z/OS Control Blocks and the Rexx Storage BIF

René Vincent Jansen, 35th International Rexx Language Symposium 2024, Brisbane, Queensland, Australia

Contents

1 **Why is this relevant**
Some very good reasons to look into this

2 **What is a Control Block**
The data part of an Operating System

3 **What can we do with them**
Explore, gain knowledge, understand and diagnose problems, build useful tools

4 **View Control Blocks: ISRDDN**
ISRDDN suprises again with useful functionality

5 **View Control Blocks: IPCS**
The old standby for OS analysis and dump formatting

6 **Program with the Rexx STORAGE built-in function**
Make you own useful and to-the-point programs

7 **Macro's and Assembler**
Make you own useful and to-the-point programs, and here the mapping is done for you!

8 **.. or even in COBOL**
Among other, how to find out if you are running under CICS or JES2

Ministerie van Financiën

№ 1 Why is this relevant

Relevant because

- Most performance monitor software reads these
- Can zoom in for specific investigations
- Can roll your own performance tool
- Know how the ASCB tool works

- Learning: by looking into the structure of the OS you will understand performance issues better

Ministerie van Financiën

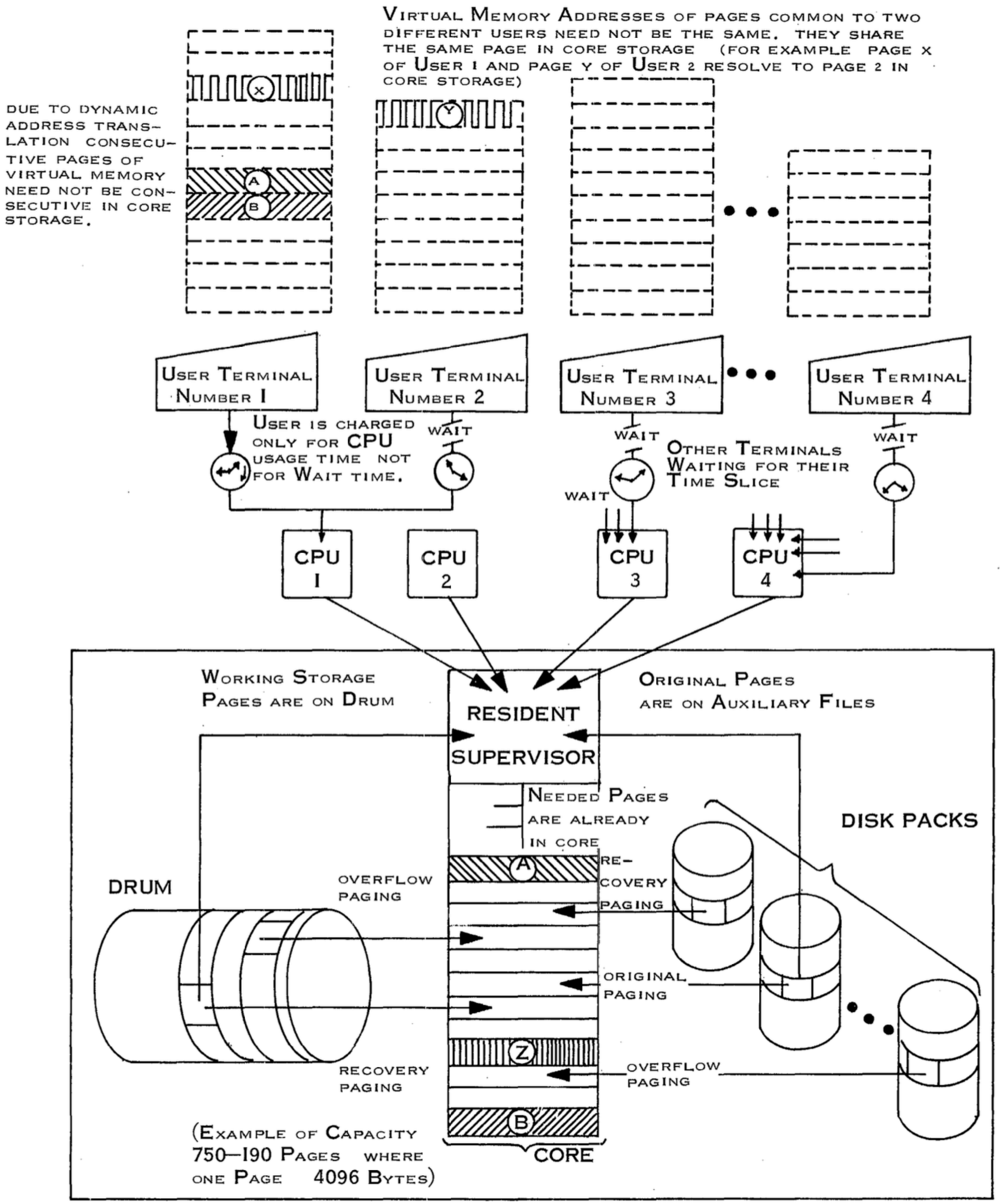
№ 2 What are Control Blocks

What is an operating system

- A Supervisor
- A Scheduler
- Utilities, loaders, linkers and compilers and other small fry
- The control blocks are the data areas (variables) of the supervisor and the scheduler
- Like JCL is the way to command the scheduler

What is Virtual Storage

- Illusion arranged by hardware and system software
- Every address space is 16MB (24bit), 2GB (31bit) or 18 ExaBytes (18 Quintillion bytes (64bit)
- A map divided in different areas, some do overlap
- z/OS has private and common areas
- Some common areas map to the same real storage
- (Different virtual addresses can even map to the same real address)


VIRTUAL MEMORY ADDRESSES OF PAGES COMMON TO TWO DIFFERENT USERS NEED NOT BE THE SAME. THEY SHARE THE SAME PAGE IN CORE STORAGE (FOR EXAMPLE PAGE X OF USER 1 AND PAGE Y OF USER 2 RESOLVE TO PAGE 2 IN CORE STORAGE)
DUE TO DYNAMIC ADDRESS TRANSLATION CONSECUTIVE PAGES OF VIRTUAL MEMORY NEED NOT BE CONSECUTIVE IN CORE STORAGE.
X
Y
A
B
USER TERMINAL NUMBER 1
USER TERMINAL NUMBER 2
USER TERMINAL NUMBER 3
USER TERMINAL NUMBER 4
USER IS CHARGED ONLY FOR CPU USAGE TIME NOT FOR WAIT TIME.
WAIT
WAIT
WAIT
WAIT
OTHER TERMINALS WAITING FOR THEIR TIME SLICE
CPU 1
CPU 2
CPU 3
CPU 4
WORKING STORAGE PAGES ARE ON DRUM
RESIDENT SUPERVISOR
ORIGINAL PAGES ARE ON AUXILIARY FILES
NEEDED PAGES ARE ALREADY IN CORE
DISK PACKS
DRUM
OVERFLOW PAGING
RE-COVERY PAGING
ORIGINAL PAGING
Z
OVERFLOW PAGING
RECOVERY PAGING
(EXAMPLE OF CAPACITY 750—190 PAGES WHERE ONE PAGE 4096 BYTES)
CORE

