 Fixed heads STC 4305 solid state drum. Drums are good for paging.

seek delay - arm movement
Rotational delay - disk turning

Mass Storage device - transfers spools of tape to disk mechanically.

RSCS

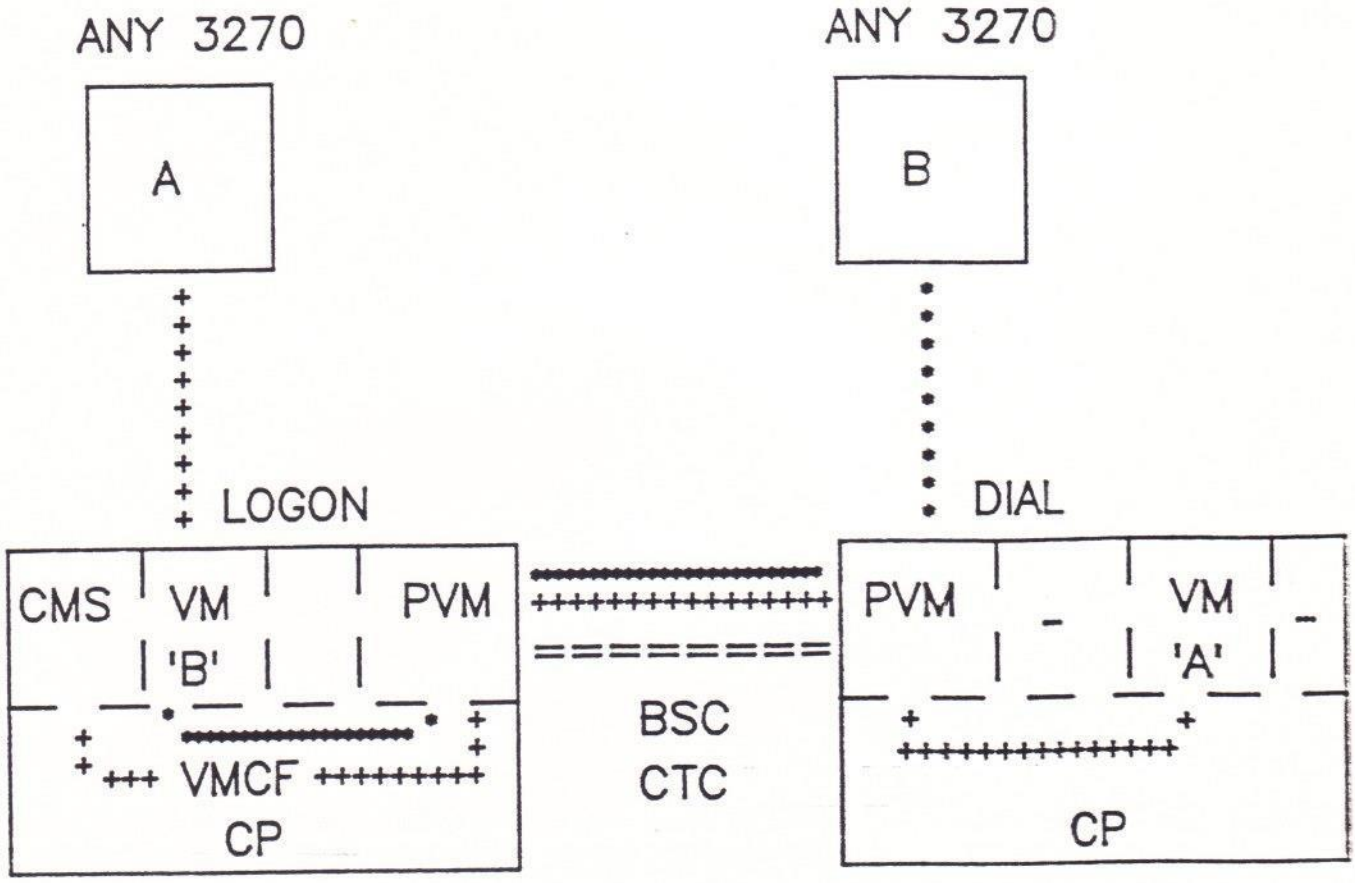
TELECOMMUNICATION SUPPORT TO TRANSFER FILES BETWEEN VIRTUAL MACHINES AND REMOTE STATIONS USING TP LINES OR CHANNEL TO CHANNEL ADAPTER



Some RJE stations can have a disk on which to store + receive data.

6-30-83

VM PASS THRU



Tube 'A' to VM 'A'.
Tube 'B' to VM 'B'.

6-30.83

DASD

track - 1 revolution on one platter.
cyl - 1 revolution on all platters.

model	cyl	
2314 -	200	
3330 -	404	100m bytes
3330-11	—808	200m bytes
3350	555	300m bytes
3380		

Large physical blocks conserve space, but take longer on channel + require more buffer space. Optimum blocking is the goal.

Program steps

1. Source
2. Compiler
3. Object deck (machine code minus external references)
4. Link edit - resolves external references.
5. Load module.
6. Execute

MVS is a 4meg load module.

IPL process

IPL button creates CCW + PSW which goes to a device and retrieves cyl Ø, trk Ø which bootstraps. Bootstrap program loads initialization program which loads the operating system. The initialization routine (loader) checks for end of storage and machine features and every device, and then loads nucleus and transfers the information he has gathered. The OS then establish all his control blocks and locates his libraries.

How to find Info:

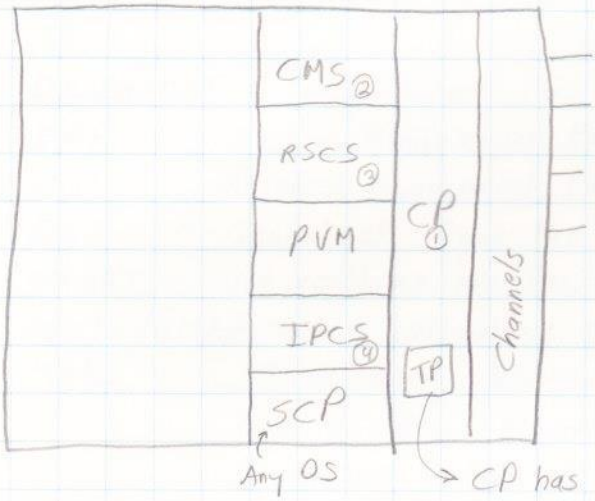
Software directory
General Info Manual
Installation Guide
Operation Manual(s)
Logic Manual (Cross Ref. in back)
Control Blocks + Data Areas
System Programmers Guide
User Guide
Messages and Codes
Performance + Planning
IPO Manual

Languages

Assembler	SAS	COGO
Fortran	Speak-Easy	RPG
Cobol	SPSS	Mark 4
Pascal	GPSS	'C'
APL	EXEC	ADA
PL1	BASIC	
SNOBOL	LISP	

A compiler creates an object module from the entire source deck.
A translator processes one statement at a time as necessary.

VM



CMS requires CP for all I/O. The DIAGnos instruction does a lot of SVC-type functions. Branches are also used.

RSCS is a spooling system for RJE. CP handles its own virtual spooling. RSCS can go through 4705 or CTCA to go to another CPU. (see handout)

PVM has logical lines from another computer, via 4705. PVM emulates a local 3270 for another OS. (see handout)

IPCS - Interactive Problem Control System. IPCS handles reading and interpreting system dumps. Extremely useful. Can be used to help with any dumps.

When installing another OS or DOS under VM, the system programmer must specify a gen parameter to correctly configure the paging & I/O systems.

VM/CMS is a better interactive system than TSO. TSO has about a 10X overhead in instructions than CMS. More terminals, faster.

IPO Installation Productivity Option

Amdahl came out with AMS first.
IPF is a subsystem of IPO. (Interactive Productivity Facility)
IPF asks option questions, assemblies correct modules, & loads nucleus.

Infosys - online problem solution: "Fix" database. Search by keyword. (IBM)
PTF
QWIK is the Amdahl equivalent (menu driven) - tracks hardware, & software problems, education, & parts.

Application Programs

SCRIPT - word processing (CMS, TSO, ICCF)
SPF -
DASDR-DSF - DeviceSupport Facilities (initialize & analyze disks)
SMF - System Management Facility (accounting)
RMF - Resource Management Facility
GTF - Generalized Trace Facility (trace produced for any logical or physical line).
SMART - VM realtime monitor.
Monitor/VMAP - VM performance collection.
TPNS - Teleprocessing Network Simulator (Measurement & Test)
Runs on one CPU under MVS connected through 4705 to another CPU. TPNS will act like a lot of terminals, each issuing different quantities of transactions. Used in benchmarking.

The PSA is a good place to begin debugging.

7-1-83
Don Stern

SPF

SPF is screen driven. The prefix line is on the left (for line + block commands).
(See SPF General Info Manual)

PSA - Prefix Storage Area = Page Ø, containing PSW, Channel Address Words (CAW), and pointers to all channel control blocks. The control blocks are linked and start in PSA. The Task Control Block (TCB) pointers begin in PSA. PSA points to active I/O control block. The trace table pointer + current entry pointer reside in the PSA.
(see printout)

Registers

R Ø, R1 } Parameter + Condition code to be passed (may be address).
R2 ↓ R11 } Working Registers
R12 → Base (Page boundary address) - beginning of routine.
R13 → Savearea (pointer) (Register contents of calling program)
Calling program sets up storage for save area. Called program actually saves registers using R13. Once registers are saved, the contents of R13 are moved to another register which can then be passed to further modules. Looking back, the called program restores the registers of the calling program and the loads his own savearea address into the savearea of the calling program. The original calling program can thus see what happen in the called program. - the location of the called program's save area. The calling program must define a storage location for the called program's save area address.

R14 - Return address
R15 - Goto address

Using multiple bases:

```
USING  *, 12, 9, 8
BALR   *, R12          ← establishes contents of R12
LA     R9, R12+4096    ← establishes R9
LA     R8, R9+4096     ← establishes R8
 .
 .
 .
DROP   ← drops a base
```

```
LOC      OBJECT CODE      ADDR1    ADDR2
000048   90ØB D01Ø        00010
```

(see p. 217 Assem. book)

Displaced Address of Instruction

90 = OP code
0 = Register
B = displacement register
01Ø = displacement

displacement from base in use. ← is this correct?

(see printouts)

CSECT - control section (program)
DSECT - data (constants defined

Timers

TOD - time-of-day → set by STCK command + button.
CC - clock comparator (counts down) based on TOD
location 8Ø - negative counter, interval → is effected by STOP store in decimal location 8Ø.

Scott 7-1-83

JCL

A "/*YYYYY" is an instruction to JES2. - to route print, etc.

JES3 = //*YYYY

Note the non-blank character after *.

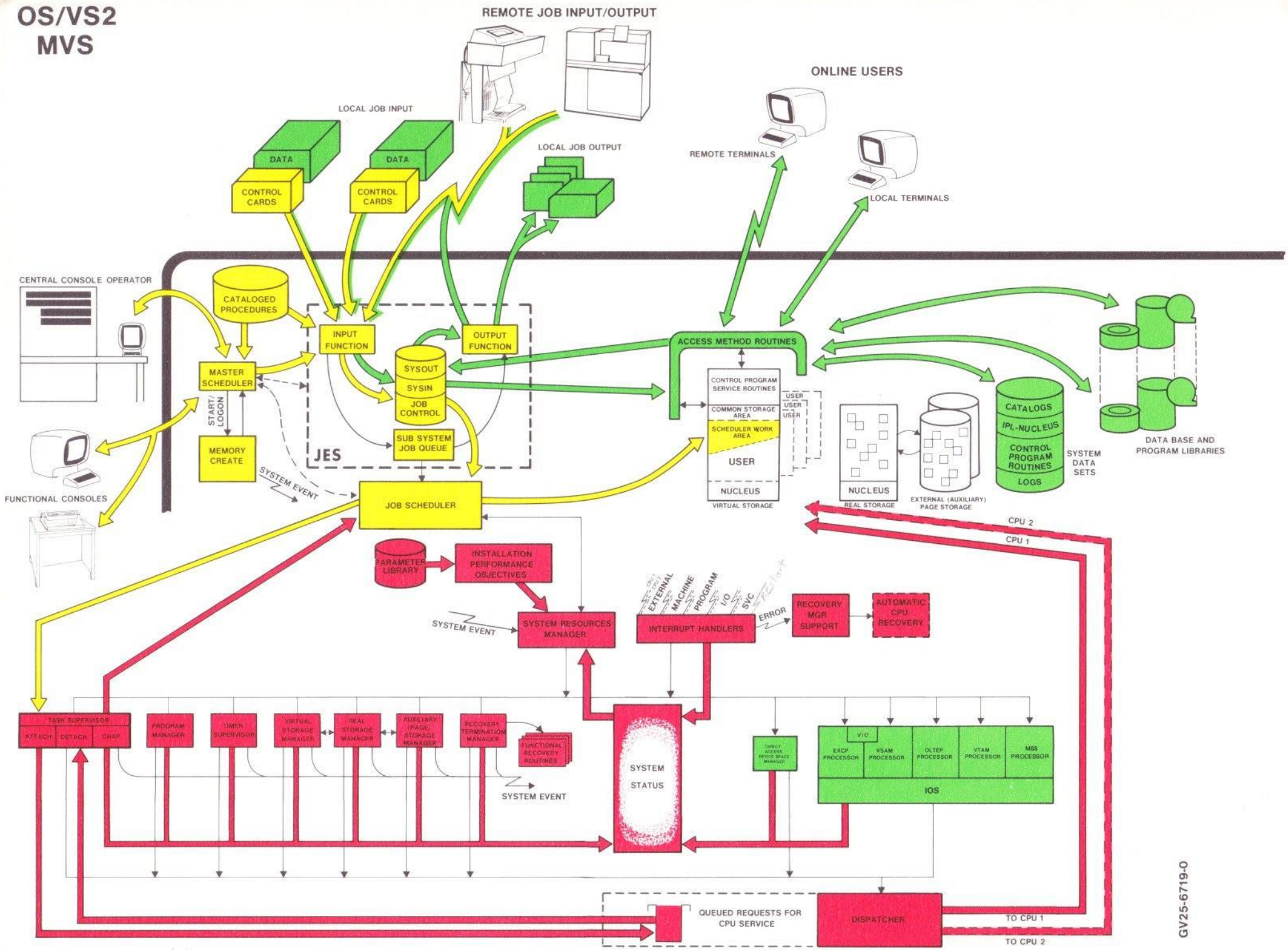
Jobcard:

//_____ JOB (5004xxx, 011C)

msgclass Y holds output on spool - must be used with NOTIFY statement on jobcard.

Amdahl ccs has a //JOBPROC statement.

OS/VS2 MVS



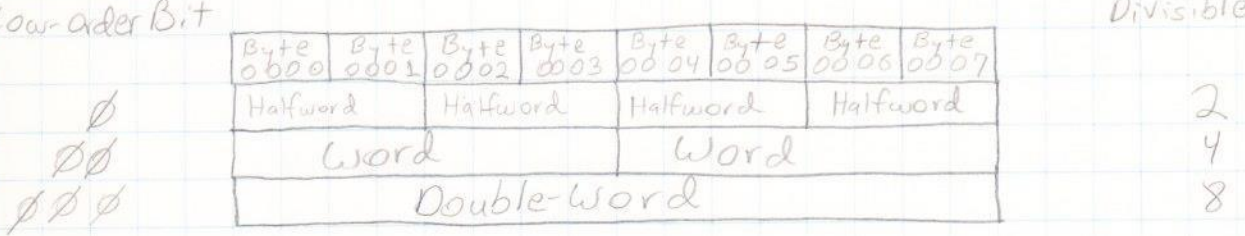
GV25-6719-0

7-5-83

Hardware Prerequisite

Question:

p.12 "The address of fixed length data fields must be divisible by the number of bytes in the field or a specification exception will occur."



"Fixed length data is addressed by the high-order byte (leftmost byte) of the field."

p. 37 "If bit position Ø contains a Ø bit, it indicates that the halfword is the true form of a number and represents a positive operand."

— The high-order bit is position Ø?

Review binary instructions in Burian.

- alternate bases
- binary mathematics

Why would anyone ever use a non-zero START base?

p. 79 Elaborate on halfword arithemetic, especially negative quantities.

What is the principle behind complementing?