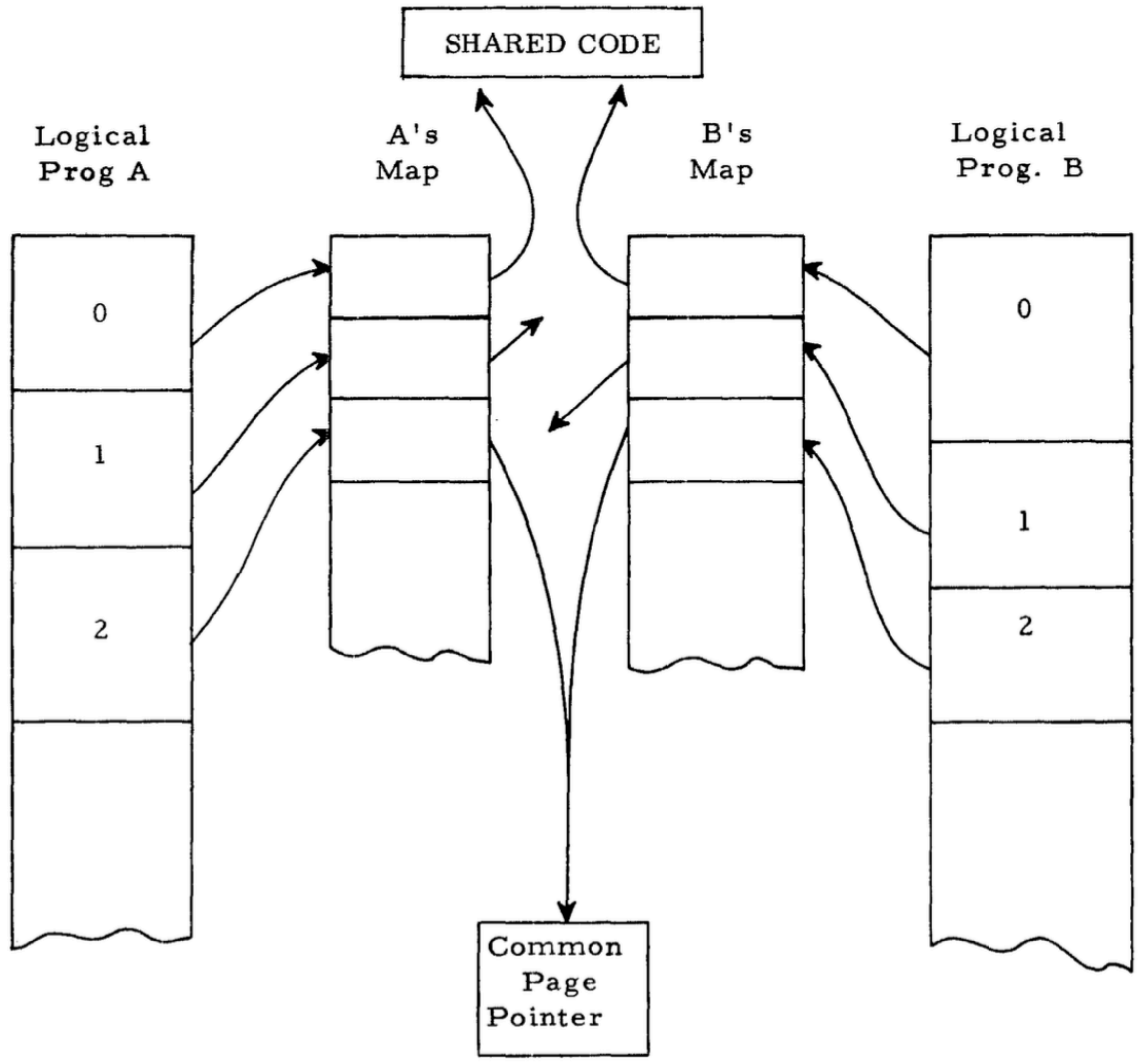
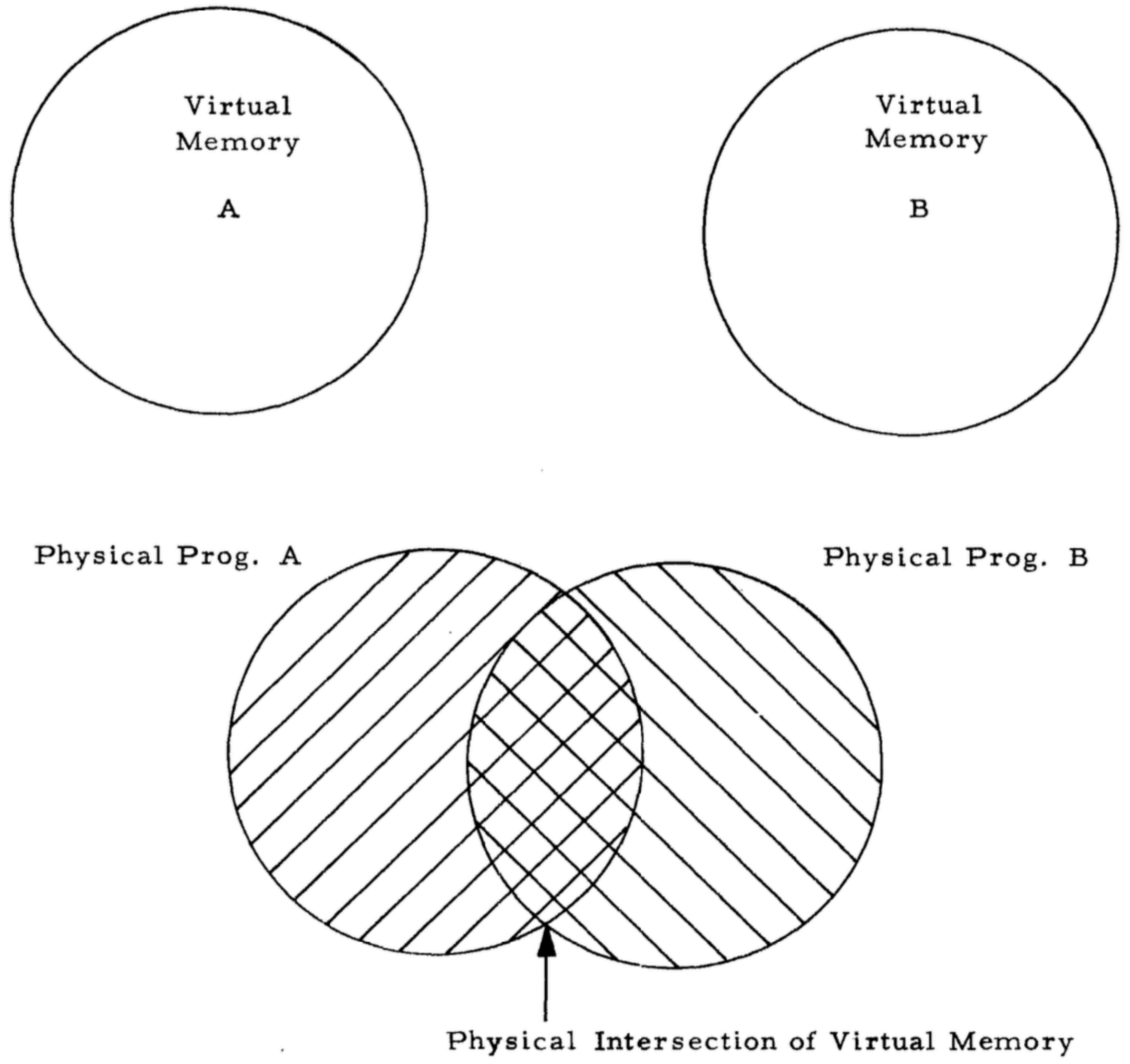
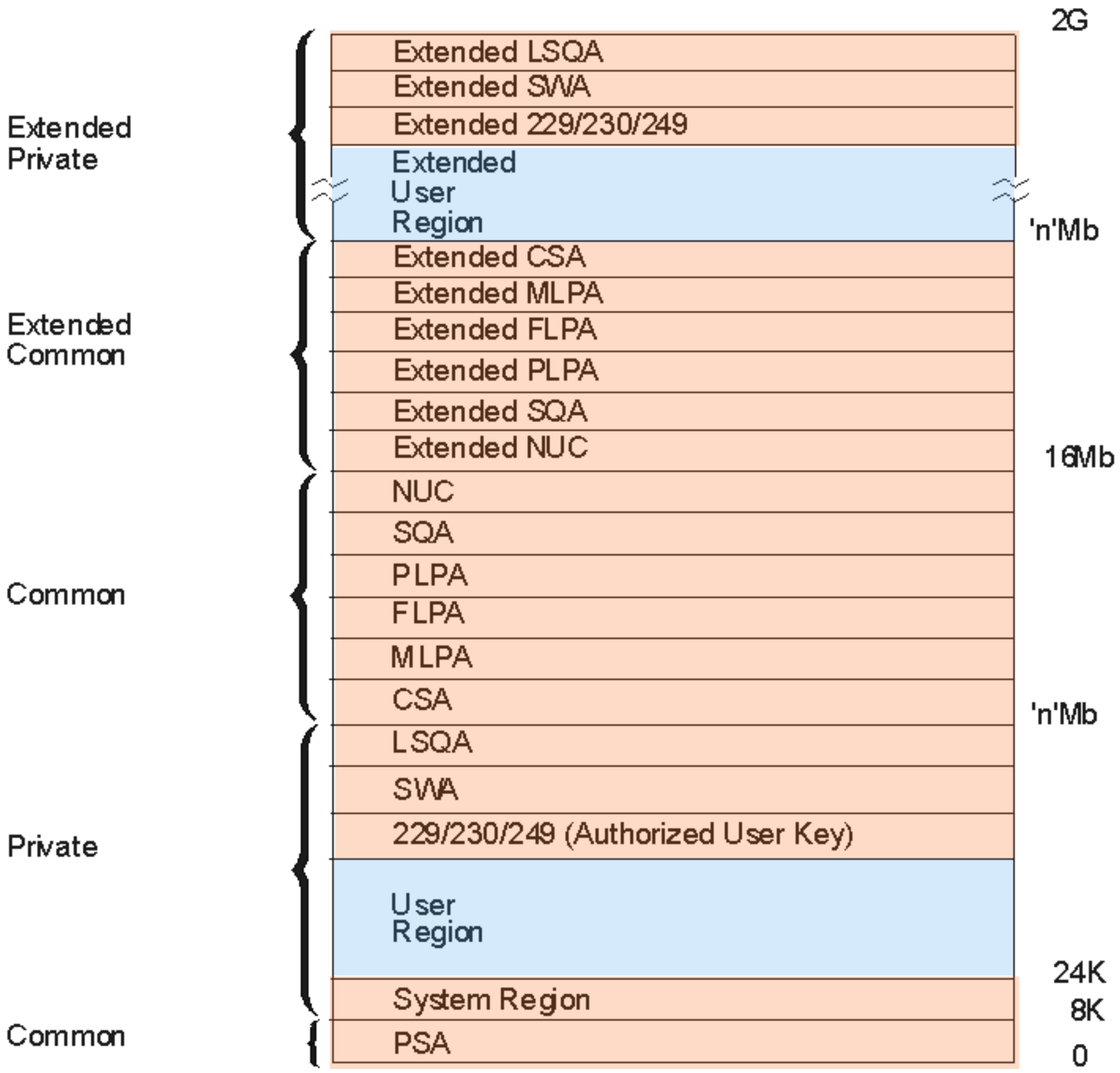


Figure 4. Example of shared code



31-bit memory map (MVS/XA)



2G

Extended Private

Extended LSQA

Extended SWA

Extended 229/230/249

Extended User Region

'n'Mb

Extended Common

Extended CSA

Extended MLPA

Extended FLPA

Extended PLPA

Extended SQA

Extended NUC

16Mb

Common

NUC

SQA

PLPA

FLPA

MLPA

CSA

'n'Mb

Private

LSQA

SWA

229/230/249 (Authorized User Key)

User Region

24K

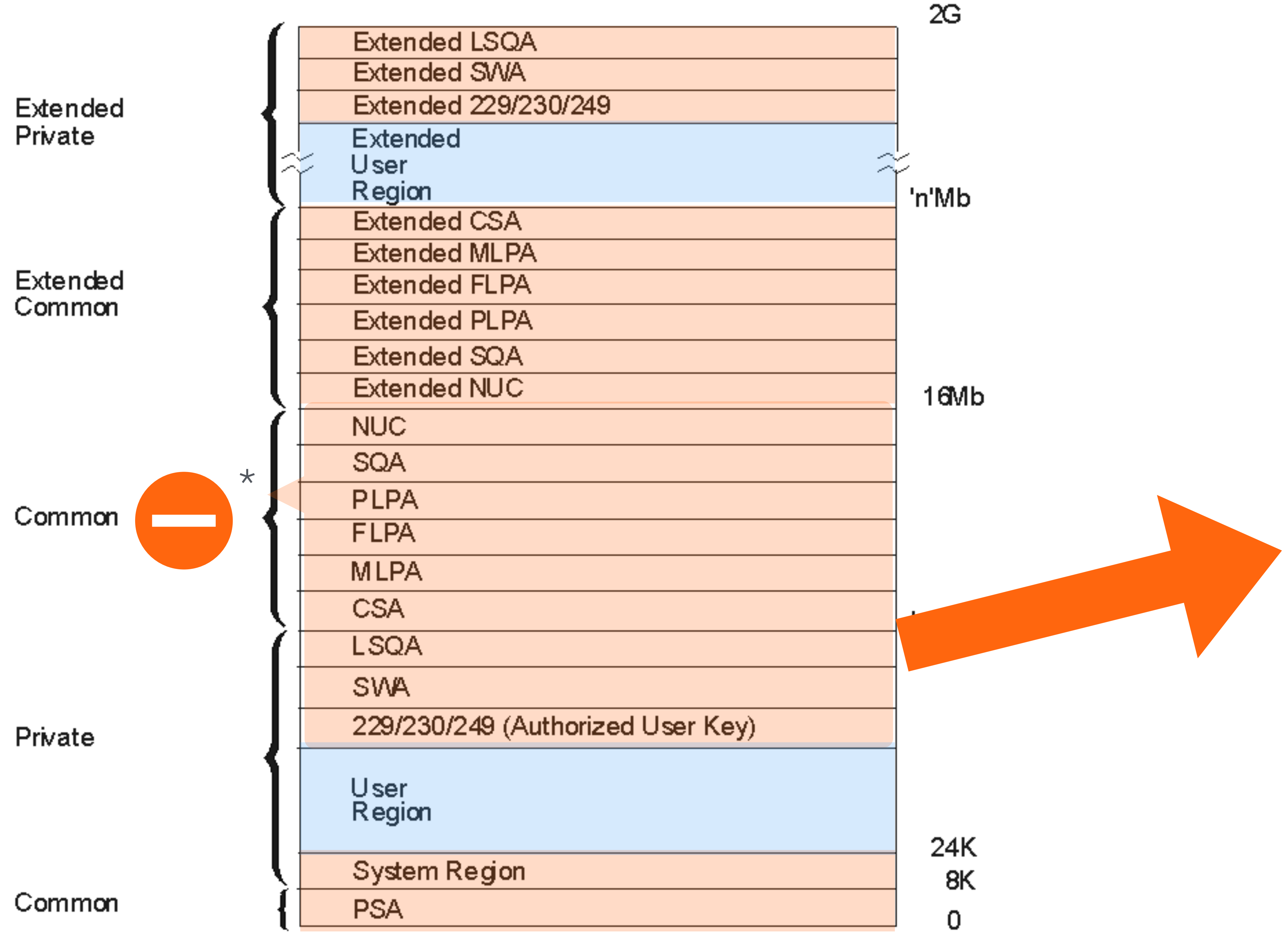
System Region

8K

Common

PSA

0



2G

Extended Private

Extended LSQA

Extended SWA

Extended 229/230/249

Extended User Region

'n'Mb

Extended Common

Extended CSA

Extended MLPA

Extended FLPA

Extended PLPA

Extended SQA

Extended NUC

16Mb

Common *

NUC

SQA

PLPA

FLPA

MLPA

CSA

Private

LSQA

SWA

229/230/249 (Authorized User Key)