Review binary division. p. 99

4-5 lunch

MVS Concepts and Facilities

7-6-83

Bob Uhrina (Chicago)

MVS is the most complicated operating system ever developed.
12,000,000 (w/o TSO)
An operational system (with subsystems) is probably about 25,000,000 lines.
MVS has a lot of exception processing which inflates the number of routines.
(150 instructions in 1964; apx 200 today - different kinds of instructions)

Hardware

3 major components:

CPU - logic & execution
I/O - data transfer
Storage -

CPU - 3 kinds of instructions
- privilged (supervisor state)
- problem state
- Semi-privilged (new state with SP 1.3) - Permits direct Inter-region communication. This involved a hardware change as well as software.

In worst case, MVS can consume 90% of CPU cycles.

PSW - Program Status word (introduced with S360)
- points to the executing instruction. PSW is necessary for multiprograming.
- Storage protection key - isolates programs.

Interrupts - IBM systems are interrupt driven
(Some machines use simple polling; one task is allotted time in uninterruptable sections).
Six kind of interrupts:
1. I/O - information transfer
2. Program - program error.
3. External - operator communication
4. Supervisor Call - program request for privilged (139) services from OS.
5. Machine Check - hardware detected error.
6. Restart - causes IPL.

Memory - consists of data and instructions.

Addressing techniques:

24 bit - addresses 16meg

26 bit - new with SP, 64 meg. (software change)

31 bit - new with XA, 2gig (hardware + software)

storage Protect keys -

old machines had 16 keys, thus limiting capacity to 16 programs.

Channels - a channel is a CPU which executes only "channel programs."

A channel can operate as fast as the CPU. This minimizes waste of CPU time.

Channel Programs use Channel Command Words as instructions.

The Channel Address Word is like a PSW; it points to the instruction.

The Channel Status Word permits communication from the channel to the CPU.

A SIO causes an interrupt on the channel. The channel uses the I/O interrupt to report back to the CPU.

Operating Systems - a group of programs which control system resources and application programs for maximum productivity.

→ CPU, Memory, + I/O devices.

Operating systems reduce the need for duplicate code.

Sysgen - System Generation

stage 1 - select software and hardware options.

stage 2 - Assemblies, link edits, + copies to bring in correct modules in accord with the stage 1 options.

MVS is a parameter driven system. MVS has defaults which are specified at Sysgen and some of which may be altered at the console.

The operating system programs reside on the SYSRES DASD volume.

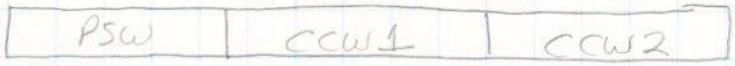
SMP directs maintenance.

A Release usually has major new enhancements + corrected errors.

IPL - Initial Program Load

An IPL brings in the operating system or a standalone dump utility (plus a few other utilities). An IPL is a combined hardware and software function.

The "IPL text" (cyl Ø, Trk Ø) contains a 24 byte field:



The console setting determines the appropriate volume. The hardware locates the IPL volume and loads the 24 byte field.

The CCW1 points to the second record on the IPL volume, the boot strap program. This is a seq read record channel program. This next program also contains CCWs which pull in the rest of the operating system. The first of these modules is the IPL module.

CCW2 causes a branch to the instructions contained in the bootstrap program. The first CCW brings in the IPL program. Only when the IPL program begins executing does the CPU startup. The entire load process is performed by the channel! To begin operating, the CPU first looks to the PSW at low core which points to the IPL program. The IPL program causes System Initialization.

IPL Program

- clears storage + registers
- tests for quantity of main storage
- selects nucleus program
  - ↳ device control blocks
- Relocates IPL Program to high storage.
- Loads Nucleus Initialization Program to low core (NIP).

A Control Block is a portion of main storage in which status and control information is kept.

NIP verifies TOD clock.
checks for DAT/DAS/model Number.
↳ Dual Address Space (SP1.3 or above)
Opens system datasets. (SMF, CPALIB, LINKLIB, etc.)
Initializes Control blocks.

Operating System Components: (see chart GV25-6719-0)

- Job Management - schedules jobs, initiates jobs, and communicates with operator. (yellow on chart)
- Data Management - handles I/O (green)
- Task Management - Allocates CPU & memory resources (Red). the System Resource Manager (SRM) is the heart of task management.

Job Control Language permits the user to communicate with the operating system. (also system console).

History

OS/360 - announced in 1964; designed to run on all 360 machines. Lasted only a few months.
PCP - dead end; 1967. (16K)
MFT - Multi-programing with Fixed Number of Tasks. 1968
MVT - Multi-programming with Variable Number of Tasks. 1969; 128K
BOS - Basic O.S. (1966)
TOS - Tape O.S.
DOS - Disk O.S (1968)
VS1
VS2/SVS
MVS
MVS/SE (requires 2meg)
MVS/SP
MVS/XA

Only about 30% of all of the MVS options are actually used. Designed to please everyone.

MFT Storage Utilization

The goal was to place as many OS routines as possible into main storage. Most routines in the operating systems would be dramatically slowed if they had to be read in each time. Storage had to be contiguous. Thus, storage fragmentation was a potential problem.

A maximum of 16 programs could execute simultaneously. The problem: all programs could request I/O and still leave the CPU idle. This caused as much as 90% idle time.

In a typical COBOL application, only about 20% of a program is actually being executed. The other 80% of the code is set-up, wrap-up, and exception routines. This finding suggested that the previously invented Virtual Storage concept could be employed.

Programs are sliced into 4K pages, some of which reside in storage and some on disk. By having small pages, each of uniform size, the fragmentation problem can be solved.

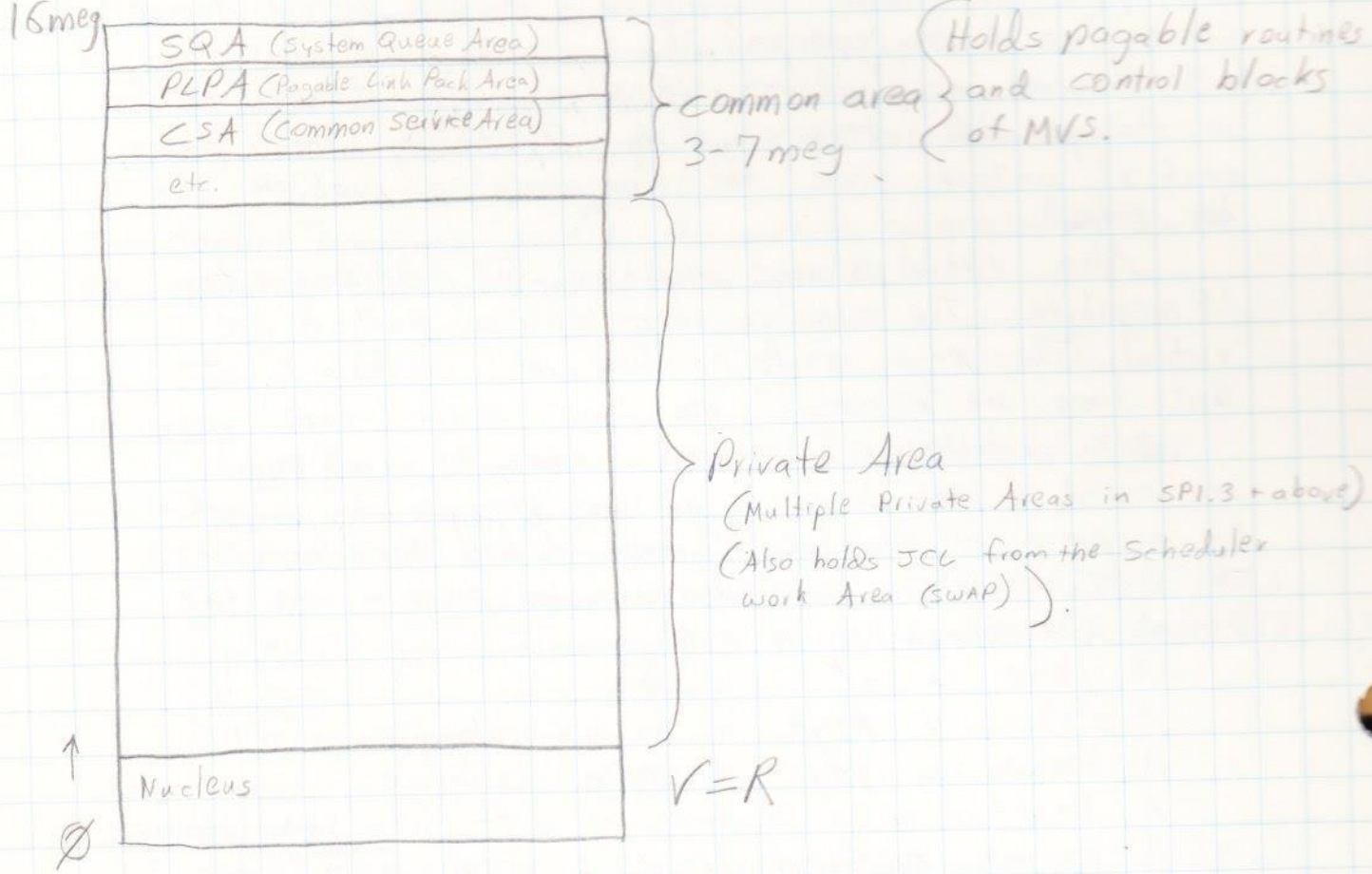
Also, virtual storage increased the addresses to 16 megabytes. The addresses above the real portion are virtual. The pages reside on disk until needed and at that time are "relocated" into lower real storage.

MVS has tables which keep track of where the virtual pages are located in real storage or auxiliary storage. This is controlled by a piece of hardware, Dynamic Address Translator (DAT). The DAT translates the addresses of the program into the actual addresses used in core.

A page is 4K.
A slot is a page in auxillary storage.
A frame is a page in main storage.
A Page-in operation moves a slot into main storage.
A Page-out operation removes a frame from main storage to auxillary storage.
Paging is the process of moving pages back & forth.
Thrashing is excessive paging and little actual problem state computing.

VS1 had a Virtual=Real option which reserved a section of memory and did not page that section. This was used for high priority applications which could not afford the time for paging.

In MVS, each program has access to an address space of up to 16 megabytes.



Data Management

Data Management handles any movement of data. It also allocates space on auxillary storage devices, keeps track of volume location, and data set management.

Data management holds the access method programs to handle all I/O requests by an application program.

The I/O Supervisor performs all requests for I/O. The access method passes requests to the IOS.

Record Formats (see handout for definitions)

Fixed length records
Variable length records
Undefined
Spanned (Record is larger than a block)

Data Set Organization

- Physical Sequential
- Partitioned
- Indexed Sequential (ISAM) - supposed to be obsolete.
- Direct Access (DAM) - access by record address
- Virtual Storage Access Method (VSAM) - KSDS, ESDS, RRDS; device independent.

Many system datasets are PDSs. (SYS1.LINKLIB, MACLIB, COBLIB, ...)

The Access Method is the middleman for I/O.
An OPEN statement causes volume verification, creates several control blocks, generates a channel program, and establish anticipatory buffering. These functions are performed by the access method in response to the OPEN.
In passing data, the access method:
- performs blocking & deblocking.
- passes I/O condition codes.
- writes trailer lables.
- At end, the control blocks are deleted.

Two basic methods: Queued and Basic.
Queued: The access method goes to the IOS for scheduling, is executed, and passes data back to requesting program. ↳records

Basic: The access method, again, goes to the IOS for scheduling, is executed, but then passes the entire block of data back to the program.

The Direct Access Device Space Manager creates control blocks on disk.
DADSM allocates space on disk, maintains VTOC, and deletes released space.

DSCB1 - DSN
Address on volume
expiration date
DS organization
Record format
LRECL
BLKSIZE
Addresses of 3 extents

DSCB2 -

DSCB3 - Address of 13 extents

DSCB4 - VTOC

DSCB5 - Freespace

DADSM uses five types of control blocks.
Volume labels } assures processing of correct data.
Data set labels } The VTOC is a Control Block.
Data set Control Block (DSCB)
data set name and location (format 4)
describes freespace (format 5)
A volume initialization program (IEHDASDR) initializes the volume, creates the VTOC, creates the volume label, inspects for defective tracks, writes a home address, assigns alternate tracks if necessary, & writes track descriptors.

An extent is a contiguous section of storage. A dataset may have up to 16 secondary extents. Each free area has a DSCB.

DSCB Format 1 - data set name
- expiration date
- address on volume
- ds organization
- record formal
- LRECL
- Blksize

Also holds location of 3 extents.

DSCB Format 3 - records beginning addresses of 13 extents.

DSCB types 1 and 3 also specify the address of last record. This permits allocation of unused space at the end of the data.

Homework: See "OS INTRO REVIEW"
Define each topic.

Copy the Acronym list from the OS/VS2 Debugging Handbook, volume 1, p. 5-90

Task Management

CPU –

System Resource Manager - "traffic cop"
Timer Supervisor

Main storage – System Resource Manager
Real Storage Manager
Virtual Storage Manager
Program Manager
Recovery Termination Manager

Channels – SRM - System Resource Manager
ASM - Auxilliary Storage Manage
PGM - Program Manager
MGR -
RTM - Recovery Termination Manager

(SRM) System Resource Manager

4 components of SRM. SRM will schedule work in all portions of the OS in accord with management objectives. SRM attempts to optimize the requests to keep the CPU busy.

Workload Manager
Resource Usage Manager
Resource Monitor
SWAP Analysis

} All provide input to SRM to balance system workload.

(VSM) Virtual Storage Manager - Maps out storage for the Nucleus, Common, and Private areas; Allocates and frees main storage.

(RSM) Real Storage Manager — Maintains tables of virtual addresses for programs in storage. Talks with ASM to page in.

(ASM) Auxiliary Storage Manager - Maintains tables of address which are paged out. Communicates with RSM to perform Page-ins.

Dispatcher Manager - Maintains CPU queues and dispatches tasks for CPU. Picks the first ready program.

(RTM) Recovery Termination Manager – maintains tables of programs in auxiliary storage. Can recover from most any program abend. Performs cleanup.

Interrupt Handler – saves hardware and system status information; Passes control to appropriate system routine. Performs PSW swaps.

(IOS) Input/Output Supervisor – handles all I/O; works with access methods. Interprets I/O interrupts.

Timer Supervisor – provides Date and Time stamps for active and waiting tasks, and elapsed time.

Program Manager – transfers control between system and application programs; Fetches application programs from auxiliary storage and places them in main storage.

Resource Control Manager –

Job Management

Tom Simpson (Amdahl) wrote HASP (Houston Aeronautical Spooling Program) for NASA. When MVT became MVS, HASP became JES. JES3 schedules jobs for multiple CPU's, and the job of being master JES3 also rotates among the CPUs. JES3 is larger and more complex than JES2.

There are 7 steps in the processing of a job.

- Input
- Conversion/Interpreter
- Initiator
- Execution
- Terminator
- Prt/Pun
- Purge

Job Management: Input, Conversion/Interpreter, Initiator

Task Management: Execution, Terminator, Prt/Pun, Purge

1. Input Services:
   detect JOB card
   assigns defaults for JES JCL
   separates JCL and Data and assigns them to separate locations on the spool pack.

2. Conversion / (If JES3, the Interpreter is here too)
   performs syntax check
   performs PROC merge
      Accesses PROCLIB
      Reads PDS
   Assigns SYSOUT data set attributes and spool location.
   Assigns MVS JCL defaults
   Creates Internal Text - the compressed and abbreviated version of the JCL. This goes back to the spool.
   Places entry on Subsystem Job Queue for the Job Scheduler.

3. Interpreter -
   Transforms Internal Text into JCT (Job Control Table) and places it in the SWA (Scheduler Work Area) in the private area. Also built ACT (Accounting Control Table), SCT (Step Control Table), and others in SWA in private area.

4. Initiator - selects Job from queue for execution.

5. Step Initiation - Enqueues data sets, & allocates virtual storage in conjunction with RSM, VSM, & ASM.
   The initiator causes an interrupt to request the services of the other managers.
   Communicates with DADSM to allocate new space on DASD volumes.
   Alerts the Timer Supervisor through an interrupt to begin timing execution.
   Finally, performs program fetch into main storage through the services of the Program Manager. The Program Manager functions through Access Method & DADSM.