User Region

24K

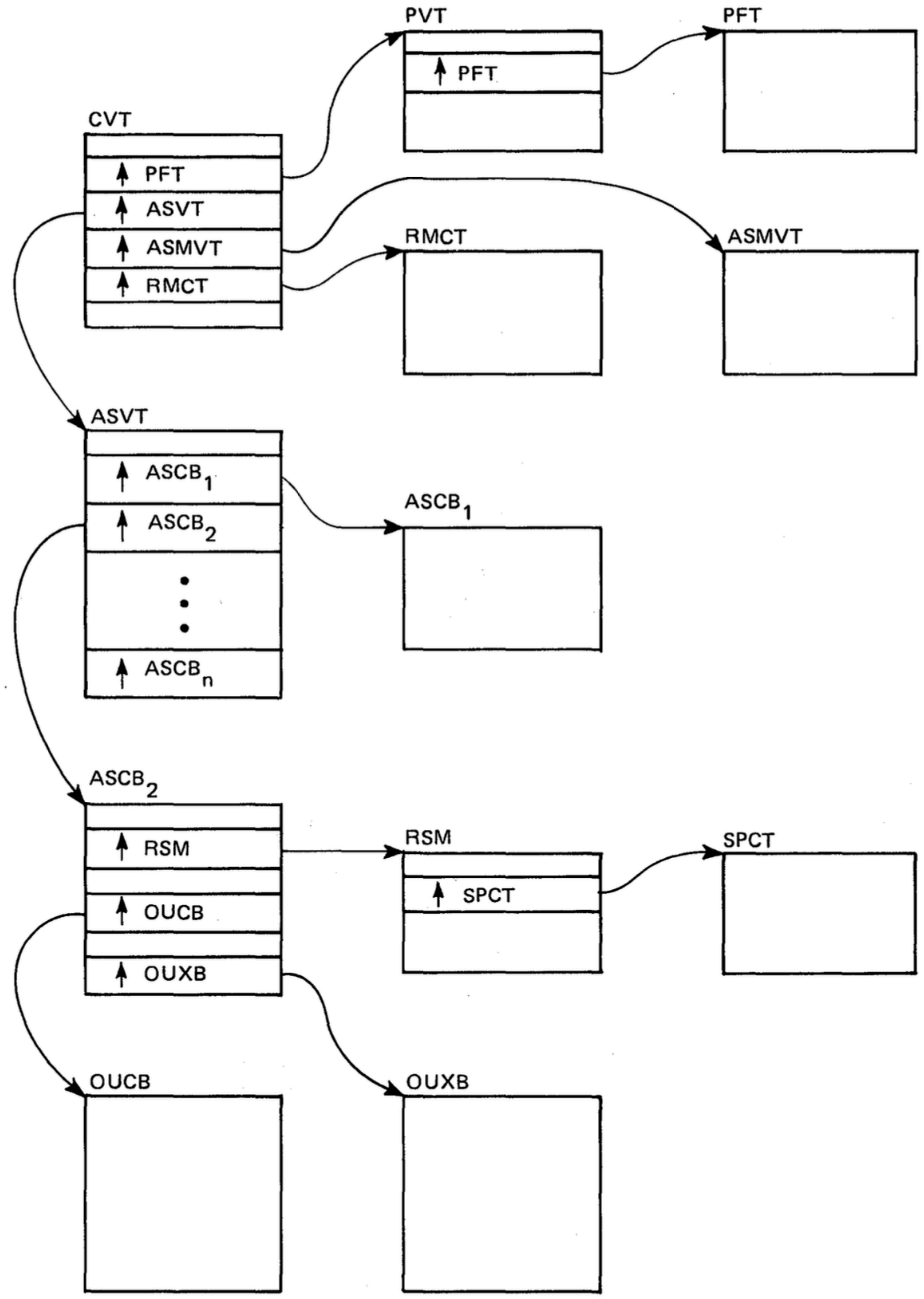
Common

System Region

8K

PSA

0



* write-protected

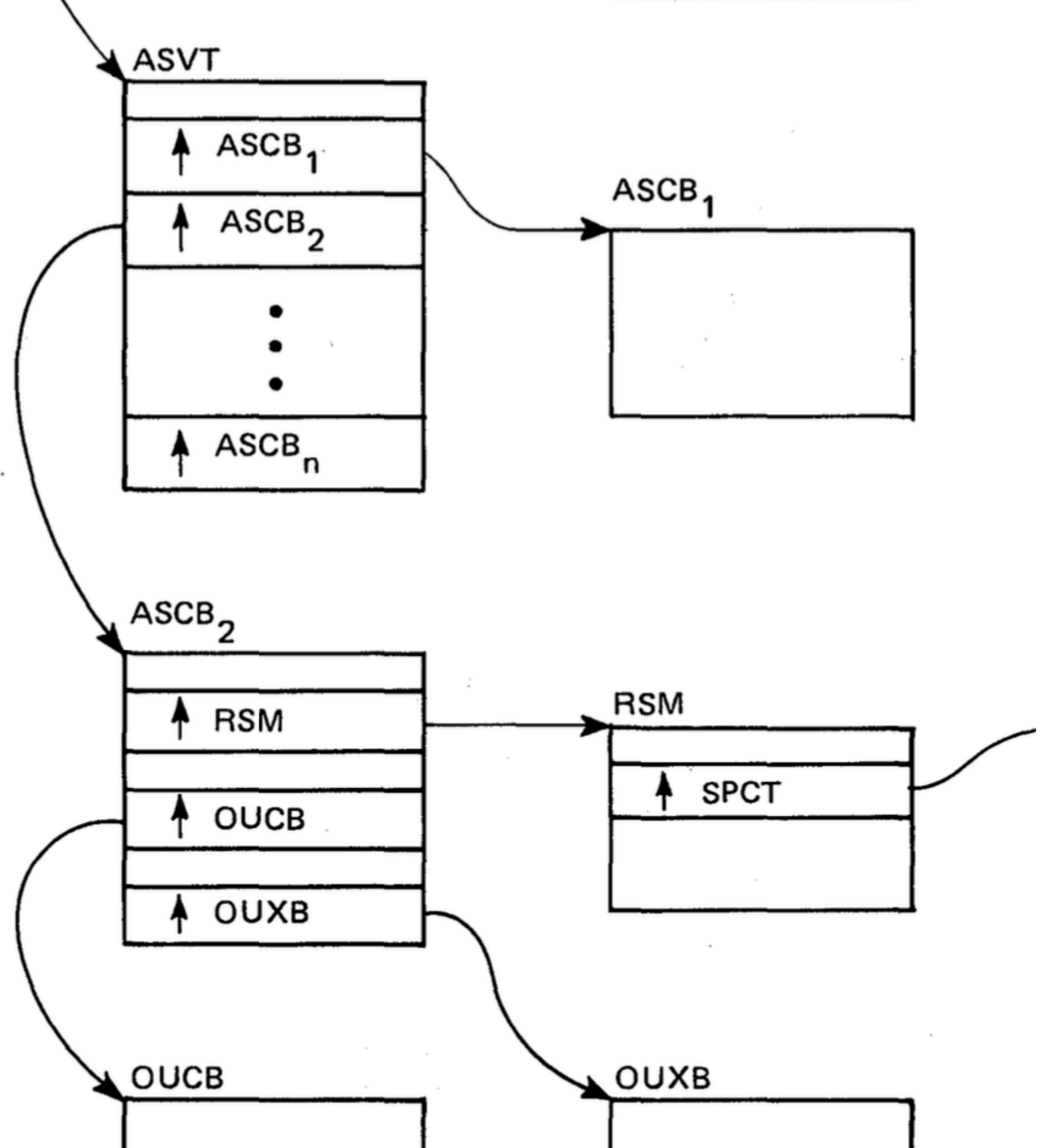
How does z/OS find programs?

When a program is requested through a system service (like LINK, LOAD, XCTL, or ATTACH) using default options, the system searches for it in the following sequence:

1. **Job pack area (JPA)** A program in JPA has already been loaded in the requesting address space. If the copy in JPA can be used, it will be used. Otherwise, the system either searches for a new copy or defers the request until the copy in JPA becomes available. (For example, the system defers a request until a previous caller is finished before reusing a serially-reusable module that is already in JPA.)
2. **TASKLIB** A program can allocate one or more data sets to a TASKLIB concatenation. Data sets concatenated to TASKLIB are searched for after JPA but before any specified STEPLIB or JOBLIB. Modules loaded by unauthorized tasks that are found in TASKLIB must be brought into private area virtual storage before they can run. Modules that have previously been loaded in common area virtual storage (LPA modules or those loaded by an authorized program into CSA) must be loaded into common area virtual storage before they can run. For more information about TASKLIB, see z/OS MVS Programming: Assembler Services Guide.
3. **STEPLIB or JOBLIB** STEPLIB and JOBLIB are specific DD names that can be used to allocate data sets to be searched ahead of the default system search order for programs. Data sets can be allocated to both the STEPLIB and JOBLIB concatenations in JCL or by a program using dynamic allocation. However, only one or the other will be searched for modules. If both STEPLIB and JOBLIB are allocated for a particular jobstep, the system searches STEPLIB and ignores JOBLIB. Any data sets concatenated to STEPLIB or JOBLIB will be searched after any TASKLIB but before LPA. Modules found in STEPLIB or JOBLIB must be brought into private area virtual storage before they can run. Modules that have previously been loaded in common area virtual storage (LPA modules or those loaded by an authorized program into CSA) must be loaded into common area virtual storage before they can run. For more information about JOBLIB and STEPLIB, see z/OS MVS JCL Reference.
4. **LPA**, which is searched in this order:
   - **Dynamic LPA** modules, as specified in **PROGxx** members
   - Fixed LPA (**FLPA**) modules, as specified in IEAFIXxx members
   - Modified LPA (**MLPA**) modules, as specified in IEALPAxx members
   - Pageable LPA (**PLPA**) modules, loaded from libraries specified in LPALSTxx or PROGxx
5. **LPA modules are loaded in common storage, shared by all address spaces in the system. Because these modules are reentrant and are not self-modifying, each can be used by any number of tasks in any number of address spaces at the same time. Modules found in LPA do not need to be brought into virtual storage, because they are already in virtual storage.**

6. Libraries in the **linklist,** as specified in PROGxx and LNKLSTxx. By default, the linklist begins with **SYS1.LINKLIB,** SYS1.MIGLIB, SYS1.CSSLIB, SYS1.SIEALNKE, and SYS1.SIEAMIGE. However, you can change this order using SYSLIB in PROGxx and add other libraries to the linklist concatenation. The system must bring modules found in the linklist into private area virtual storage before the programs can run.

Find program, look in:
JPA
TASKLIB
STEPLIB or JOBLIB
LPA
  Dynamic (PROGXX)
  FLPA
  MLPA
  PLPA
  Linklist (concatenation) (LLA, VLF cache)



Address Space Control Block – ASCB

The ASVT contains an entry for each potential address space. Each entry points to an ASCB, which contains job-related data. The following fields in the ASCB are of interest:

ASCBSEQN The sequence number of this ASCB on the dispatching queue. Valid only if the address space is currently swapped-in.

ASCBDP The current dispatching priority for this address space. Valid only if the address space is swapped-in.

ASCBEJST This doubleword (in time-of-day clock format) represents the total task time received by this address space.

ASCBSWCT Contains a count of the number of short waits issued by this address space. This value is used in the APG mean-time-to-wait calculation.

ASCBVSC Contains a count of the total number of VIO slots allocated within the page data sets for this address space.

ASCBNVSC Contains a count of the total number of non-VIO slots allocated within the page data sets to this address space.