6. Execution -
Access Method (Data Management) works with spool to pass data from executing job.

7. Terminator -
Deallocates DASD space through DADSM.
Deallocates main storage through RSM, ASM, & VSM.
stops timer in Timer Supervisor & records elapsed time.
check Condition Code for possible execution of next step. If next step is to run, return to process 5.

8. Catalog data sets

9. Prt/Pun
Outputs sysout datasets

10. Purge
Reclaim space on spool.

7-8-83

PTF - Program Temporary Fix - a response to a program bug. PTFs are released on tape.

APAR - a really temporary fix, usually small. Not on tape.

RMS (Recovery Management Support) - an improved error-recovery routine for Amdahl computers.

If a PTF effects any Amdahl code (RMS, for instance), a user-mod, then Amdahl must check and possibly rewrite the PTF.
Amdahl does not charge for maintenance tapes, or conflict tapes.

COMET - a new approach to maintenance.
"Consolidated Maintenance Tape" - all fixes for a particular customers are placed on one tape. Each tape is customized according to a profile of their system.

9-11-83

PSW - Hardware Prereq.

There are six kinds of interrupts:
1. Restart
2. External
3. Program
4. Supervisor Call
5. I/O
6. Machine Check

Functions of an Interrupt:

1. Store Current PSW in a main storage Old PSW location. } PSW swap
2. Load New PSW into Current PSW register.
3. Execute Interrupt Handler routine.
4. Load Program Status Word (LPSW) of Interrupted Program.

The PSW may be altered with Program Masks and System Masks to change the response to certain interrupts. The Program Mask controls Program Checks; the System Mask controls I/O and external interrupts.

7-12-83

The two bits of the condition code field can form four combinations:

00 = 0	The meaning of the code depends	
01 = 1	on the type of comparison or	
10 = 2	operation which set the code.	
11 = 3		

For Arithmetic operations, the condition code reflects the value of the result:

Result	Condition Code
$\emptyset$	$\emptyset$
$<\emptyset$	1
$>\emptyset$	2
overflow	3

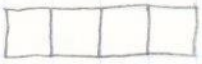
A Comparison involves a relationship between two operands:

If the First operand is EQUAL to the second, CC = Ø
If the First operand is LOWER than the second, CC = 1
If the First operand is HIGHER than the second, CC = 2

Note that the relationship is described in terms of low and high. This positioning is relative to the EBCDIC sequence order.

Condition Code Masks:

The mask is a four bit field in an instruction. Each bit position in the mask corresponds to a possible condition code value:



Bit mask
Ø 1 2 3 Condition Code

Note the left-to-right correlation.
A condition code of 1 would appear as ØlØØ.
2 ØØlØ
3 ØØØl
Ø lØØØ

The mask is referenced by its decimal equivalent. Thus, to test for a CC of 3, the mask would be 1. Likewise a mask of 7 (Ø111) would satisfy condition codes of 1, 2, or 3.

CPU States

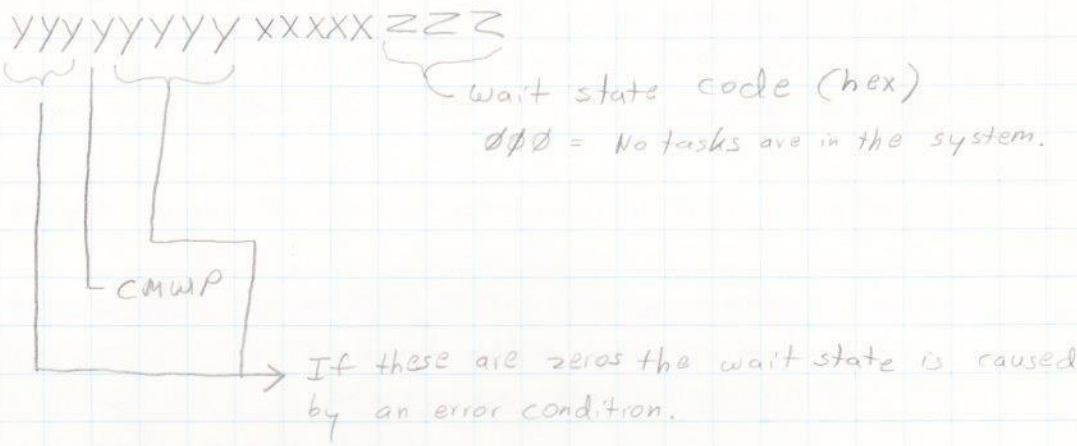
System 370 has nine possible program states:

1. Stopped
2. Operating Running
3. Operating Waiting
4. Running Problem
5. Running Supervisor
6. Problem Masked
7. Problem Interruptible
8. Supervisor Masked
9. Supervisor Interruptible

Debugging

Loops are of two varieties:
Enabled - the operator can communicate with the system.
Disabled - operator cannot communicate.

Wait States -
PSW codes for waits:



If the CPU enters a wait state during IPL, check Register 1Ø for the address of the device accessed during an I/O.

A disabled (hard) wait occurs when a task fails and the CPU is masked to ignore interrupts.
An enabled (soft) wait occurs when the dispatcher has no tasks ready.

The Address Space:

The Compiler translates symbolic names into displaced addresses. The set of displaced addresses form the address space. The address spaces of several programs may be combined by the linkage editor into a single address space.

In order to execute, the displaced addresses must be translated into real storage address. If this translation is done as the program is loaded for execution, the process is called "static relocation." If translation is performed during execution, it is called "dynamic relocation."

Dynamic Relocation is performed by the Dynamic Address Translation hardware facility. (DAT)

Question: the Relative Address is shown in the book with a segment number appended to the front of it. Where did it come from. How can the OS know which segment to look in by looking at the displaced address? (pp. 58-60)

Thad Dupper
7-13-83

MVS System Fundamentals

In earlier single-threading systems, the CPU had to wait for the completion of every I/O operation. I/O was handled synchronously.

Multiprogramming permits several programs to reside in main storage simultaneously. The CPU is allowed to execute each program in turn according to a timeslicing scheme. Since several programs are in storage simultaneously, storage is protected with storage protection keys.

Each program has its own PSW.

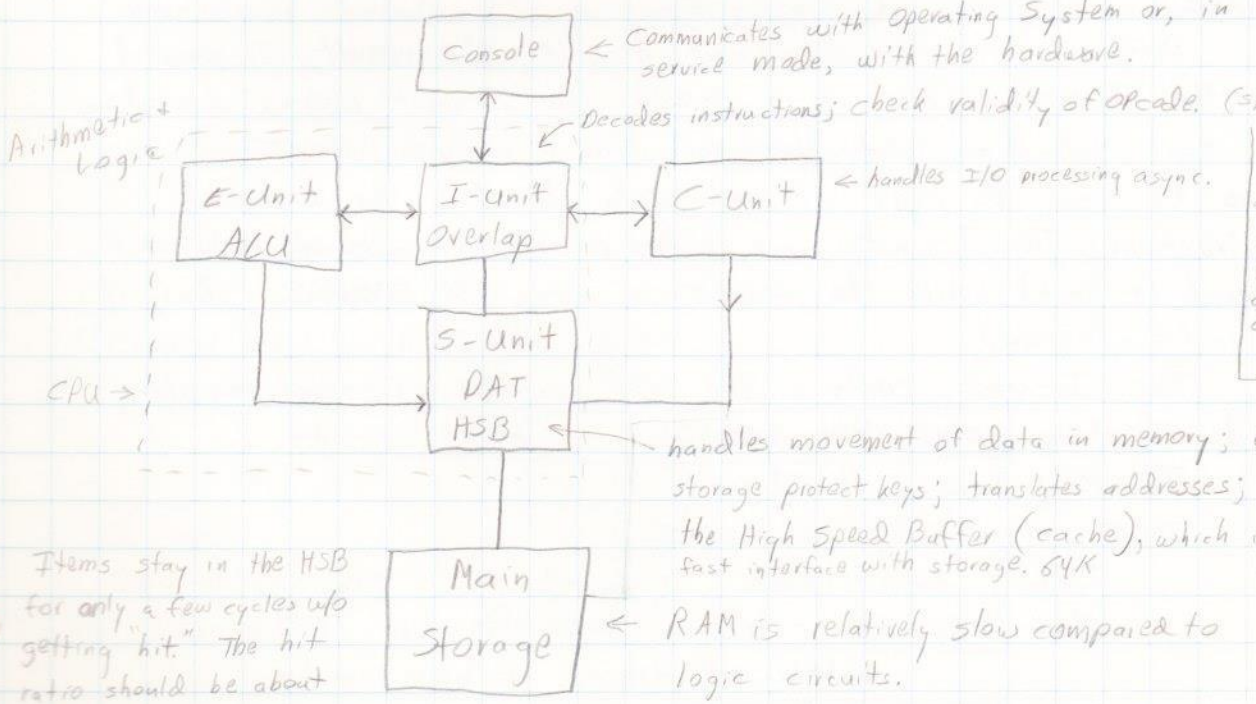
The Channel is a specialized processor which executes channel programs asynchronously with CPU operation. The CPU can switch processing to another job while an I/O completes for the first job.

Interrupts signal the need of communication between system components.

Multiprocessing - two or more CPUs sharing memory + channels.

Virtual Storage - achieved through address translation and auxiliary storage. The address space is defined larger than real storage. Only the portions in use reside in real storage.

Hardware



When an instruction goes to the pipe-line the associated operands are usually moved to the HSB.

Console

Normal mode - the operating system (SCP); "Device Support Mode:" the Amdahl console emulates several IBM console devices. The device type is specified during Sysgen. (3215, 3274, 3066, etc.)

Service mode - permits communication with hardware with commands such as IPL and Restart. shift/STOP places console in service mode. Memory can be directly altered or cleared. (Be sure to lift your finger off of STOP before shift).

Diagnostic Mode - for FE use. "Hardware Command Mode:" shift/STOP and turn CE key on. This permits more advanced commands.

Driver - the CPS operating system of the console. A minicomputer actually performs the interrogation of the main CPU. CPS runs the mini.

On the 470, the console processor is probably the weakest link in the system.

I-Unit

Performs instruction fetch and decode; the result is passed to the E-unit. Once it is passed, the I-unit begins decoding the next instruction. If the instruction is a branch, the I-unit may have to flush the pipe-line and begin decoding the branched instruction. The I-unit, since it has pre-fetched knows about a branch (+ mask) while the E-unit is doing the compare. Thus, it can be prepared.

The I-unit contains the GP registers, control registers, and Floating point registers.

The PSW lives in the I-unit.

The I-unit handles the clocks:

TOD - date + time

Clock Comparator - "alarm" goes off when TOD clock reaches set time.

CPU timer - clock which runs only when CPU is in running.

E-Unit and S-Unit

The translation lookaside buffer scans the table of the last several memory addresses used.

At the same time that address translation is starting, the TLB checks to see if the requested address was recently used. If so, translation stops & the TLB supplies the real address.

DAT - Dynamic Address Translation
HSB - High speed Buffer

Everybody uses the S-unit to access storage - E, I, & C.

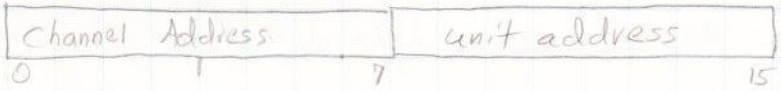
C-Unit

Input is in form of CCWs. CCWs are analygous to instructions. The CCW specifies read, write, seek, etc. A collection of CCWs is a channel program.

Once I/O has been performed, the C-unit issues an interrupt. One C-unit handles 16 channels; a 470 can have two C-units. The C-unit functions asynchronously from the CPU.

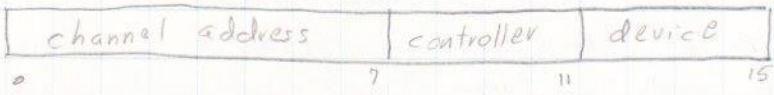
Device Addressing (p. CA90100)

Non-shared control unit - a unit record device which does <u>not</u> need a controller:



Note that the channel number is not part of the channel address.

Control unit with 16 devices:



Control unit with 32 devices:

↑ limits the number of CUs & increase the number of devices.

Channel Types (p. CA00110)

Three types: selector, Block, & Byte.

Selector: Channel is busy for the length of the I/O; No multiplexing on selector channels. The channel issues the Set File Mask (in response to IOS) to the controller to permit certain kinds of access (RO, RW, etc.).
Selector channels have no subchannels & are used primarily for tape.

Block: can multiplex I/O operations.
After the SFM has been issued to the controller, the channel is freed. Later the seek is issued. The controller will alert the channel when an operation is complete, or in the case of a read or write when the appropriate sector is about to be under the heads. Inbetween communications to/from the channel and a particular controller the channel can talk to other controllers.
Blocks of data are transferred from the device to buffers established by the channel. ~~On~~ Separate I/Os must be issued for each block.
Used primarily for DASD.

Byte - like block, except that the unit of data transfer is a byte rather than a block.
Used for printers & CRTs.

Channel 0 in MVS must be a byte multiplexor channel.

Channel Operation

An SIO wakes up the device.
Location 48 (hex) should already contain the CAW which points to address of the channel program.
The I-unit starts the I/O. The C-unit returns the I/O interrupt upon completion of an I/O. The C-unit updates the CSW in hex 40 for the I-unit to check the status of the I/O operation.

Channel Status Word (see yellow card p. 18)

The CCW address in the CSW is really the address of the next CCW. Thus, upon end of channel program, the address contained in the CSW is 8 bytes too long. Be sure to subtract 8 bytes to locate the channel program that was really executed.

The CSW, like the CCW & PSW is a doubleword.

The CSW has two important flags, bits 32-39, and 40-47. These indicate unit status and channel status. These codes are like masks. 08 = channel end } good I/O
04 = device end }
0C = channel/device end.

CCW fields (see Yellow Card, p. 11, p. 18)

0 - 7	Command Code
8 - 31	Data Address; a buffer address in MVS (for either input or output depending on operation).
32 - 37	Flags
38 - 40	Not used 00
40 - 47	Used by BTAM
48 - 63	Byte Count; length of data to be transferred.

Flags (see p00)
00 - last CCW in channel program

CCW may be written as a series of DCs.

```
DC  X'
DC  data
DC  flags
DC  length
```

The byte count left after the I/O is recorded in the CSW. It should usually be zero if I/O completed successfully. The control unit will decrement the byte count each time data is passed.

Exercise: (p. CA00140)

		correct
DC	X'02'	DC X'02'
DC	X'064000'	DC X'064000'
DC	X'00'	DC X'00'
DC	X'00000'	
DC	X'00000320'	DC XL3'320'

Channel Status Word

Main Storage

Main storage holds data and instructions.
Addressing:

Logical addresses are the displaced addresses within the program.
This is input to DAT.

Real - output of DAT, input to prefixing.

Absolute - A single prefix register, used in multiprocessing.
Each CPU has its own first 4K of storage for PSW, CAWs, CSW, & other data. Each has its own prefix Storage Area (PSA). The prefix register determines which PSA area the CPU should use. On a UP, the Register is Ø.
Register is one word.
Two instructions are used to set and store prefix register values.
The Prefix Register only notes bits 8-19 of the address.
(see p. CA ØØ2ØØ)

Storage Protection - the key in the PSW must match the key of all storage locations which the program attempts to reference.

Protect key is 1 byte.

0-3 is the actual key.

4 Fetch Ø = Fetch yes, Store = No
1 = Store No, Fetch = No

5 Referenced Ø = not referenced since page in.
1 = page has been accessed

6 Change Ø = No
1 = contents have been changed since page in.

7

The Change and Reference bits are used by the paging manager.
If the PSW key does not equal the storage key, then the storage controller checks the Fetch bits.
PSW key of Ø can do anything, anywhere.
No store can take place unless the keys match or the PSW has a key of Ø.