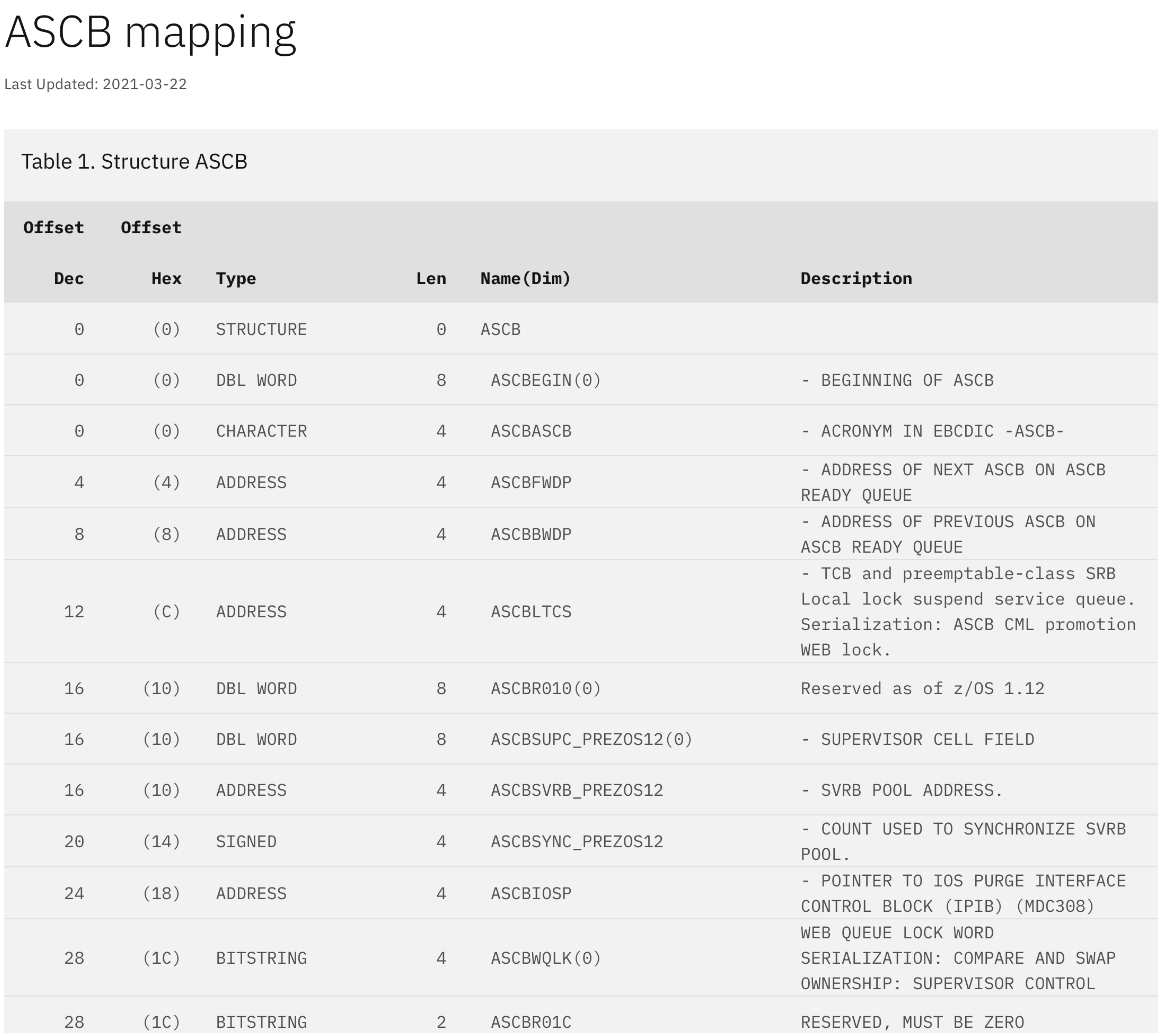
ASCBFMCT Contains a count of the number of real storage page frames currently occupied by this address space.

ASCBJBNI Contains a pointer to the 8-character jobname for a batch job. Zero if not a batch job.

ASCBJBNS Contains a pointer to the 8-character jobname for started tasks, mounts, and TSO users.

ASCBSRBT This doubleword (in time-of-day clock format) contains the SRB time accumulated by this address space.

Current day version (nearly unchanged) at https://www.ibm.com/docs/en/zos/2.2.0?topic=information-ascb-mapping



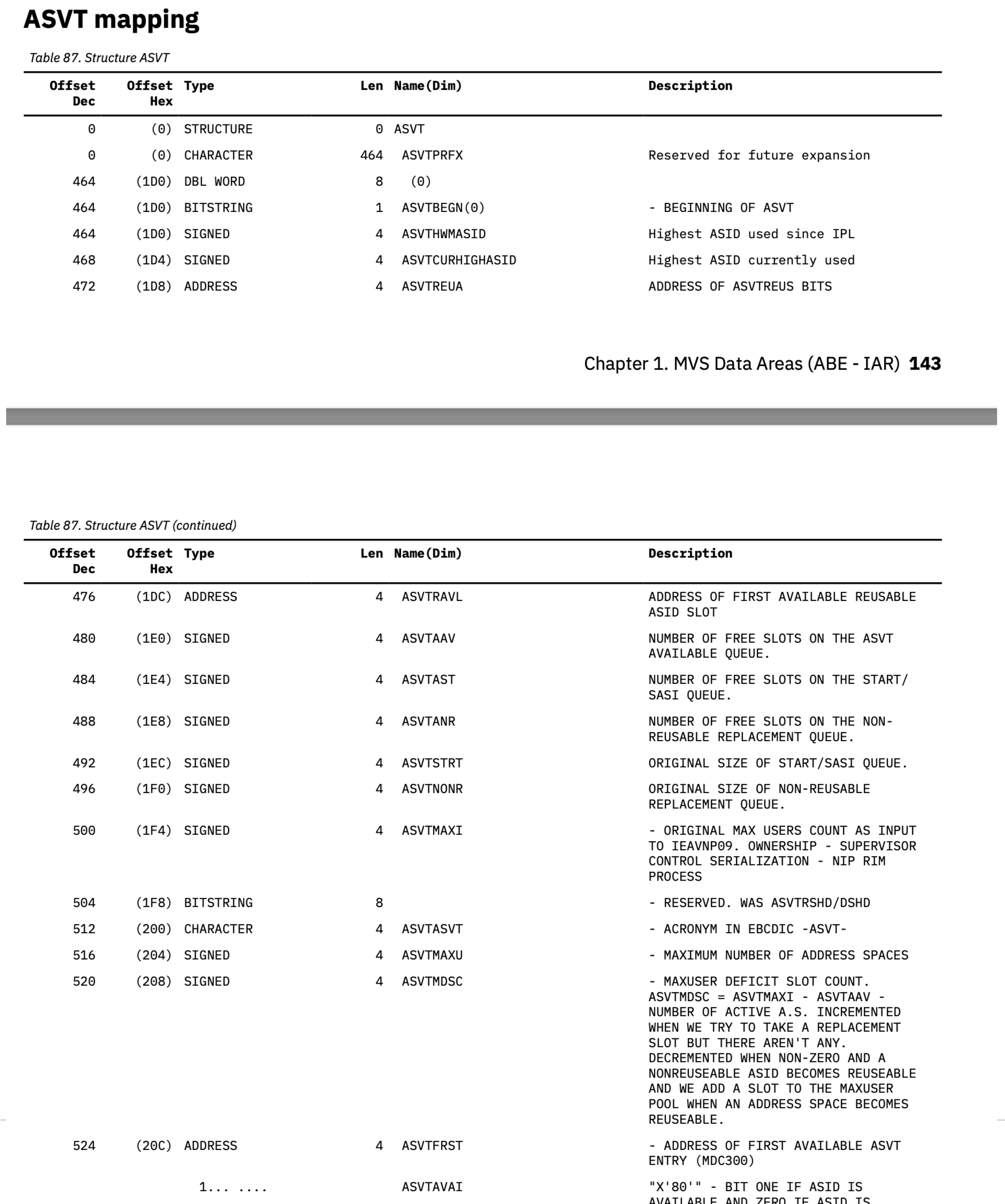
ASCB mapping

Last Updated: 2021-03-22

Table 1. Structure ASCB

Offset Dec	Offset Hex	Type	Len	Name(Dim)	Description
0	(0)	STRUCTURE	0	ASCB	
0	(0)	DBL WORD	8	ASCBEGIN(0)	- BEGINNING OF ASCB
0	(0)	CHARACTER	4	ASCBASCB	- ACRONYM IN EBCDIC -ASCB-
4	(4)	ADDRESS	4	ASCBFWDP	- ADDRESS OF NEXT ASCB ON ASCB READY QUEUE
8	(8)	ADDRESS	4	ASCBBWDP	- ADDRESS OF PREVIOUS ASCB ON ASCB READY QUEUE
12	(C)	ADDRESS	4	ASCBLTCS	- TCB and preemptable-class SRB Local lock suspend service queue. Serialization: ASCB CML promotion WEB lock.
16	(10)	DBL WORD	8	ASCBR010(0)	Reserved as of z/OS 1.12
16	(10)	DBL WORD	8	ASCBSUPC_PREZOS12(0)	- SUPERVISOR CELL FIELD
16	(10)	ADDRESS	4	ASCBSVRB_PREZOS12	- SVRB POOL ADDRESS.
20	(14)	SIGNED	4	ASCBSYNC_PREZOS12	- COUNT USED TO SYNCHRONIZE SVRB POOL.
24	(18)	ADDRESS	4	ASCBIOSP	- POINTER TO IOS PURGE INTERFACE CONTROL BLOCK (IPIB) (MDC308)
28	(1C)	BITSTRING	4	ASCBWQLK(0)	WEB QUEUE LOCK WORD SERIALIZATION: COMPARE AND SWAP OWNERSHIP: SUPERVISOR CONTROL
28	(1C)	BITSTRING	2	ASCBR01C	RESERVED, MUST BE ZERO

(But I think the PDF books are preferable)



ASVT mapping

Table 87. Structure ASVT

Offset Dec	Offset Hex	Type	Len	Name(Dim)	Description
0	(0)	STRUCTURE	0	ASVT	
0	(0)	CHARACTER	464	ASVTPRFX	Reserved for future expansion
464	(1D0)	DBL WORD	8	(0)	
464	(1D0)	BITSTRING	1	ASVTBEGN(0)	- BEGINNING OF ASVT
464	(1D0)	SIGNED	4	ASVTHWMASID	Highest ASID used since IPL
468	(1D4)	SIGNED	4	ASVTCURHIGHASID	Highest ASID currently used
472	(1D8)	ADDRESS	4	ASVTREUA	ADDRESS OF ASVTREUS BITS

Chapter 1. MVS Data Areas (ABE - IAR) **143**

Table 87. Structure ASVT (continued)

Offset Dec	Offset Hex	Type	Len	Name(Dim)	Description
476	(1DC)	ADDRESS	4	ASVTRAVL	ADDRESS OF FIRST AVAILABLE REUSABLE ASID SLOT
480	(1E0)	SIGNED	4	ASVTAAV	NUMBER OF FREE SLOTS ON THE ASVT AVAILABLE QUEUE.
484	(1E4)	SIGNED	4	ASVTAST	NUMBER OF FREE SLOTS ON THE START/ SASI QUEUE.
488	(1E8)	SIGNED	4	ASVTANR	NUMBER OF FREE SLOTS ON THE NON-REUSABLE REPLACEMENT QUEUE.
492	(1EC)	SIGNED	4	ASVTSTRT	ORIGINAL SIZE OF START/SASI QUEUE.
496	(1F0)	SIGNED	4	ASVTNONR	ORIGINAL SIZE OF NON-REUSABLE REPLACEMENT QUEUE.
500	(1F4)	SIGNED	4	ASVTMAXI	- ORIGINAL MAX USERS COUNT AS INPUT TO IEAVNP09. OWNERSHIP - SUPERVISOR CONTROL SERIALIZATION - NIP RIM PROCESS
504	(1F8)	BITSTRING	8		- RESERVED. WAS ASVTRSHD/DSHD
512	(200)	CHARACTER	4	ASVTASVT	- ACRONYM IN EBCDIC -ASVT-
516	(204)	SIGNED	4	ASVTMAXU	- MAXIMUM NUMBER OF ADDRESS SPACES
520	(208)	SIGNED	4	ASVTMDSC	- MAXUSER DEFICIT SLOT COUNT. ASVTMDSC = ASVTMAXI - ASVTAAV - NUMBER OF ACTIVE A.S. INCREMENTED WHEN WE TRY TO TAKE A REPLACEMENT SLOT BUT THERE AREN'T ANY. DECREMENTED WHEN NON-ZERO AND A NONREUSEABLE ASID BECOMES REUSEABLE AND WE ADD A SLOT TO THE MAXUSER POOL WHEN AN ADDRESS SPACE BECOMES REUSEABLE.
524	(20C)	ADDRESS	4	ASVTFRST	- ADDRESS OF FIRST AVAILABLE ASVT ENTRY (MDC300)
		1...		ASVTAVAI	"X'80'" - BIT ONE IF ASID IS AVAILABLE AND ZERO IF ASID IS

Ministerie van Financiën

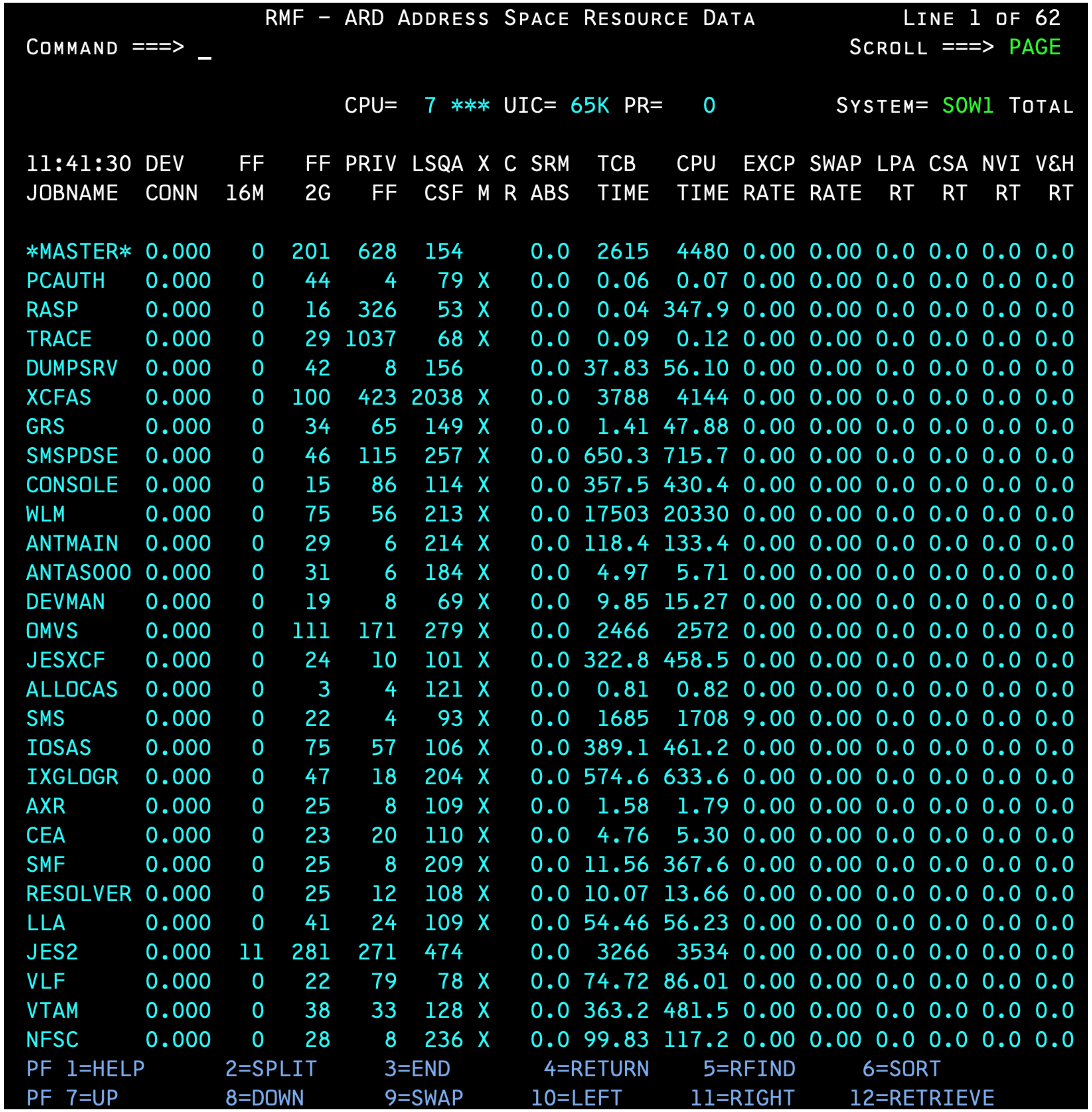
№ 3 What can we do with them

Activities

- Follow **chains** from **anchors**
- Format fields
- Extract real-time information
- Correlate values with events
- Draw conclusions about resource usage and serialization delays