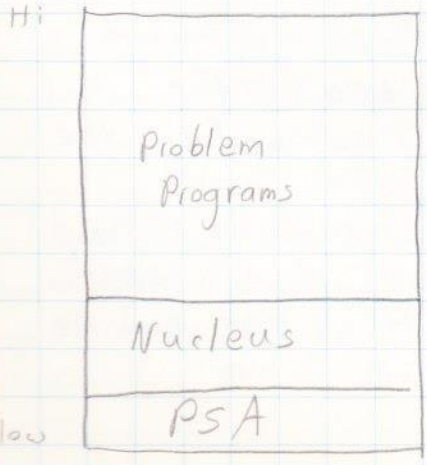
Hardware has the storage Protection byte for every 2K, but software treats storage in 4K chunks.

with only 16 keys how can we have more than 16 users ? ~~segment Protection bit in the segment table entries.~~ ASN Authorization Table & CR4. (5-27, R00) overview, p 2-5

Do "Computer Architecture" exercise.

Transfer in Channel is usually used to branch to the channel program after SFM + Seek.

Main Storage Layout



← heart of the OS

} 4K

A PSA will exist for each CPU in an MP configuration.
Problem Programs can only issue non-privileged instructions.
The P bit in the PSW determines whether a program can issue privileged instructions.

PSW

Two types: BC = Basic Control
EC = Extended Control

Bit 12 of PSW determines BC or EC mode.

15	Problem = 1 Supervisor = Ø
16	Secondary Space Mode - permits communication to other address spaces
14	Wait = 1 Running = Ø
5	Translation (DAT) = 1 No = Ø
18-20	Condition Code
8-11	Storage Protection Key
40-63	Address of next instruction. (Rightmost word) The instruction address is not updated until the last phase of decoding in the I-unit. The I-unit anticipates which instruction to decode.
0-7	System Mask bits for masking interruptions
20-23	Program Mask bits for program interruptions.

Usually, instructions are not interruptible.
There are 3 instructions in S/370 which are interruptible.

1. CLCL - Compare Logical Long
2. MVCL - Move Character Long
3. TB - Test Block

Instruction Address - when the I-unit decodes the OP code, it determines the instruction length

The I-unit multiplies the ILC by 2 bytes for the actual length. Upon completion of the instruction the I-unit updates the PSW Instruction Address by that amount.

If an interrupt occurs during execution of an instruction, the I-unit subtracts the ILC from the PSW Instruction Address. then handles the interrupt and

In debugging dumps we must subtract the ILC from the PSW Instruction Address.

7-14-83

Interrupts :

SVC Interrupt (see Debugging Handbook, Vol 1. p. 5-4)

The SVC interrupt is a system routine which supplies supervisor services to problem programs. The SVC interrupt handler decodes the request and passes it to the appropriate routine.

For instance, the OPEN macro in assembler is expanded into several instructions including a SVC 19_{10}. (SVC operands are coded in decimal).

To begin executing, the SVC routine has its PSW (X'60) loaded and then looks at the PSA for details on the interrupt.

SVC interrupts cannot be masked. There are up to 256 kinds.

Program Interrupt

Commonly incurred by issuing a privileged instruction; Also, if the hardware has not been updated to include new instructions.

Other problems: Data exception
Protection exception
Fixed-point overflow
Decimal divide exception

There are 25 Program Interrupt codes (yellow card, p. 19). Codes ØØ10, and ØØ11 (hex) are not necessarily errors. They signify segment exceptions and page faults. The page fault SVC alerts the OS to initiate a paging operation to bring in an address.

The interrupt code is contained in the SOCx abend code.

Some program interrupts can be masked (SPM instruction), such as exponant & decimal overflow. The code is the sub-class.

Machine Check

The hardware is constantly checking things like parity. When an error occurs, the hardware issues the interrupt. Then a routine tries to diagnose and circumvent the error.

Two types : Exigent (hard) error - is not recoverable.
Repressible (soft) error - recoverable.

Machine checks can be masked; the machine check interrupt handler should not be interrupted by another machine check. Control Register 14 contains 8 bits which control response to machine checks. If required only some types of machine checks can be masked. Bit 13 of PSW is the mask.

External Interrupt

Allows communication among MP CPUs, Timer events (Clock comparator). The External New PSW (x'58') points to the routine.

The Signal Processor (SIGP) instruction causes external interrupts among CPUs.

The console interrupt key will also generate an External Interrupt. (If all Operator buffers (WTO buffers) get full the system can lock up. An External Interrupt will clear the buffers).

The E-bit (7) in the PSW masks External Interrupts.

Control Register Ø contains bits which can enable/disable certain subclasses - Clock Comparator, + CPU timer.

I/O Interrupt

I/O Interrupts occur at the completion of every I/O event. The I/O New PSW (x'78') points the Interrupt routine. The reason for the Interrupt is contained in the CSW (x'40')

Bit 6 in the PSW sets the mask. 1 = Enabled, that is, the Interrupt will get handled.

D = Ø
E = 1
> Alphabetic order.

Control Register 2 contains masks for each channel (0-31). The Channel number is the sub-class of the interrupt.

Restart Interrupt

Causes the Restart PSW to be loaded and executed. The Restart Interrupt cannot be masked.

This is usually operator initiated to get out of a disabled wait, or a disabled loop.

The Restart does not necessarily reload the system. The Restart routine looks around to see why he has been invoked. Some disabled waits are recoverable (page pack drops ready).

The PSW can be altered by the Set System Mask (SSM) instruction to alter the masks of maskable interrupts. This is a privileged instruction.

The Load PSW instruction (privileged) is used by the OS to swap PSWs.

A problem program can alter the Program Mask bits of the PSW with the SPM instruction.

Only during a disabled wait is a wait state code placed in the instruction address area (right 12 bits).

4 Possible combinations:

Enabled Wait - no work wait
Enabled Loop - will accept interrupts but running
Disabled Wait - no interrupts, not running
Disabled Loop - no interrupts accepted, but running.

"Loop" is not by definition bad; it means running.

Wait State codes are located in Debugging Handbook, Vol 1, p. 46.

Once the system is "Enabled Wait, only an interrupt can start processing again.

PSW Swap

PSW swap is the heart of multiprogramming.

An interrupt causes the Current PSW to be placed in the associated Old PSW area in PSA and loads the appropriate New PSW to the Current PSW. This is done by hardware. The New PSWs point to their interrupt handlers.

Old PSWs are placed in areas unique to the type of interrupt. Thus, for an I/O interrupt the problem program's PSW is placed in the I/O Old PSW area in PSA. ~~This is how the interrupt handler can return control to the problem program.~~

Debugging

Display PSW and ascertain status:

Bit 14 - 1 = Waiting
0 = looping

Bits 6+7 If either is on, the system is enabled.

The old PSW is not used to return control. After an interrupt, the handler returns control to the dispatcher. If the dispatcher chooses to execute the job that was interrupted the PSW is fetched from the TCB associated with the job. This TCB PSW field is also updated at the same time as the PSW was loaded into the I/O OLD PSW field in PSA (for example).

When masked for interrupt, the interrupt is not lost. It is queued.

Do Exercise CA00400 + dump exercise from last night.

MVS Introduction

MVS runs on 4300 class and larger S/370 machines.

company	machine
NAS	AS5000
	AS9000
IPC Systems	
Magnuson	
Amdahl	V5 - V8

MVS was announced in early 1970s. MVS is the state-of-the-art operating system for large business-class computers.
MVS/XA uses 31 bit addressing.

Amdahl machines are optimized for RX + RS instructions. They are faster, more reliable, and cheaper than IBM counterparts. PCMs brought the prices down for everyone. The end users benefit.
Systems and hardware must be upwardly compatible. Downward compatibility is not guaranteed.

The 360 family was planned to be upgradable. This was a first.

Around 1978, IBM "unbundled" their software. Previously, software was free with hardware.

MVS/SP releases:

1.0
1.1
1.1.1
1.2 - withdrawn by IBM

1.3.0
1.3.1
1.3.2
1.3.3

Allows communication between address spaces.

MVS/XA 2.1 - current & very expensive.

Compatibility is maintained for Services, Control Blocks, and JCL. Upgrades should be possible with little or no change. Control blocks do evolve & care should be taken when upgrading, particularly when skipping a release. Often, control blocks are extended by chaining.

Task Structure

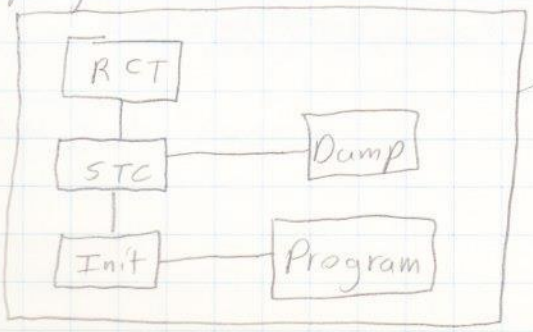
As soon as an Address Space is created, several Task control blocks are created:

RCT: Region Control Table will control swapability. Region Contol will issue an attach to chain the Started Task Control (STC) which will create the initiator task and the dump task. (The dump task stays idle until needed).

Batch jobs execute in an address space already in existence. TSO creates an address space for each logon.

The initiator will request work from JES, allocate resources, serialize (queue for a resource) using ENQ macro (which expands to an SVC) or a LOCK (for resources in MP environment. The CPUID is placed in the lockword). The initiator will assign attributes to the job - non-cancellable, special-key, etc. These assignments are made from the Program Properties Table.

Then the initiator will issue an attach macro to the problem program.



Address Space

When done the Recovery Termination Manager will undo the things the initiator did.

Program Manager will fetch, or make available, all load modules requested by the job.

Attach -
Link -
Load - make accessible; add the address to list of load modules.
XCTL - transfer control of task to this program; usually one-way.

To build a TCB, issue an ATTACH macro which will create the TCB and Request Block. The Request Block contains the information about a resource which the task wants. The RB is accessed in servicing the task's request.

Both the TCB and RB have register save areas. When a job is interrupted, its PSW is stored in the TCB. (see dump, p. 52) The Interrupt Handlers save status in appropriate Old PSW area, store Registers in TCB, + store PSWs in RB. (wazey)
The Dispatcher restores the Registers from the TCB and the PSW from the RB. The Dispatcher maintains tables of task statuses. The Dispatcher runs after every interrupt handler.

A TCB can have many RBs but only one active RB.

The Task, TCB, & RB are all in the Private area.

The Dispatcher just checks TCBs and his lists. The Dispatcher will check the TCBs of tasks in descending priority order. When it encounters a dispatchable task, it selects it to run.

Requesting I/O

The access method called upon is determined by the data set type and the type of macro issued (Read, GET, etc). The Access Method builds the channel program, and asks IOS to schedule it. IOS issues the SIO and loads the CAWs. The job will be marked as nondispatchable.

The Access Method issues SVC 0 (EXCP) to alert IOS to the I/O. IOS loads CAW with info passed by the SVC and issues SIO. This is the front end of I/O processing.

Back end: when I/O is complete, the I/O Interrupt Handler saves status, checks CSW, + tells Access Method that the I/O is done. This is posted in fields established by the Access Method (via SVCs WAIT + POST).

The Wait Count in the RB shows how many events the Task is waiting for. The SVC Interrupt handler marks the TCB as nondispatchable; then the wait count is incremented, signifying that the task is waiting on one more event. Upon a return from the Access Method, the Wait Count is decremented by 1. When zero is reached the task is marked dispatchable.

Task Supervisor

The Task Supervisor goes between the TCBs and the dispatcher. This routine performs the wait and post functions. Also, task supervisor will mark a TCB as nondispatchable if the task queues on a resource.

The Recovery Termination Manager controls both normal and abnormal termination. Issues Freemains, + deallocates resources.

In Abends, RTM checks for ESTAE macros (a recovery routine written into the job itself!), and produces a dump. ESTAEs are only for problem programs. MVS routines use an FRR (Functional Recovery Routine) to diagnose the error. Several FRRs may be used. The esculation process is called "percolation." The goal is to save the system.

The MVS system command D, D will display any FRR dumps (SVC dumps) which might have been taken. On a normal system the dump data sets should be empty.

Do "MVS Intro" Exercise.

7-15-83

Virtual Storage Overview

Read Green book for Monday.

During Assembly, relative addresses are assigned to each program statement.

In the beginning, the linkage editor could only create non-relocatable load modules! Next, with S/360 came Static Relocation which could relocate, but storage still had to be contiguous. This was a great improvement but still resulted in significant storage fragmentation.

Segmentation was developed to avoid fragmentation. The start point of every segment was kept in a table. The address structure then referenced the segment number, and the displacement within the segment. A segment table was created for each job. The beginning address of the segment (obtained from the segment table) would be added to the displacement within segment to create the real hardware address.

The Segment Table Origin Register (STOR) contains the address of the segment table. The STOR is loaded whenever the program is executing. The above process is called a "one-stage lookup."

Segments may be subdivided into pages. A page is 4K in size. With this scheme, a two-stage lookup is required. The address structure now requires the segment number, the page number within the segment, and the byte displacement within the page

The address is still only 24 bits in length. This form of address is a "virtual address." A Page Table is required. The Segment Table now points to its very own page table. Each page in the page table has listed its starting address. Thus, in decoding the virtual address, the system finds the address of the Segment Table from the STOR, goes to the segment table and finds the entry for the segment it needs and reads the address of that segment's page table. The system then

goes to the page table, and using the relative page number as a key, finds the beginning real address of that page. This address is added to the displaced address within a page (that is contained in the virtual address) to produce an absolute address.

Note that pages do not need to be contiguous within a segment. The pages may be "scatter loaded" to any 4K areas in storage.

> Who creates the virtual address?

Segments are 64K and hold 16 pages.

A Page Frame Table records status for every page in memory. The system has only one page Frame Table. Each entry holds:

Program ID of owning program.
Segment and Page number of virtual page.
Status of frame:
1 = in use
0 = not in use.

Having the Segment in Page number in the frame table allows the system to go backwards from real storage to the owning program and the virtual address.

Unused pages within a segment are not allocated. The segment table may only hold, for instance, one page table pointer. Likewise, the page table will only contain entries for pages which actually contain portions of the program. If the program needs to grow dynamically, additional page table entries will be validated. If necessary, additional segment table entries will also be validated as well as another page table.

Associated with every page table is an External Page Table which records the location of pages on auxiliary storage. The contents of the Page Table and the External Page Table comprise entries for the entire address space.

The EPT is contiguous to the page table.
The page table has an Invalid bit.

1 = page not in memory
Ø = page in memory.

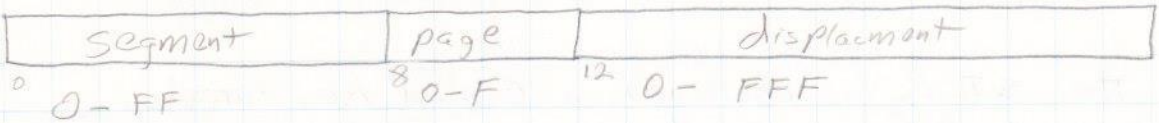
If the page table is searched and the Invalid is 1, then the system goes down to the EPT to get the information necessary to perform a page-in operation. (see VSCØ22Ø, handout book for diagram).

When a page fault occurs and a page is moved into storage:

1. the page table invalid bit is changed and the real address of the page is recorded.
2. the Page Frame table receives the program id and the page + segment number, and the status flag is reset to 1.
3. The EPT entry is invalidated to show that the page is now in real storage.

16 meg can hold 256 segments.

The Virtual Address



The page frame table actually has several flags:

Status 1 = in use
Ø = not in use

Reference Ø = not referenced since page-in
1 = has been referenced

Change Ø = has not been changed since page-in
1 = has been updated.

When MVS needs a page it will scan the page frame table for a page with a status of Ø. If none are available, the system will "steal" a page. First it will look for a frame with the reference bit of Ø. If it finds one, then the system will not page it out since the copy out on disk is the same as in storage. It will simply reset the flags in the appropriate tables and thereby save an I/O. If the change bit is on the frame will be paged out.

Translation Lookaside Buffer

As soon as the system has the virtual address that it wants, hardware scans a Translation Look-Aside Buffer to see if it may also contain the virtual address that is needed. If found, the other translation process stops and the real address is picked up from the TLB.

The TLB contains the last several address that have been successfully translated and found to be in real storage.

The TLB process is much faster than the virtual storage translation process. TLB helps because a program usually references a relatively small number of pages in a given time.

The STOR is contained in Control Register 1. It is loaded by hardware when a task is initiated.

Note: how does optimum working set size relate to TLB size?

Control Blocks

Volumes 2 & 3 of Debugging Handbook describe most MVS control blocks. Vol 1 describes dump and trace formats, Problem Analysis, and debugging.
Section 2 covers IPCS and dump formats.

See Vol. 2, p. 67
Macro ID - is a DSECT of field labels.
Created by - when, & which module did creation.
Subpool & Key - shows subpool location in the Common Area (see Vol 1 p. 5-77) and storage protect key.
Epaals Subpools are a logical way to divide storage.
Size -
Pointed to by - where pointer lives. (May be multiple pointers)
Serialization -
Function - short description.

The Acronyms are in alphabetical order & are printed in the bottom right (& left) corner of the page for easy access.
Negative offsets denote a prefix area.
Some control block sections will have a cross-reference of labels within the control block. The labels are in alphabetic order with their offset & value.
Offset = displaced location from beginning of control block
Value = bit setting & particular label. Each bit setting has a label (for the whole field). Each label is an alias for the entire byte. The bit pattern shown shows which bit is of interest when using a particular label.

Exercise:	offset hex	Value	Definition
UCBID	2	F F	UCB identification
UCBONLI	3	1000 0000	Device is online
ASCBASID	24		Address Space Identifier
CVTMVS2	74	Ø 1	Multiple Memory Option

Dumps

SNAP- dump at user request. of load module, registers, TCBs, RBs

ABDUMP - SYSUDUMP - small; about like SNAP
SYSABEND - all storage
SYSMDUMP - unformatted

SVCDUMP - Error while in key Ø; output of IFR.
Writes to Sys1.DumpXX

SADUMP - Stand alone dump from dead system.
Records memory on tape. IPL is necessary after SADUMP.

The CVT is the anchor for almost all control blocks. Find the CVT and look up the pointers to whatever control block is of interest.

The addresses in a formatted dump are Virtual Addresses. Each line is 20_{16} bytes. The address of the CVT is usually at hex 10 and hex 4C.
(if store status was issued.

Exercise: unit type (& everything)
Find the device type for the UCBs at locations:
57FØ
53D8
6090
26B8

		p.91 dump	p. 90	p. 92	p. 84
		57FØ	53D8	6090	26B8
X'2'	Valid UCB	FF	FF	FF	FF
X'0'	Device Addr (x'4)	741	6A4	947	ØAE
X'3'	Online	yes	no	no	yes
X'3'	Allocated	yes	no	no	yes
X'5'	Busy	No	No	No	No
X'12'	Device Group	DASD	DASD	DASD	Tape - display
X'1C'	Volser	EDCPGE	—	—	—
X'10'	Device Type	3350	3350	3350	~~3158~~ 327X

7-18-83

Storage Management

Three components control storage:

VSM - maps Vs for each address space
handles requests for acquiring + releasing storage
maintains CBs.

RSM - control use of real storage
maintains CBs.

ASM - controls auxiliary storage
maintains CBs.

VSM - tracks the use of common system area.
Assigns protect keys to blocks of storage for system components.
The nucleus and common areas are mapped in each address space with common tables. (see Debugging Handbook, Vol. 1, 5-74).
Assigns fetch protection to certain control blocks.
Assigns pagable status.
Assigns common or private status to certain CBs. Until SP1.3, all inter-address space communication passed through the common area.
(DAS) The SP1.3 feature is called Dual Address Spacing. This is horizontal communication.
The Common Area got too small for IMS.
In XA, DAS may not be as important.

Subpools - a group of logically related control blocks labeled by a number. The subpool number:
0-127 - user (acquired with Getmain, SVC
128-255 - system
this is an artificial division. The attributes of each are listed in the Debugging Handbook, Vol. 1, 5-77.

Storage Areas:

Nucleus - bottom of real and virtual storage. After IPL the nucleus will not change in size; field contents may change. Holds PSA, UCBs, Interrupt Handlers, some SVCs, + Page Frame Table (PFT).

Fixed BLDL Table - optional directory of frequently used data sets. Actual PDS directors can be kept here. This may be done to improve performance.

Fixed LPA (Link Pack Area) - optional field; may contain copies of not-frequently used user-type load modules but which need to be accessed very quickly. used in support of online routines.

RMS Nucleus Extension – Error and machine check handlers. This space is larger for MP and AP configurations. Used for Alternate CPU Recovery (ACR) in which a surviving CPU can take over the work.

The above areas are not swappable.

(A swap is a massive page operation, i.e. when a task becomes idle. Also, swapped data sets reside on swap data sets.)

Private Region –

System – 16K, pagable; used by Region Control Task (RCT) the first task in an address space. Controls swapping and dump facilities.

V=R areas – subset of Private area reserved for non-paging tasks.

V=V areas – permits paging + swapping.

Only crucial applications, those which use time stamps, etc. can use V=R. Also, some software monitors may run real (Omegamon). Nowadays Virtual Storage Constraint is a problem: an installation has the real storage frames, but not enough virtual.

Most private areas actually hold about 3–8 megabytes.

Subpools 229-230 – job related control blocks; DEB

LSQA and SWA – LSQA is swappable but not pagable.

SWA – job management CB's, JCL information.

↳ Scheduler Work Area.

The Local System Queue Area holds segment and page tables for the address space. LSQA is page-fixed. When swapped the whole area is out.

SRM decides when to swap an area out. The job cannot be at all referenced when swapped out. The SRM returns the job to execution by bringing in the Segment and Page Tables.

The RCT talks to the LSQA a lot.

Common Area

The online systems have caused the CSA to grow very large in recent years; IMU is a CSA hog.

The Common Service Area is addressable to all users. CSA is pagable; size is set at IPL time. Nonswapable

PSA (for MPs Only) - this is the second "low core" for a multiprocessor. This is accessed through Prefixing. Nonswappable.

Pageable BLDL Table - holds PDS directories of system datasets and page on an LRU basis. Fixed and Pageable BLDLs are mutually exclusive.

Modified LPA - used for test routines. This area is searched before PLPA. During IPL a particular module in MLPA may be dropped. The system would thus default to the PLPA. Pageable, Nonswapable.

PLPA - about 4 meg; Pageable, Nonswappable. Holds frequently used routines: SVCs, Access Methods, open/close/EOV logic, + other system + online-system routines. The frequently used routines will tend to be in storage most of the time due to LRU rule. This area is loaded from SYS1.LPALIB. At IPL time, this dataset is loaded along with the Link Pack Directory. There is a special LPA page dataset. This is simply a source for the normal copies in storage. No page-outs occur because LPA modules are change-protected. Lots of MVS modules live in PLPA. PLPA is often as large as the CSA. Only reentrant code is allowed.

System Queue Area - allocated in 64K segments, a minimum of three. Fixed + nonswappable. Holds system control blocks which are used by system routines in nucleus and LPA. Also holds segment, and page tables for the common area, and nucleus.

This area is the top of virtual storage.

At every start, logon, or mount, an address space is created. If a GETMAIN for subpools Ø-127 is issued, area in the private area will be allocated to the task.

During start-up, MVS creates 5 address spaces for itself, and 1 for JES. DAS permits communication among the system address spaces.

ASID 1 - Communications Task
ASID 2 - PC/Auth
ASID 3 - Global Resource Serialization
ASID 4 - Dumping Services
ASID 5 - Allocas
ASID 6 - JES (usually)

Often, a security package like ACF or RACF will come in before JES.

ASID 7 - TCAM or VTAM (usually)

The Address Space Vector Table (ASVT) contains a maximum-user limit. This limits the number of ASIDs. The ASVT only creates the specified number of ASIDs. One must be available in order for an address space to be created. The ASVTE will point to an unused ASID. (ASVTE = Address Space Vector Table Entry)

SYS1.PARMLIB contains all of the parameters used to tailor MVS to the environment and management objectives. These parameters are adjusted during SYSGEN.

During Virtual Address translation, the Segment Number and Page Number (Virtual Block Number, VBN) are converted into a Real Block Number (RBN).

Hardware Components for Address Translation (see Yellow Card, p. 21)

PSW - bits 5+12 on for DAT+EC
CR Ø - segment and Pagesize (can be 2 or 4K - bits 8-9) Segment size 64K
CR 1 - STOR of current task
Translation Lookaside Buffer - a 64K buffer of last successful translations.
STO stack - the list of Segment Table Origins for different tasks.
Also called SBR - segment Base Register.

Control blocks needed for address translation:

1. Segment table - if not yet allocated, an invalid bit will be on.
2. Page table - 16 pages/segment
3. External Page table - has an entry for each page in the load module.
4. Page Frame Table - one per system; controlled by RSM.
5. Page Vector Table - pointed to by CVT + 164_{16}.

(See Diagram SMØ13Ø for translation diagram w/ page fault).

RSM Available Frame Queue (AFQ) - RSM's list of available frames. If none are available, a page must be stolen. Once a program interrupt 11 (page fault) occurs, RSM must be asked to supply a real frame.

Page stealing will occur whenever the AFQ falls below a certain value. RSM will check the Unreferenced Interval Count (UIC) of each frame. Each time the RSM scans storage, it will add 1 to UIC. Whenever the page is referenced, the UIC is set to zero. When RSM needs to steal a page it will take one with the highest UIC.

If a lot of page stealing is occurring, the customer may need to buy another 4 meg.

A segment table entry (SGTE) is 4 bytes in length. Thus, the (segment number * 4) + STOR = segment table entry. The page table origin + (page number * 2 bytes) = Page table Entry (PGTE).

See SMØ13Ø.

Page Reclaiming - when the AFQ goes up, RSM steals pages and places them on the AFQ. However, if that page is referenced before it is overwritten, SRM will reclaim it rather than perform a page-in operation. The page table still points to the correct frame.

Every time a page fault occurs, SRM first checks the AFQ before performing page-in. Then it only needs to change the valid bit in the page table, and remove the page from the AFQ.

The purpose of page reclamation is to avoid I/O.

Control tables necessary to translate from Real to Virtual.
The CVT points to the PVT. Check PVT+C for apparent origin of Page Frame Table. The PFT will have an entry for every page frame in Real Storage.
Each Page Frame Table Entry will hold the ASID of owning address space and Virtual Block Number.

PFTE

To translate:
Find ASTE field in ASCB.
Look a ASTE+8 for STO. This STO is a real address.

Exercise: Translate 465CØØ from real to Virtual.

EECØØ = PFT origin
4650 = VBN * 10_{16}
F3250 = PFT entry for appropriate VBN

PFTE 000B | 7FFD | 01FF | D000 | 0-0 | 20058400
ASID | VBN | PTA | PTR

double check this against ASCB. ↑

VBN + Displacement
7FF + CØØ (appended) = 7FFCØØ

Real segment table address of ASID = 0003 is 3EBCØØ (p.24)
check CVT for address of PVT (Field = PVTP) = Ø3COAØ (p.77)
PVTP is location of Apparent Origin of PVT ØEECØØ (p.208)

EECØØ = PFT origin
3EBØ = VBN * 10_{16}
F2ABØ = location in PFT of appropriate entry (p. 612)

12
14
26
16

PFTE = 0003 | 7FFD | 07E30000
ASID | VBN

↓

7FFCØØ = STOR Virtual Address
displacemot

When doing a Virtual to Real translation, we must do two real to virtual translations.

1. The STOR contents is real and must become virtual.
2. The Page Table Address in the Segment Table is real and must become virtual.

Exercise V to R

7CFØ4Ø
seg | page | displacement

Go to ASTE for STOR (in ASCB) FCBØBØ (p. 40)
FCBØBØ contains STOR 465CØØ (p. 645)
CVT has field PVTP = 3CDAØ (p. 77)
3CDAØ + C = Apparent Origin of PFT
contains EECØØ

SGTE = F017B7BØ
Real Page Table Address

PFT + RBN * length = Address of PFTE
EECØØ + (17B7 * 10_{16}) = FØ3BØ ← p. 606
↑ p. 208 ↑ p. 886

VBN = 7E1
displacement 7BØ
7E17BØ = Virtual Address of Page Table.

Page No. * 2 (length) + Virt address of PGT
(F * 2) + 7E17BØ = 7E17CE (p. 870)

↓
18C1
RBN ← Flags
↓
0001

seg | Page | displacement

Convert V to R DD | Ø | 215

(see VT<0010 in workbook)

1. Real address of segment table = 465CØØ (p.40)
2.
3. Find CVT (p.077)
   CVT field PVTP contains address of PVT = 3CDAØ (p.77)
   3CDAØ + C = address of PFT = 3CDAC
   3CDAC contains EECØØ, Origin of PFT (p.208)

```
  EECØØ   PFT origin
+ 465Ø    Displacement into PFT for appropriate segment * 1Ø
  F325Ø   Virtual address of PFTE (p.631)
```

Contents of PFTE = ØØØB | 7FF | Ø
ASID | VBN

Virtual Address of STO 7FFCØØ

5. DD * 4 = 374 segment number * 4 bytes
   11|01 11|0100 ← multiply by shifting

6.
```
  7FFCØØ   virtual STOR
+   374
  7FFF74   virtual address of SGTE (p.886)
```

7. Page Table Real Address = 7D8Ø4Ø

```
  EECØØ
  7D8Ø
  F698Ø   Virtual Address of PFTE (p.621)
```

7D8 * 1Ø = 7D8Ø

Contents of F698Ø = FD7Ø4Ø Virtual Address of Page Table

8. D * 2 = 1A Page Number * 2

9. 1A + FD7Ø4Ø = FD7Ø5A address PGTE (p.669)
10. 458
11. 167
12. 458215

In DAS, a secondary STOR resides in CR7.

Homework: Convert V to R : 11ØBCØ
ØØØØ4C

Note: The Segment Table Origin and the Page Table Origin begin on page boundaries. This is why it is possible to find their Virtual Addresses in the Page Frame Table.
Check the Page Frame Table for the appropriate virtual address using the three most significant digits of the Real Address as an index. ?

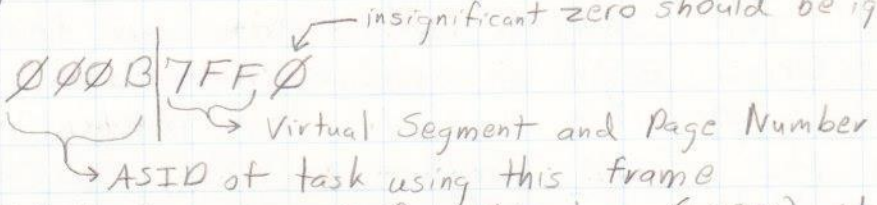
Convert Real to Virtual:

1. Locate the CVT (usually at the beginning of the dump, right before unformatted PSA).
2. Locate the field PVTP within the CVT, at hex 164.
3. Goto location specified in PVTP.
   At hex 'C' into the PVT is the virtual address origin of the Page Frame Table.
4. Each Page Frame Table Entry is hex '1Ø' in length.
   Turn to the origin of the Page Frame Table. Take the three most significant digits of the Real Address we are converting, and multiply it by hex '1Ø.' The result is how far into the Page Frame Table we must go.

   Example: Real Address = 465CØØ
   Origin of Page Frame Table = EECØØ

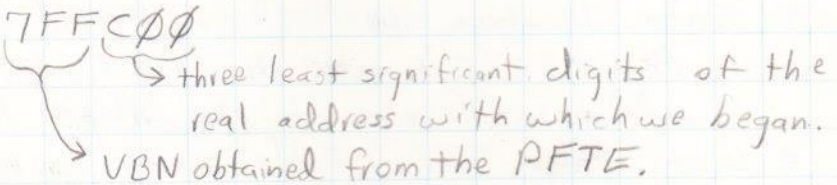
```
  465   = 3 most significant digits
* 1Ø    = hex '1Ø' bytes / entry
465Ø    = depth into Page Frame Table

EECØØ   = Origin of Page Frame Table.
+ 465Ø  = depth into Page Frame Table.
F325Ø   = Virtual Address of the entry in the
          Page Frame Table that we want.
```

5. Locate the Virtual Address of the entry in the Page Frame Table that we calculated. Volume three of the Debugging Handbook describes the meaning of PFTEntries.
Example:



The Virtual Segment and Page Number (VBN) that we find here may simply be concatenated to the front of the rightmost three digits of the Real Address that we started with.
This works because the specific byte which we wish to find is the same displacement from the start of the page in real storage as it is from the start of the page in virtual storage.