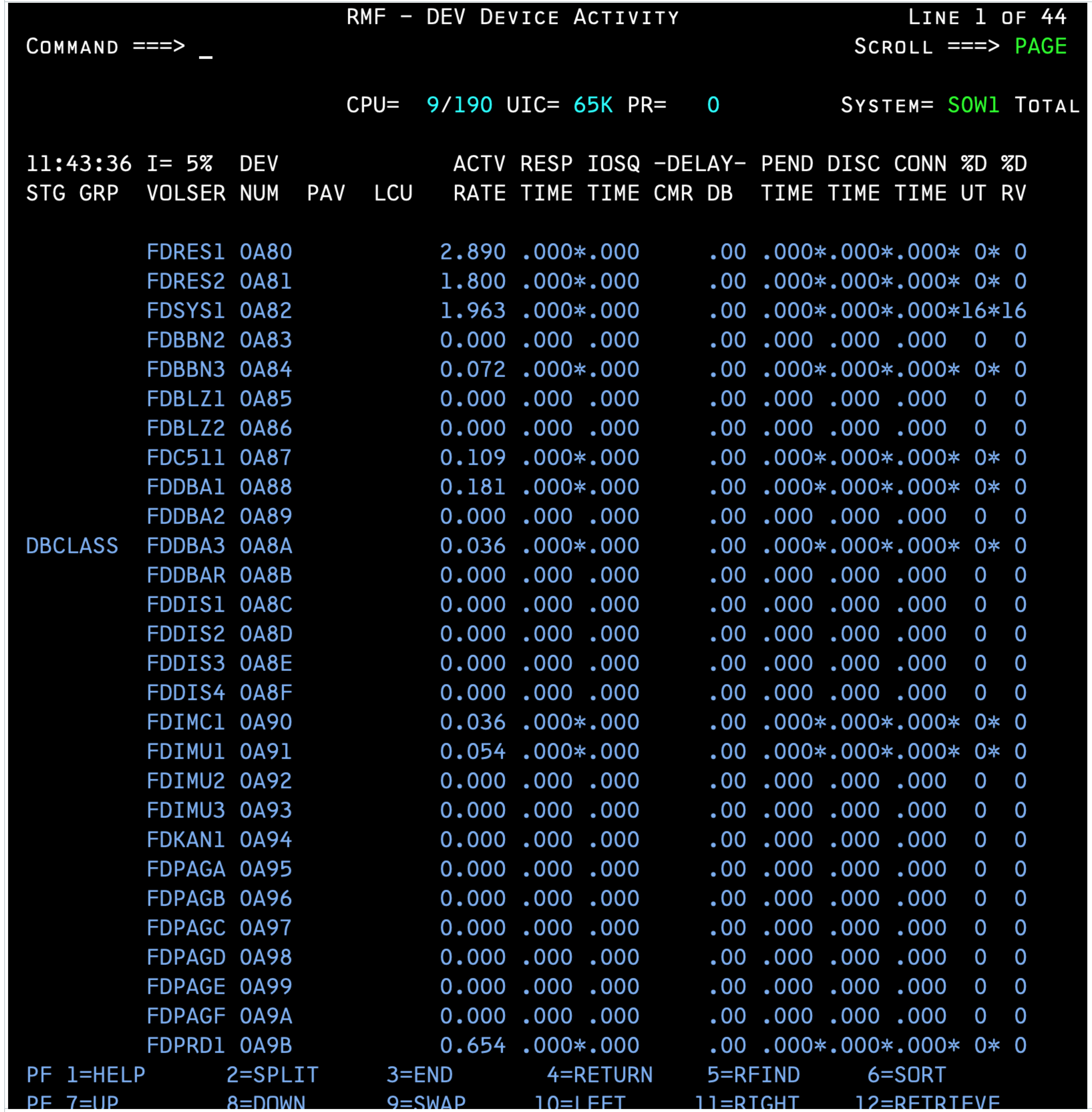
- When using SDSF and RMF(II, III), you look into pre-cooked views of control blocks
  - And more challenging endeavours, to be shown hereafter

RMF II Address Space Resource Data



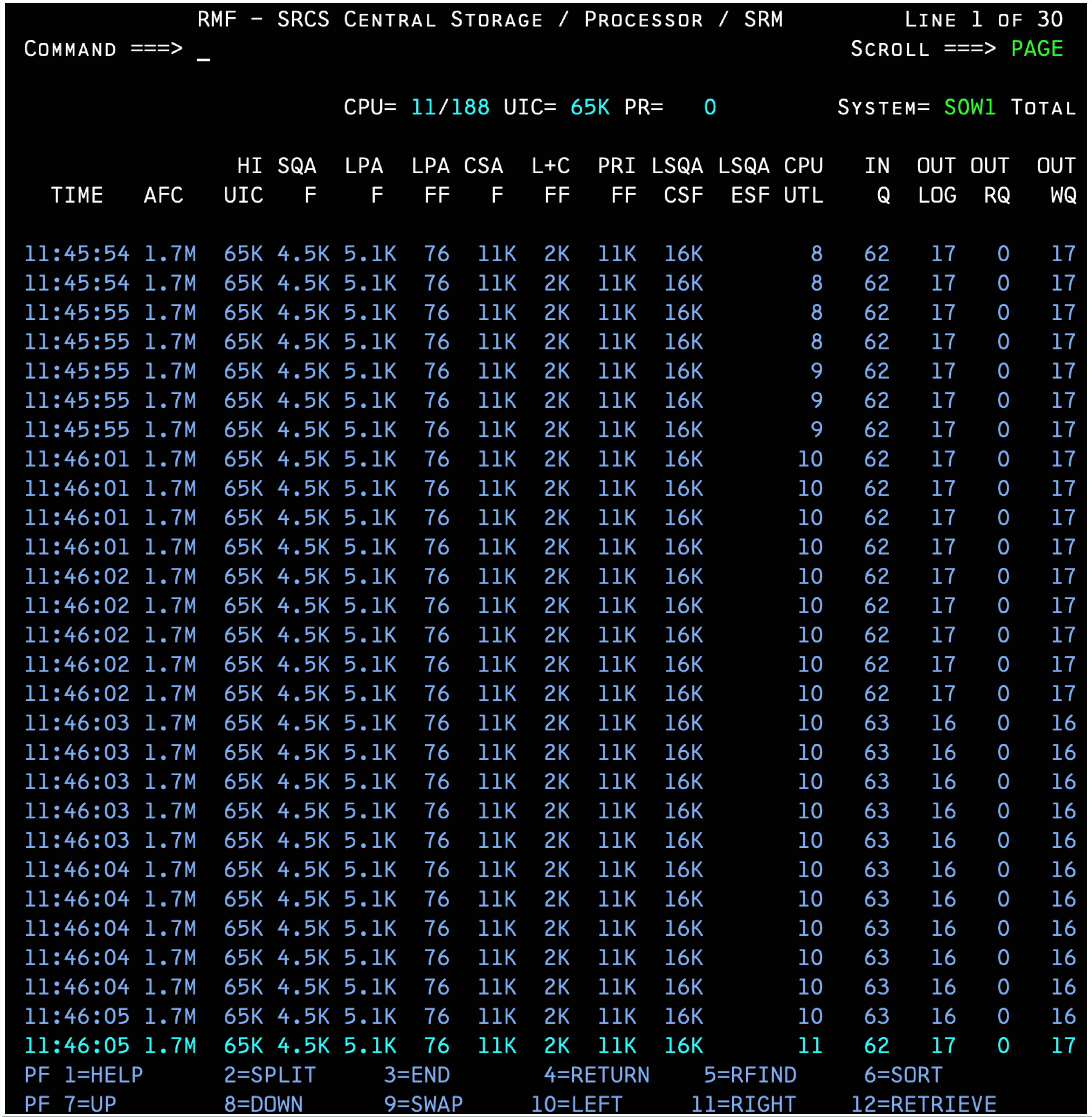
```
                   RMF - ARD ADDRESS SPACE RESOURCE DATA             LINE 1 OF 62
COMMAND ===> _                                                   SCROLL ===> PAGE

                             CPU=  7 *** UIC= 65K PR=   0        SYSTEM= SOW1 TOTAL

11:41:30 DEV    FF    FF PRIV LSQA X C SRM   TCB    CPU  EXCP SWAP LPA CSA NVI V&H
JOBNAME  CONN  16M    2G   FF  CSF M R ABS  TIME   TIME  RATE RATE  RT  RT  RT  RT

*MASTER* 0.000   0   201  628  154     0.0  2615   4480 0.00 0.00 0.0 0.0 0.0 0.0
PCAUTH   0.000   0    44    4   79 X   0.0  0.06   0.07 0.00 0.00 0.0 0.0 0.0 0.0
RASP     0.000   0    16  326   53 X   0.0  0.04  347.9 0.00 0.00 0.0 0.0 0.0 0.0
TRACE    0.000   0    29 1037   68 X   0.0  0.09   0.12 0.00 0.00 0.0 0.0 0.0 0.0
DUMPSRV  0.000   0    42    8  156     0.0 37.83  56.10 0.00 0.00 0.0 0.0 0.0 0.0
XCFAS    0.000   0   100  423 2038 X   0.0  3788   4144 0.00 0.00 0.0 0.0 0.0 0.0
GRS      0.000   0    34   65  149 X   0.0  1.41  47.88 0.00 0.00 0.0 0.0 0.0 0.0
SMSPDSE  0.000   0    46  115  257 X   0.0 650.3  715.7 0.00 0.00 0.0 0.0 0.0 0.0
CONSOLE  0.000   0    15   86  114 X   0.0 357.5  430.4 0.00 0.00 0.0 0.0 0.0 0.0
WLM      0.000   0    75   56  213 X   0.0 17503  20330 0.00 0.00 0.0 0.0 0.0 0.0
ANTMAIN  0.000   0    29    6  214 X   0.0 118.4  133.4 0.00 0.00 0.0 0.0 0.0 0.0
ANTAS000 0.000   0    31    6  184 X   0.0  4.97   5.71 0.00 0.00 0.0 0.0 0.0 0.0
DEVMAN   0.000   0    19    8   69 X   0.0  9.85  15.27 0.00 0.00 0.0 0.0 0.0 0.0
OMVS     0.000   0   111  171  279 X   0.0  2466   2572 0.00 0.00 0.0 0.0 0.0 0.0
JESXCF   0.000   0    24   10  101 X   0.0 322.8  458.5 0.00 0.00 0.0 0.0 0.0 0.0
ALLOCAS  0.000   0     3    4  121 X   0.0  0.81   0.82 0.00 0.00 0.0 0.0 0.0 0.0
SMS      0.000   0    22    4   93 X   0.0  1685   1708 9.00 0.00 0.0 0.0 0.0 0.0
IOSAS    0.000   0    75   57  106 X   0.0 389.1  461.2 0.00 0.00 0.0 0.0 0.0 0.0
IXGLOGR  0.000   0    47   18  204 X   0.0 574.6  633.6 0.00 0.00 0.0 0.0 0.0 0.0
AXR      0.000   0    25    8  109 X   0.0  1.58   1.79 0.00 0.00 0.0 0.0 0.0 0.0
CEA      0.000   0    23   20  110 X   0.0  4.76   5.30 0.00 0.00 0.0 0.0 0.0 0.0
SMF      0.000   0    25    8  209 X   0.0 11.56  367.6 0.00 0.00 0.0 0.0 0.0 0.0
RESOLVER 0.000   0    25   12  108 X   0.0 10.07  13.66 0.00 0.00 0.0 0.0 0.0 0.0
LLA      0.000   0    41   24  109 X   0.0 54.46  56.23 0.00 0.00 0.0 0.0 0.0 0.0
JES2     0.000  11   281  271  474     0.0  3266   3534 0.00 0.00 0.0 0.0 0.0 0.0
VLF      0.000   0    22   79   78 X   0.0 74.72  86.01 0.00 0.00 0.0 0.0 0.0 0.0
VTAM     0.000   0    38   33  128 X   0.0 363.2  481.5 0.00 0.00 0.0 0.0 0.0 0.0
NFSC     0.000   0    28    8  236 X   0.0 99.83  117.2 0.00 0.00 0.0 0.0 0.0 0.0
PF 1=HELP     2=SPLIT     3=END       4=RETURN     5=RFIND      6=SORT
PF 7=UP       8=DOWN      9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE
```