Example:



Convert Virtual to Real

1. First, convert the Real Segment Table Origin to a virtual address. The real STO may be found in a formatted dump printed below the ASCB for the task in question. If it is not there, the field ASTE within the ASCB points to the beginning of the ASTE which holds the value of the Segment Table Origin. The Real Segment Table Origin is located at offset hex '9' in the ASTE.

2. Each entry in the Segment Table is 4 bytes in length. Therefore, now that we know the Virtual address Origin of the segment table we can displace into it 4 * Number of Segments.

Example:

11ØBCØ = Address we want to convert to Real.
7FFCØØ = Virtual address beginning of Segment Table.

```
  11   = Number of Segments
* 4    = bytes/Segment Table Entry
----
  44
```

Remember, the above multiplication is in hex.

```
7FFCØØ = Virtual beginning of Segment Table.
+ 44   = displacement in SGT for segment 11.
------
7FFC44 = Virtual address of Segment Table
         Entry we wish to find.
```

3. Turn to the Virtual Address of the Segment Table Entry we wish to find. The Segment Table Entry may be decoded using the Debugging Handbook, Volume 3 under label /SGTE.

Example:

FØ1539FØ ← invalid bit

The last digit must be zero.

this value:

1539FØ

This address is the real address of the Page Table.

4. Convert the Real address of the Page Table into its corresponding virtual address. This results in the beginning address of the Page Table.

5. Each entry in the Page Table is 2 bytes. Thus, to find the entry we need, multiply the Page Number contained in the original virtual address by 2, and add that value to the beginning virtual address of the Page Table.

Example:

11ØBCØ = Virtual address we wish to convert
↑ Page number
↳ Segment number

Ø = Page Number
2 = Bytes per Page Table Entry
Ø = Offset from beginning of Page Table.

7FC9FØ = Virtual address beginning of Page Table.
Ø = Offset from beginning of Page Table.
7FC9FØ = Page Table entry we desire.

6. Turn to Page Table Entry. This may be decoded using Volume 3 of the Debugging Handbook, under the label PGTE.

Example:

471ØØØ8
↑ Validity Flag
↳ Three most significant digits of real storage address described by this page table entry.

7. Concatenate the high-order three digits located in the Page Table with the low-order three digits contained in the virtual address we wish to convert.

Example:

11ØBCØ = Virtual address we wish to translate.
471 = High Order digits from Page Table.

471BCØ = Corresponding Real Address.

7-19-83

Initialization

Sysgen - the process of tailoring an MVS system to the hardware environment and the management objectives. (Mostly describing I/O devices). A SYSRES is the output of the sysgen process.
In Stage 1, sysgen macros will describe I/O devices (units, channels, controllers) and system data sets.
(This is explained in System Programmer's Library: Sysgen Reference, GC26-3792)
These macros are then assembled.
During assembly, the Distribution Library (DLIB) may be access. It holds object code for MVS. The output is an object module which is the stage two input.
During Stage 2, the object deck from stage 1 is link-edited to the object and load modules contained in the DLIB.

Summary: Stage one collects all information which is used to access the appropriate modules in stage 2.

Initialization begins by pressing the load button, and ends when MVS is ready to process jobs. Initialization has three steps:

1. IPL Program
2. NIP
3. Master Scheduler Initialization

The IPL pack is selected with the Load Unit Address dials. (In CE mode on 470, IPL can be software initiated).
IPL is a hardware function. The load unit is usually DASD, but card readers and tape units may also be used. Standalone utilities and system dump programs may also be IPLed.
For a brand-new system, stand-alone utilities may be used.

IPL Operation -

At the beginning of IPL, memory holds all zeros. Instead, hardware uses an implied CAW (hardwired) =

key = Ø, Read of 24 bytes (implied ccw)

to read from the load unit the contents of Cylinder Ø, track Ø, & record 1. (or the first 24 bytes of tape, etc.). The I/O device will then signal Channel End in CSW in hex location 4Ø.

The first 24 bytes is the IPL text: one PSW and two CCWs. This is record 1, of cyl Ø, Trk Ø.

1st doubleword = PSW
2nd doubleword = CCW1
3rd doubleword = CCW2

When the load button is pushed the R/W heads will position over Cyl Ø, Trk Ø.

CCW1 = Ø6ØØ3A98 4ØØØØØ5Ø

- Ø6 = command (Ø6 = Read Data)
- 003A98 = data store location
- 40 = another CCW follows.
- 0005Ø = data length

CCW2 = Ø8ØØ3A98 ØØØØØØØØ

- Ø8 = command (branch = Ø8)
- 003A98 = Branch to this address.

CCW1 loads the bootstrap from Record 2. CCW2 says branch to the routine just loaded. The bootstrap routine will load the IPL program. The bootstrap is a channel program.

See workbook, p. OA5ØØ2Ø.

Record 3 is the volser.

Record 4 is the IPL Program. (IEAIPLØØ)

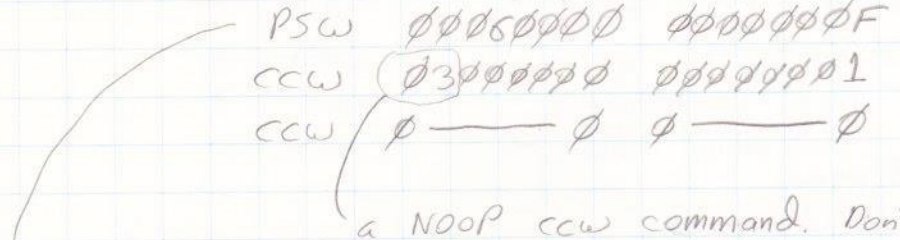
Bootstrap

addr.			
3A98	07003AB8	40000006	Positron R/W head to Cyl. Ø, Trk Ø
3AAØ	31003ABE	40000005	Search for identifier
3AA8	08003AAØ	Ø —— Ø	Branch to 3AAØ until found
3ABØ	06000000	20001960	Read
3AB8			
3ACØ			

The IPL program clears GPR and FPR, finds top of storage and clears storage, relocates himself to high core, and loads NIP (IEAVNIPØ).

No DAT occurs during the first portion of IPL. For problems around this time be sure to check bit 12 of the PSW to see if DAT is on. If the PSW indicates BC mode, note that the interrupt code is contained in the PSW.

Every pack has an IPL text, but not every pack has a valid IPL text. Thus, on a normal data pack, the IPL text contains



The machine will simply go into a wait state if you try to IPL from a data pack. The wait state PSW will be this PSW. A 'ØØF' wait state code means "An IPLed Volume does not contain the IPL text." (Debugging Handbook, vol. 1, p. 4-6).

Nucleus Initialization Program

The IPL program loads NIP and passes control to it. NIP will check the type and state of hardware, presence of DAS and DAT, store clock, open system data sets and page data sets, and build tables.

NIP builds a UCB for SYSRES and prepares for lots of I/O. Also, it builds a DEB for SYS1. Nucleus.

Initializes COMTASK for console communications.

Opens Logrec data set.
" SVCLIB.
" LINKLIB.
" Master Catalog.
" Parmlib

Obtains space in Nucleus for Page Frame Table. (Also checks every frame for errors and marks them offline if found).

Checks to see if CLPA was issued.

Checks if optional Fixed BLDL was specified.

Initializes CVT.

See Debugging Handbook, Vol 1, p. 5-85

The first 24 bytes is the IPL text: one PSW and two CCWs. This is record 1, of cyl Ø, Trk Ø.

1st doubleword = PSW
2nd doubleword = CCW1
3rd doubleword = CCW2

When the load button is pushed the R/W heads will position over Cyl Ø, Trk Ø.

CCW1 = Ø6ØØ3A98 6ØØØØØ5Ø

Ø6 = command (Ø6 = Read Data)
3A98 = data store location
6 = another CCW follows.
ØØ5Ø = data length

CCW2 = Ø8ØØ3A98 ØØØØØØØØ

Ø8 = command (branch = Ø8)
3A98 = Branch to this address.

CCW1 loads the bootstrap from Record 2. CCW2 says branch to the routine just loaded. The bootstrap routine will load the IPL program. The bootstrap is a channel program.

See workbook, p. DA5ØØ2Ø.

Record 3 is the volser.

Record 4 is the IPL Program. (IEAIPLØØ)

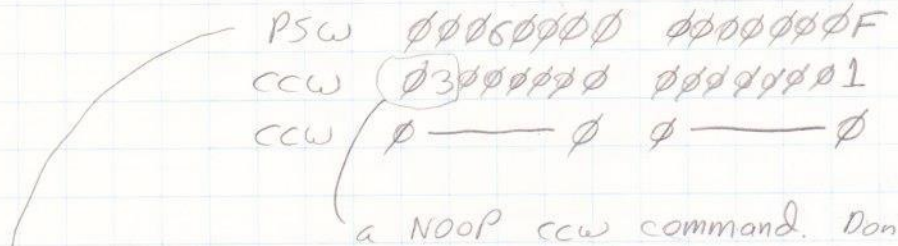
Bootstrap

addr.			
3A98	07003AB8	40000006	Positron R/W head to Cyl. Ø, Trk Ø
3AAØ	31003ABE	40000005	Search for identifier
3AA8	08003AAØ	Ø —— Ø	Branch to 3AAØ until found
3ABØ	06000000	20001960	Read
3AB8			
3ACØ			

The IPL program clears GPR and FPR, finds top of storage and clears storage, relocates himself to high core, and loads NIP (IEAVNPØ).

No DAT occurs during the first portion of IPL. For problems around this time be sure to check bit 12 of the PSW to see if DAT is on. If the PSW indicates BC mode, note that the interrupt code is contained in the PSW.

Every pack has an IPL text, but not every pack has a valid IPL text. Thus, on a normal data pack, the IPL text contains



The machine will simply go into a wait state if you try to IPL from a data pack. The wait state PSW will be this PSW. A 'ØØF' wait state code means "An IPLed Volume does not contain the IPL text." (Debugging Handbook, vol. 1, p. 4-6).

Nucleus Initialization Program

The IPL program loads NIP and passes control to it. NIP will check the type and state of hardware, presence of DAS and DAT, store clock, open system data sets and page data sets, and build tables.

NIP builds a UCB for SYSRES and prepares for lots of I/O. Also, it builds a DEB for SYS1.Nucleus.

Initializes COMTASK for console communications.

Opens Logrec data set.
" SVCLIB.
" LINKLIB.
" Master Catalog.
" Parmlib

Obtains space in Nucleus for Page Frame Table. (Also checks every frame for errors and marks them offline if found).

Checks to see if CLPA was issued.

Checks if optional Fixed BLDL was specified.

Initializes CVT.

See Debugging Handbook, Vol 1, p. 5-85

A Resource Initialization Module exists for each major MVS component:

	IOS	– IEAVNP02
catalog	VSAM	– IEAVNP11
	ASM	– IEAVNP04

NIP checks SYS1.Parmlib for settings of:

Size of SQA.
Size of CSA.
Names of Link Libraries.
Names of Authorized Libraries.
Dump Specifications for SYSUDUMP and SYSABEND dumps.
Names of all Page and swap datasets.

Some parameters may be overridden with console input during NIP execution.

Authorized Program Facility

A list of programs permitted to run in key 0. Member IEAAPFxx in PARMLIB contains list of programs. Once a program is on the list, it may be linkedited with an AC of 1.

Also, items in member CNKCISTxx are authorized.

When an IPL is performed with CLPA, the copy of the LPA on the LPA Page Data Set is rewritten. Otherwise it is not. If the LPA Page Data Set is not identical to the real LPA problems may occur.

Master Scheduler Initialization

Master scheduler has ASID 0. It is the first created. The Master Scheduler loads SWA management routines, and subsystem interface control blocks (for JES, IMS, etc).

Complete initialization of COMMTASK.

The Master Scheduler receives control from IEANIPx, the last module in NIP.

A Master Scheduler subsystem will substitute for JES until JES gets going. It starts the first seven address spaces. Then it looks in Sys1.Linklib for member MSTRJCL for the job entry subsystem (JES2 or JES3, usually). Prior to MVS 1.3, all system subsystems were kept in one address space. Now each has its own.

ASID = 1 — 2. PC/Auth - XMS (Cross Memory Services, new with 1.3)
= 2 — 3. GRS - Global Resource Serialization (hold Enq/Deque)
= 3 — 4. Dump SRV - Dump Services controls SVC dumps.
= 4 — 5. Comm Task - communicates with console.
= 5 — 6. Allocas - handles allocations.

Address Spaces for system modules are created by the IEEMB881 program.

NIP will analyze the resources needed by PC/Auth, GRS, and Dump SRV prior to passing control to Master Scheduler and the actual creation of these address spaces.

The Allocas module can now display (D, U, DASD, ALLOC) allocated devices and the job name with the allocation.

Once JES is up and running, the IPL is complete.

Exercise: Convert virtual addresses:
7FD230
FEØ714
7E3686

Job Management

7-20-83

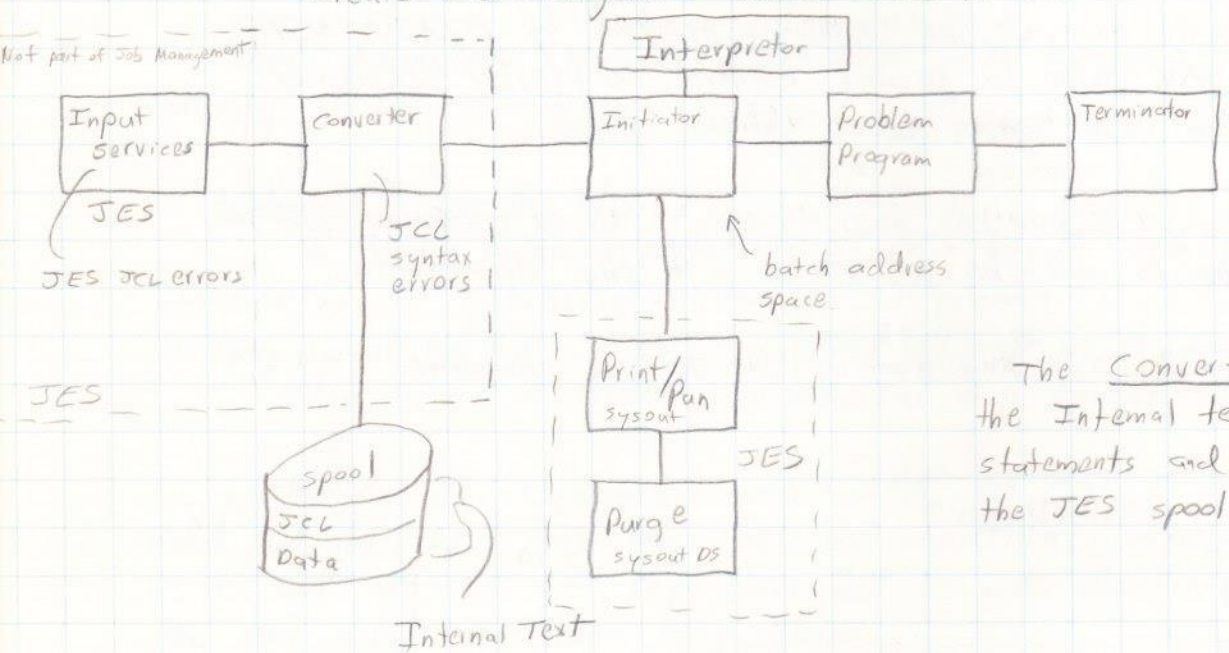
Job Management CBs live in SWA:

ACT
JCT
SCT
JFCB
SIOT

The SWA is in the Private Area. The information contained in the JCL stream is manipulated into Control Blocks by Job Management.

Job Management:

Handles Allocation services - volume attributes.
Handles SVC99 - dynamic allocation.
Job Scheduler Restart - checkpoint restart after S2F3.
Assigns Special Program Properties via PPT.
Create Job Management Control blocks in SWA.



The Converter builds the Internal text from JCL statements and places it on the JES spool.

The initiator will check with JES for any work of the appropriate class. The initiator requests work from JES. If work exists, the initiator will call the interpreter to take the internal text and convert it into SWA control blocks. (This is called building the Scheduler Control Blocks, SCBs). The initiator also initializes the Subsystem Interface (SSI) to communicate between JES and MVS; will allocate units, volumes, and datasets using ENQUE macros to serialize; assign special properties via the PPT; and issue an ATTACH to the problem program (via the TCB).

The terminator is invoked at step end and job end.

The Terminator will undo the set-up work of the initiator. It will:

Issue DEQueues to deallocate resources.

In a multistep job, a dataset may be allocated FREE=CLOSE which will cause a DEQ to be issued as soon as the data set is closed. This type of DEQ is not handled by the Terminator routine.

Updates disposition of data sets.

Will DEQ sysout datasets on spool volume so they may be printed by JES.

Will take condition code from R15 and store it in the Step Control Table (SCT).

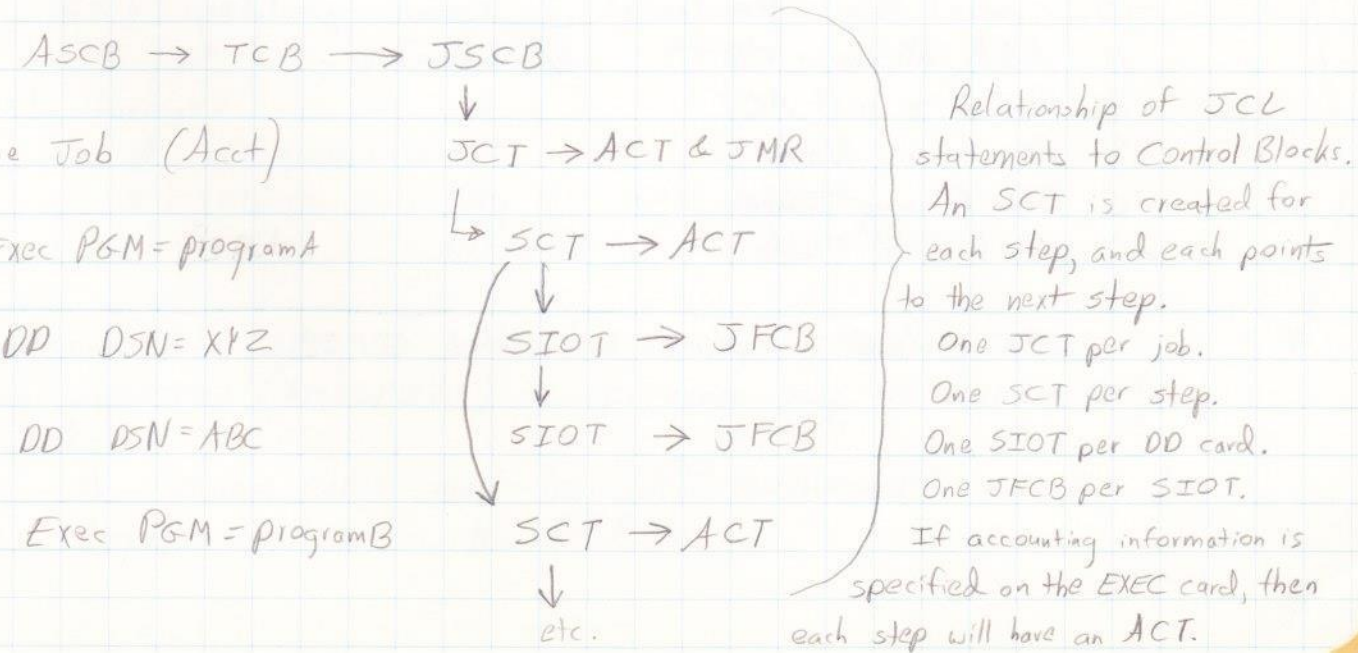
A Job Entry Subsystem is required for MVS operation. The JES communicates through SSI. A table in SYS1.LINKLIB, IEFJSSNT defines all secondary subsystems. The primary subsystem is designated at Sysgen. Thus, more than one subsystem can reside in the same MVS.

JES2 is based on HASP
JES3 is based on ASP.

Each is quite different from the other. JES console messages begin with '$'.

Job Management Control Blocks (see Debugging Handbook, vol. 1, p. 6-5)

The owning control block in an address space is the ASCB. It points the the TCB. At TCB+B5 is a pointer to the JSCB. The JSCB points to the JCT, which points to the ACT and JMR.



Job Control Table (JCT)

Jobname
Pointers to other SCB's.
Job Statement Parameters
- Msgclass
- Msg level
- Restart capability
- DPRTY

All Job Management CB's point to each other at -10 bytes. Thus, add X'10' to each pointer. Also, the pointer address is left justified in its half word. (See JM0080)

Control Block Chaining

ASCB + 6C	points to ASXB
ASXB + 4	points to TCB.
TCB + B4	points to JSCB.
JSCB + 104	points to JCT. (addr. + X'10')
JCT + 20	points to first SCT (addr. + X'10')
JCT + 28	points to first ACT (addr + X'10')
SCT + C	points to SIOT (addr. + X'10')
SCT + '14'	points to next SCT (addr. + X'10')
SIOT + 1C	points to next SIOT (addr. + X'10')

Control block fields are filled in based on JCL statements and system defaults.

ACT (one per JCT + SCT)

-10	Control Block Header
0	Addr. of this CB
4	Pointer to next ACT
8	Programmer's Name
1C	Job Running Time
20	Acct Data Fields

X'01' — padding

SIOT + '20' points to related JFCB

(See Workbook, p. JOBØ31Ø)

SCT - Step Control Table

Holds information from Exec statement:

- Maximum running time
- condition code (return code) - supplied by RTM at step end.
- Proc step name
- step name
- Program Name

SIOT

One per DD card; Does not include DCB information:
Device type, # Volumes, UCB address, DDName, and pointer to JFCB.

JFCB

Holds DCB information for each DD; DSN, Creation + Exp dates, + volsers of first five volumes.

Note: Pay attention to addresses of various control blocks to get a feel for where they live in storage - SWA, PLPA, etc.

Program Properties Table

- determines swappability
- allows dataset integrity bypass
- controls time parameter bypass
- controls operator cancel authority

The name of the program is entered in the PPT, not job or PROC name. The PPT is a DSECT, built by SGIEFOPT at Sysgen, + linked to IEFSDØ6Ø (the initiator).

Each entry is decimal 16 bytes.
The last entry is followed by FFØ——Ø

Entry format:

Ø	program name		
8	Flags	Key	CPU Affinity
C	Flags	Reserved	

Straight from IBM, the table has 5 dummy entries. Thus, using Superzap (AMASPZAP) a program may be dynamically added.

A Program name must be APF authorized in order to be eligible to live in the PPT.

Typical programs which may have PPT special characteristics:

- TCAM
- VTAM
- GTF
- Initiator
- Master Scheduler
- Mount
- JES
- IMS
- RMF
- CCW Trace (for SP1)
- ACF2

The PPT is simply a list of DC's.

(See Workbook, p. PPTØØ1Ø for examples)

MVS System Dispatcher (IEAVEDSØ)

7-21-83
Vic Michels

(See Special handouts for this topic)

The dispatcher has few bugs, but is often involved in problems. It's job is to give work to the CPU via the load PSW instructions. It selects the highest priority task or RB and gives it to the CPU.

SRBs are used for inter-address space communications. SRBs are both control blocks and executable code. The SRB is responsible for switching control from one address space to another.

The most common PSWs: Ø7ØCØ = 0000 0111 0000 1100 0000
(check these)
TIE KEY CMWP

Enabled loop:
- Enabled for interrupts
- condition code Ø
- supervisor state
- EC mode
- running

Also common: Ø78Ø = 0000 0111 1000 1101 0000

Enabled loop:
- problem state
- Enabled for interrupts
- Key = 8
- running

Ø7ØE = 0000 0111 0000 1110

Enabled wait:
- Enabled for interrupts & machine check
- Wait state
- Key = Ø
- Supervisor state

PSW — 040C = 0000 0100 0000 1100

Disabled loop:
- Disabled
- EC mode
- Key = 0
- Running

Happens after I/O or external interrupt.

PSW — 0002 or 000A = 0000 0000 0000 0010 1010

Disabled wait:
- Disabled wait
- BC mode or EC
- Probably Machine Check or IPL error.

The dispatcher is called by an Interrupt Handler - particularly the SVC handler. As soon as SVC handles a WAIT, the SVC turns back to the dispatcher. Also called by RTM when a task is completely terminated.

The dispatcher does not place ready work on its own queues. Other routines do this. The dispatcher simply scans its queues. To select the highest priority task, the dispatcher uses a fixed algorithm. (Tasks are guaranteed 70ms of run time before being interrupted. If an interrupt occurs, I/O for instance, the I/O interrupt handler (FLIH) will return to the running task until it accumulates 70 ms. This is the non-preemptable execution.)

Algorithm

1. Check special Exits - alternate CPU recovery option in AP and MP environments. ACR would be the highest priority if it occurs.
2. Check Global SRB - (Global SRBs access resources which are common to all address spaces. Does not care which address space is in control). Usually very short routines.
3. Check ready address spaces. If so, load the appropriate STO in CR 1. Memory is now switched.

PSAAOLD, PSAANEW > ASCB address used in address space dispatch. When 0, system is in SRB mode.

PSATOLD, PSATNEW will point to the TCB currently dispatched. If 0, then in SRB mode

   A. Once address space is in, check for local SRB queued to the address space.
   B. Then check for highest priority task within address space.
4. If none of the above, a dummy task is dispatched. PSW = 070E0 → 0
   An interrupt must occur to exit the wait.

Sometimes the dispatcher is working fine but no tasks are being placed on the dispatcher queues. Hit enter on the console, I/O interrupt occurs, system light flashes, & then back to wait state.

The ASCB lives in Common Area and is not swapped because it holds the Segment Table Origin. The ASCB points to its TCB. Also, it may have an ASXB (which lives in Private area in LSQA).

Each ASCB points to the next Ready ASCB and the previous ready ASCB. This is called "double-threading" and forms the queue through which the Dispatcher scans. Both are included so that an ASCB may be inserted. The system will have to know which previous ASCB to alter to point to the new entry. The ASID is a halfword field and thus the values must be reused. The SEQ. No. in the ASCB is the ASCBs dispatching order.
(see handout, DISP080)
The ASCB points to its Local SRBs and Ready TCBs. The AS Extension block points to the first and last TCBs, and holds the number of TCBs.

Learn to find the current ASCB.

MVS Tasks

An address space can have multiple tasks. They are created via an ATTACH macro and compete for CPU resources within the address space. The RCT is only active during swap in and out. However, it is the first task in an address space. Usually, in dumps, it is in a wait. Also, the Dump task will usually be in a wait. The RCT issues the attach to the initiator.

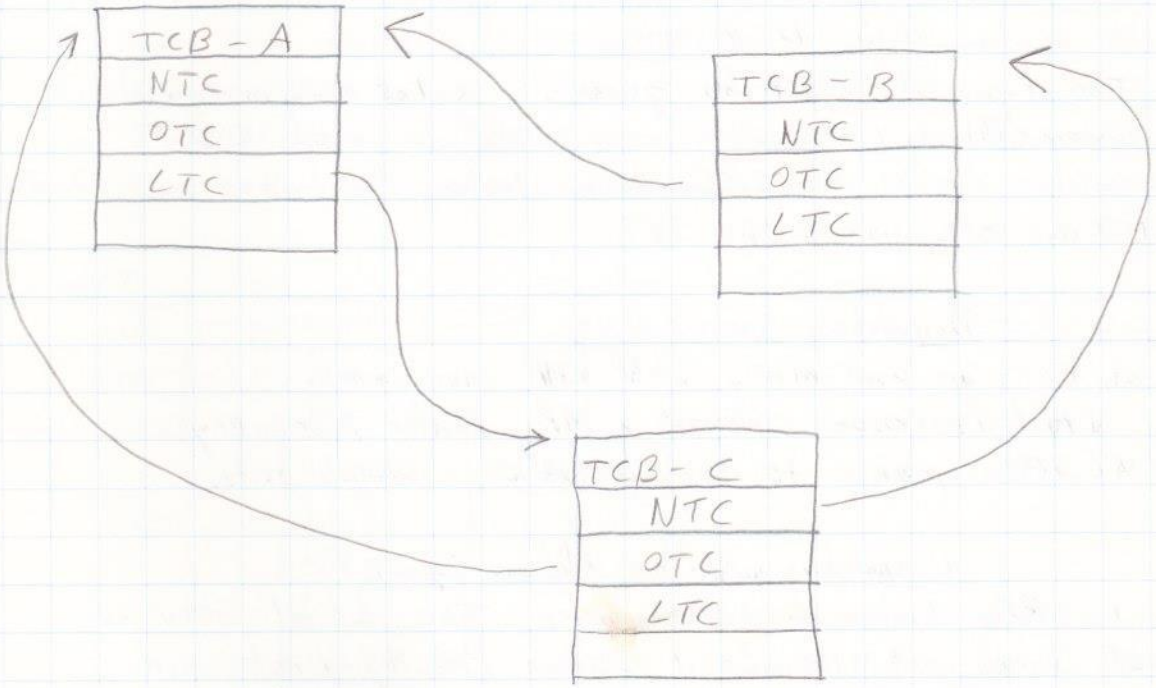
TCBs can each issue ATTACH. TCBs created by other TCBs are called daughter TCBs. They are single-threaded. (See Supervisor Services Macro Instructions and System Programming Library Supervisor [for very high-level macros]) Look up ATTACH.

Important TCB fields:

0	TCBRBP - pointer to RBs
10	CMP - completion code (usually contents of R1 from RTM; 80XXXYYY ← system cc, user cc)
1c	PKF - Protection key
	DSP - Dispatching Priority
	SCNDY - status
	JSCB - pointer to JSCB
	SEQNO - sequence number
	TCB - pointer to next ready TCB

There is no limit on the number of daughter TCBs. Three fields are used to chain TCBs:

TCBNTC - older sister
OTC - mother
LTC - daughter - youngest

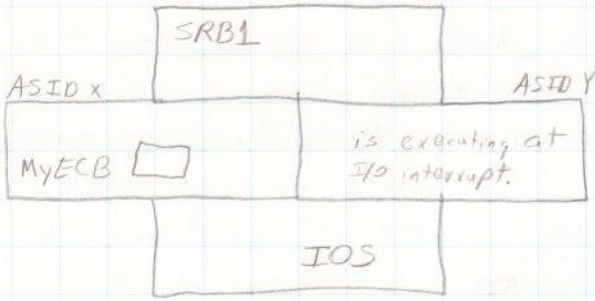


Each TCB points to its mother TCB and its older sister.
Each mother points to its youngest daughter.

Service Request Block Routines

SRBs are invoked by the SCHEDULE macro.
- Has pointer to next SRB.
- pointer to receiving ASCB.
- EPA of routine.
- EPA of parameter area to be passed.

Example of cross-memory post: IOS Post



The most common SRB is the communication of an I/O ending.

SRB Dispatching

Two types of dispatcher queues: Service Management Queue, and System Priority List. These are pointed to in the CVT. The dispatcher checks for SRBs before looking for tasks. The SMQ and SPL live in the SVT.

Dispatching Special Exits

Special Exits are used mostly in AP & MP environments. Also, VARY Processor command in MP, Timer Recovery. LCCAOSF1 point to the queue of special exits.

Dispatching within an Address Space

Local SRBs live in an Address Space. The Local Service Management Queue and the Local Service Priority List are check by the dispatcher.
- Local Lock Holders are dispatched first.
- Finally, ready tasks are dispatched.

Resource Control

(See OS/VS2 System Programming Library: Supervisor, pp. 13 - 65).

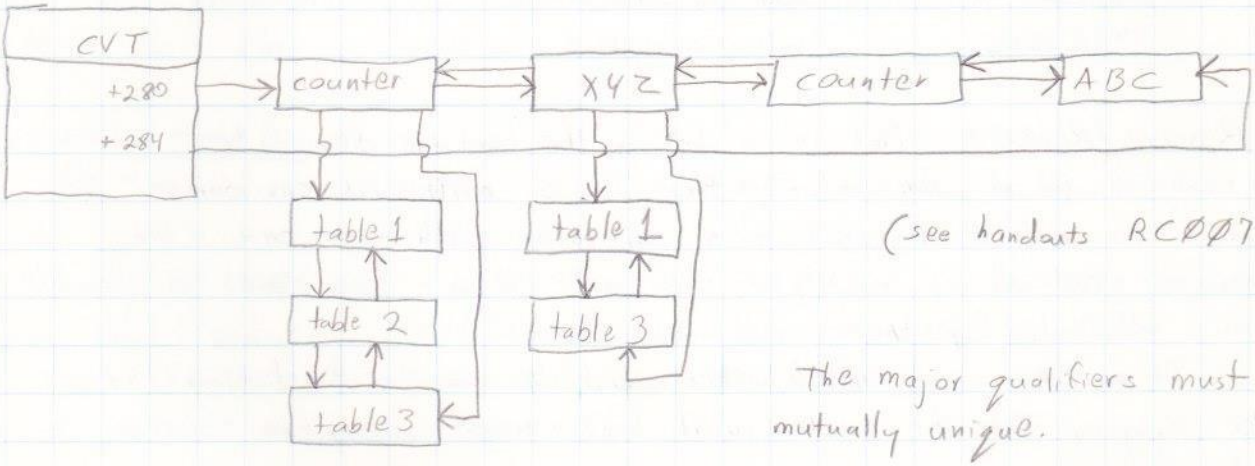
Covers: ENQ, DEQ
Reserve, Release (macros)
Locks (MP operation)

SRR - Serially Reuseable Resource.

Since SRRs can be accessed by many tasks, a method must control use. We must ensure serial access, not concurrent.
ENQ/DEQ is a convention. It is not absolutely mandatory.
The ENQ/DEQ itself runs disabled for interrupts.
A task issues an ENQ when it needs an SRR. When done, it issues a DEQ. If all tasks issue ENQ prior to using the resource then access is controlled.

Two types of ENQ: Exclusive and Share.
ENQs on resources are FIFO except for Shared enques. Many tasks may share a resource (read-only). Exclusive requests are serviced FIFO.
A Share with arrives <u>after</u> an Exclusive will wait on the Exclusive ENQ.

ENQ/DEQ is handled with a Queued Control Block (QCB).
Each resource has a major and minor name.



(see handouts RC0070)

The major qualifiers must be mutually unique.

Note that each node is double-threaded.
The major QCB is just a pointer to the minor name.

Queue Element List

Points to the TCB that is enqued.
Is chained off of Minor QCB.

ENQ Macro

QNAME = major name
RNAME = minor name
E = Exclusive
S = Shared
RNAME LENGTH = Length of minor name.
STEP = Resource use within the JOBSTEP
SYSTEM = Between address spaces
SYSTEMS = The entire system

To avoid deadlocks, always DEQ before issuing an ENQ.

See Debugging Handbook, Vol 1, Section 5 for a list of system major names.

Major	Minor
SYSDSN	dsname
SYSIEA01	IEA
SYSVTOC	volser
SYSIEWLP	dsn for SYSLMOD
SYSZVOLS	volser

Reserve/Release This is a bit in the hardware device that allocates a resource to a particular path. In a multi CPU environment this serves to restrict a unit to a particular path from one CPU. When the Release is received the unit again becomes accessible from all valid paths.

If a device is shared, this must be specified during the I/O Sysgen. Then the system will take care of issuing the Reserve and Release.

If a system crashes while reserving a device, that device will be unavailable to the other CPUs. (Press STOP on other CPU + reset the controller).

Global Resource Serialization

Scope = Systems.

An entire system is treated as an I/O device through a Channel-to-Channel-Adaptor (CTCA). CTCAs are called CCAs in Amdahl.

Tightly Coupled MPs

ENQ/DEQ will not work on tightly coupled MP.

Instead,

TS - Test & Set

Test byte for bit 1 state. If bit 1 is Ø, turn all bits on, also set condition code.

```
Tryagn    TS    SRRFLGA
          BNZ   Tryagn
          .
          .
          .
          MVI   SRRFLGA, Flgoff
```

TS, CS, & CDS have hardware interlocks which actually place a lock on a storage location. The the example above, SRRFLGA will be locked as soon as the TS gets into the pipeline of one of the processors. TS is faster than QCBs in UP environments. Note that this is still a convention.

Locks

A UCB is an SRR. At UCB-8 is the lockword for the particular UCB. (-4 points to the IOQ) Processors are usually X'4Ø' and X'41'. These CPUID codes are placed in the lockword. All zeros mean that the UCB is free. The SETLOCK macro executes through the one system dispatcher & uses the CS instruction.

Compare and Swap (CS)

B	A	R1	R3	B2	D2

Operand 1 and 2 are compared.

If =, operand 3 is stored in operand 2.

If ≠, operand 2 is stored in operand 1.

(See Handouts RCØØ23Ø)

Locks may have global or local scope. Globals control resources accessible by more than one address space. Local locks protect from competing TCBs.

A list of lockable resources is on RCØØ27Ø
A lock may also be either a SPIN or suspend type.
A lock that permits SPIN will permit the CPU to repeatedly execute CS until the resource becomes available – a tight loop. For a Suspend lock, the task requesting the resource is placed in a wait state.

MVS has a lock manager to handle the placing of locks. (Has bugs, too).

Locks are formed in a hierarchy. A task cannot request a lower lock than the one it has. This prevents deadlocks.

Spin locks-holders must be
- Global
- Disabled
- No page faults
- No SVCs

Suspend lock-holders
- Enabled/Disabled
- can page fault
- No SVCs

/ Cross Memory Services

Not all locks show CPUID. In the CMS lock, the lock holds the ASCB address
Local locks are held by task within an address space. Held in ASCB. lock contents:

X Ø = available
X '7FFFFFFF' = Suspend-Nondispatchable – to show the lock owner is currently non-dispatchable (i.e. Page fault)
X 'F — F' = Suspend Dispatchable
X '40' = CPU Ø
X '41' = CPU 1

The lock owner is suspended, but is dispatchable (ie: interrupt).

If a task abends while holding a lock, an FRR will release the lock.

The PSA+2FA hold the "highest Lock held Indicator" which helps preserve the hierarchy. (2 byte field) see RCØØ32Ø
Each lockable resource has a bit setting.
see Debugging 1 + look at SUC routines to see which locks are used for which services.

Cross Memory Services

This lock is a catch-all for many cross memory services. There is only one CMS lock + many cross memory services. If a particular service is tied to the lock, then the Address space must use the lock. Other ASIDs queue on the CMS lock.

The requestor for a CMS lock is usually an SRB routine, but can be a task.

The CMS suspend queue will have only one entry. If several SRBs or ASCBs are queued, the first one will point to the next, which will point to the third, etc.

To shoot lock dumps, go to the PSA Highest Lock Held Indicator (PSAHLHI) and decode which bits (locks) are one. Then, check the PSA or IEAVESLA for pointers to the locks.

In PSA:

- ASM
- IOSYNCH
- IOSCAT
- IOSUCB
- IOSLCH

} lockwords

CITA pointer → IEAVELIT

Dispatcher pointer → IEAVESLA

lockwords:
- DISP
- SALLOC
- SRM
- CMS

CMS suspend Q

Note: UCBs contain their lockword at -8. All UCBs are found in storage beginning on the first page after PSA.

Dispatcher + ENQ/DEQ will be on test.

Locks may have global or local scope. Globals control resources accessible by more than one address space. Local locks protect from competing TCBs.

A list of lockable resources is on RCØØ27Ø
A lock may also be either a SPIN or suspend type.
A lock that permits SPIN will permit the CPU to repeatedly execute CS until the resource becomes available – a tight loop. For a Suspend lock, the task requesting the resource is placed in a wait state.

MVS has a lock manager to handle the placing of locks. (Has bugs, too).

Locks are formed in a hierarchy. A task cannot request a lower lock than the one it has. This prevents deadlocks.

Spin locks-holders must be
- Global
- Disabled
- No page faults
- No SVCs

Suspend lock-holders
- Enabled/Disabled
- can page fault
- No SVCs

Not all locks show CPUID. In the CMS lock, the lock holds the ASCB address (Cross Memory Services)
Local locks are held by task within an address space. Held in ASCB. lock contents:

- X'Ø' = available
- X'7FFFFFFF' = Suspend-Nondispatchable – to show the lock owner is currently non-dispatchable (i.e. Page fault)
- X'F—F' = Suspend Dispatchable
- X'40' = CPU Ø
- X'41' = CPU 1

The lock owner is suspended, but is dispatchable (ie: interrupt).

If a task abends while holding a lock, an FRR will release the lock.

The PSA + 2FA hold the "highest Lock held Indicator" which helps preserve the hierarchy. (2 byte field) see RCØØ32Ø
Each lockable resource has a bit setting.
See Debugging 1 & look at SUC routines to see which locks are used for which services.

Cross Memory Services

This lock is a catch-all for many cross memory services. There is only one CMS lock + many cross memory services. If a particular service is tied to the lock, then the Address space must use the lock. Other ASIDs queue on the CMS lock.

The requestor for a CMS lock is usually an SRB routine, but can be a task.

The CMS suspend queue will have only one entry. If several SRBs or ASCBs are queued, the first one will point to the next, which will point to the third, etc.

To shoot lock dumps, go to the PSA Highest Lock Held Indicator (PSAHLHI) and decode which bits (locks) are one. Then, check the PSA or IEAVESLA for pointers to the locks.

In PSA:

- ASM
- IOSYNCH
- IOSCAT
- IOSUCB
- IOSLCH

} lockwords

CITA pointer → IEAVELIT

Dispatcher pointer → IEAVESLA

lockwords:
- DISP
- SALLOC
- SRM
- CMS
- CMS suspend Q

Note: UCBs contain their lockword at -8. All UCBs are found in storage beginning on the first page after PSA.

Dispatcher & ENQ/DEQ will be on test.

Locks may have global or local scope. Globals control resources accessible by more than one address space. Local locks protect from competing TCBs.

A list of lockable resources is on RC0027Ø
A lock may also be either a SPIN or suspend type.
A lock that permits SPIN will permit the CPU to repeatedly execute CS until the resource becomes available – a tight loop. For a Suspend lock, the task requesting the resource is placed in a wait state.

MVS has a lock manager to handle the placing of locks. (Has bugs, too).

Locks are formed in a hierarchy. A task cannot request a lower lock than the one it has. This prevents deadlocks.

Spin locks - holders must be
- Global
- Disabled
- No page faults
- No SVCs

Suspend lock-holders
- Enabled/Disabled
- can page fault
- No SVCs

Not all locks show CPUID. In the CMS lock, the lock holds the ASCB address (Cross Memory Services)
Local locks are held by task within an address space. Held in ASCB. lock contents:

X'Ø' = available
X'7FFFFFFF' = Suspend - Nondispatchable - to show the lock owner is currently non-dispatchable (i.e. Page fault)
X'F—F' = Suspend
X'40' = CPU Ø
X'41' = CPU 1

If a task abends while holding a lock, an FRR will release the lock.

Dispatchable
The lock owner is suspended, but is dispatchable (ie: interrupt).

The PSA+2FA hold the "highest Lock held Indicator" which helps preserve the hierarchy. (2 byte field) see RCØØ32Ø
Each lockable resource has a bit setting.
see Debugging 1 + look at SVC routines to see which locks are used for which services.

Cross Memory Services

This lock is a catch-all for many cross memory services. There is only one CMS lock + many cross memory services. If a particular service is tied to the lock, then the Address space must use the lock. Other ASIDs queue on the CMS lock.

The requestor for a CMS lock is usually an SRB routine, but can be a task.

The CMS suspend queue will have only one entry. If several SRBs or ASCBs are queued, the first one will point to the next, which will point to the third, etc.

To shoot lock dumps, go to the PSA Highest Lock Held Indicator (PSAHLHI) and decode which bits (locks) are one. Then, check the PSA or IEAVESLA for pointers to the locks.

In PSA:
- ASM
- IOSYNCH
- IOSCAT
- IOSUCB
- IOSLCH

} lockwords

CITA pointer → IEAVELIT

Dispatcher pointer → IEAVESLA

lockwords:
- DISP
- SALLOC
- SRM
- CMS

CMS suspend Q

Note: UCBs contain their lockword at -8. All UCBs are found in storage beginning on the first page after PSA.

Dispatcher + ENQ/DEQ will be on test.

7-22-83

SVC (See Debugging Handbook, Vol. 1, section 5)

Supervisor Call Op code is ØA.
The operand of the SVC is stored in the PSA beginning at 8A for 2 bytes.

The SVC is used to permit problem programs to have access to privileged instructions. It also saves coding since the SVCs are ready, available, and tested.

Some macros expand to SVC to make it easy to use the particular SVC. (Modeset, for example) The user may issue

```
SVC 107
```

but then would have to correctly code all the operands.
Moral: use the macro & don't screw up.
(See: OS/VS2 MVS Supervisor Services and Macro Instructions)

GETMAIN is an SVC which key 8 users may use.

```
        GETMAIN RC,LV=72
+       LA      Ø,72(Ø,Ø)
+       BAL     1,*+4       - turns on high order bit quick to signal GETMAIN
+       SVC     1Ø
```

macro expansion

The system completion code from an abending SVC contains the SVC type in the last byte:

S 8ØA = GETMAIN

Lots of problems involve SVC-related situations.

GETMAINs are a must for reentrant coding, since the "gotten" areas stay with the task.

Six types of SVCs, referring to type of environment.
1, 2, +6 Nuc resident (1, 2, +6 have some restrictions)
3 + 4 LPA
5 dummy entry

The First Level Interrupt Handler (FLIH) saves status of the requesting program, then determines the type of SVC, + obtains a lock if necessary.

Type 1 Cannot issue another SVC. Must have local lock.
2, 3, 4 Can issue recursive SVCs.
6 Cannot issue another SVC and will run disabled for I/O and External interrupts.

SVCs are not maskable because the program itself causes the interrupt. An SVC interrupt cannot come from the outside.

The SVC FLIH will save Regs into either the TCB or the SVRB. Types 1 and 6 SVC cause Regs to be saved in the TCB. Types 2, 3, + 4 save into (SVRB. the previous PRB or TCB

SVCs 1 + 6 run in PRBs.
SVCs 2, 3, + 4 run in SVRBs.

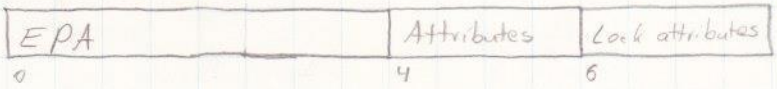
In dumps, check the current TCB of the active ASCB and the "active RBS" will show either an PRB or an SVRB in accord with the last SVC issued. (see Dump, Vol. I, p. 42). Checks regs to see paramaters passed by program to SVC.

For SVRB, check pointer to previous PRB or TCB to see where he saved registers. (see Dump, Vol. 1, p. 52)

FLIH will prevent SRBs from issuing SVCs. Lockholders may not issue SVCs. A user cannot be disabled for interrupts when calling SVCs.

SVC Table

The SVC table contains the list of routines for each SVC. The CVT field CVTABEND points to SCVT which has field SCVTSVCT (+84) which points to the SVC table. Each entry in the table is 8 bytes long. The SVC number * 8 gives the depth into the table.



The SVCTable may be used to branch directly into an SVC routine and thus avoid the FLIH checks. Once the address of the routine is found, go there and check the label (program name, assembly date, + PTF level).

Low storage can be browsed on TSO.

To flip a bit the mapping macro may be used. Flipping is done with an OR and labels. Always use labels because the location may change in the future. The labels will be kept consistent. The mapping macro will generate the DSECT (IHASVC for SVC table).

SVC Register Conventions: (FLIH sets these up.)

3	CVT Address
4	TCB Address
5	RB "
6	EPA "
7	ASCB "
14	EXIT " to Exit Prolog.
0, 1, 13, & 15	remain unchanged.

SVCs must be fixed to virtual addresses. This is because the SVCTable is built during sysgen & is not created dynamically during load.

Naming

Types 1, 2, & 6 begin with IGCxxx (xxx: decimal)
3 & 4 IGCyyxxx (yy: 00 for type 3, module # for type 4)

SVC routines must be reentrant and refreshable.

(See handouts, SVC0180)

The Exit prolog will free locks, free SVRB, Pass information via regs. 0, 1, & 15, and branches to the dispatcher or the calling program (if NP bit is on).

the unpacked format of the decimal form.
#122 is displayed as 12B

F1 F2 2C
↓ ↓ ↓
1 2 B

'2C' is an EBCDIC 'B'

Type 3 SVCs have only one load module. Type 4 SVCs have two or more modules.

An SVC routine can issue recursive SVCs provided that it does not already hold a lock.