RMF II Device Activity



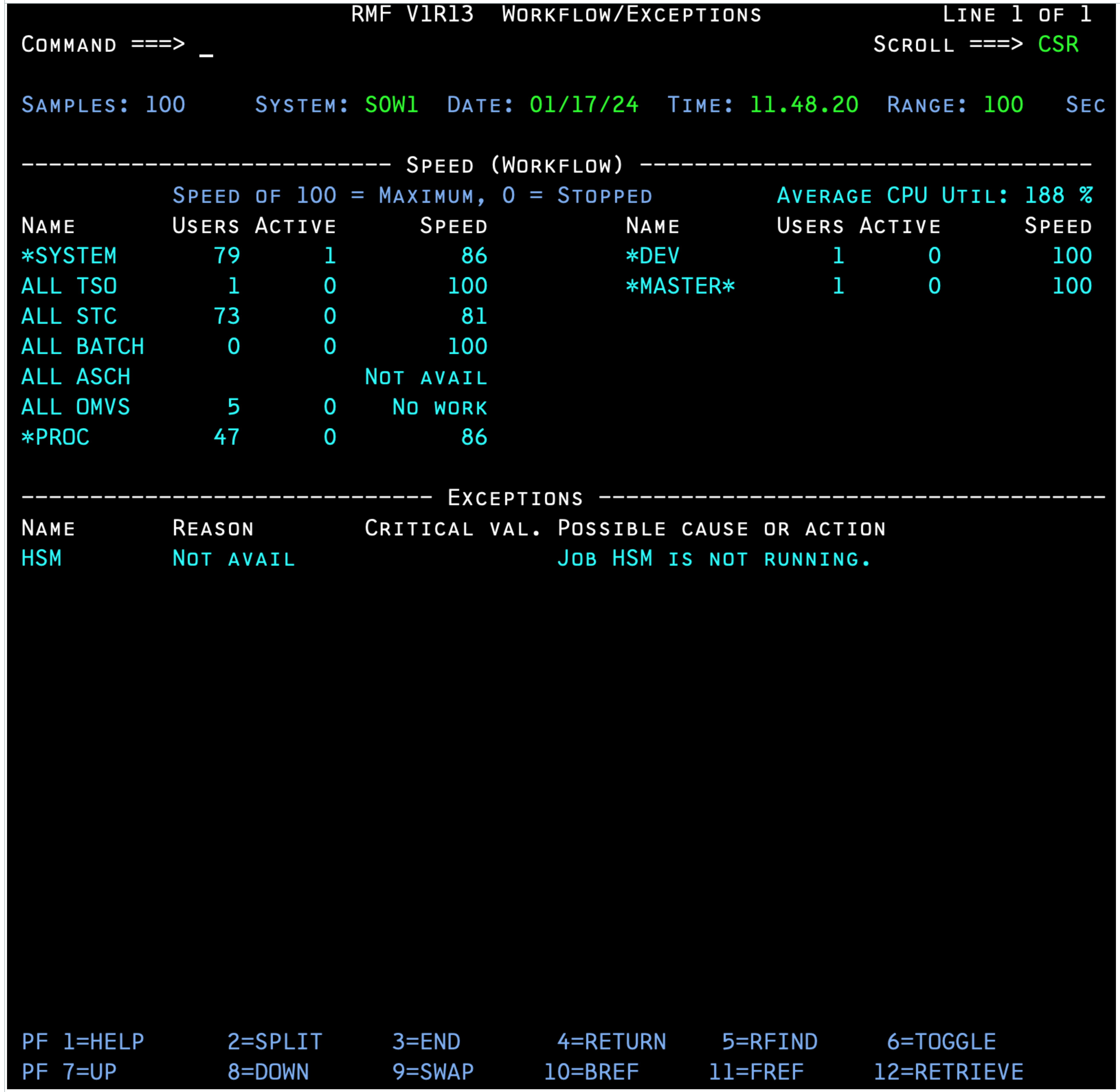
```
                        RMF - DEV Device Activity                 Line 1 of 44
Command ===> _                                                Scroll ===> PAGE

                        CPU=  9/190 UIC= 65K PR=    0          System= SOW1 Total

11:43:36 I= 5%  DEV              ACTV RESP IOSQ -DELAY- PEND DISC CONN %D %D
STG GRP  VOLSER NUM  PAV  LCU    RATE TIME TIME CMR DB  TIME TIME TIME UT RV

         FDRES1 0A80            2.890 .000*.000      .00 .000*.000*.000* 0* 0
         FDRES2 0A81            1.800 .000*.000      .00 .000*.000*.000* 0* 0
         FDSYS1 0A82            1.963 .000*.000      .00 .000*.000*.000*16*16
         FDBBN2 0A83            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDBBN3 0A84            0.072 .000*.000      .00 .000*.000*.000* 0* 0
         FDBLZ1 0A85            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDBLZ2 0A86            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDC511 0A87            0.109 .000*.000      .00 .000*.000*.000* 0* 0
         FDDBA1 0A88            0.181 .000*.000      .00 .000*.000*.000* 0* 0
         FDDBA2 0A89            0.000 .000 .000      .00 .000 .000 .000  0  0
DBCLASS  FDDBA3 0A8A            0.036 .000*.000      .00 .000*.000*.000* 0* 0
         FDDBAR 0A8B            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDDIS1 0A8C            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDDIS2 0A8D            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDDIS3 0A8E            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDDIS4 0A8F            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDIMC1 0A90            0.036 .000*.000      .00 .000*.000*.000* 0* 0
         FDIMU1 0A91            0.054 .000*.000      .00 .000*.000*.000* 0* 0
         FDIMU2 0A92            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDIMU3 0A93            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDKAN1 0A94            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDPAGA 0A95            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDPAGB 0A96            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDPAGC 0A97            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDPAGD 0A98            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDPAGE 0A99            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDPAGF 0A9A            0.000 .000 .000      .00 .000 .000 .000  0  0
         FDPRD1 0A9B            0.654 .000*.000      .00 .000*.000*.000* 0* 0
PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=RFIND     6=SORT
PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

RMF II Memory Activity incl UIC

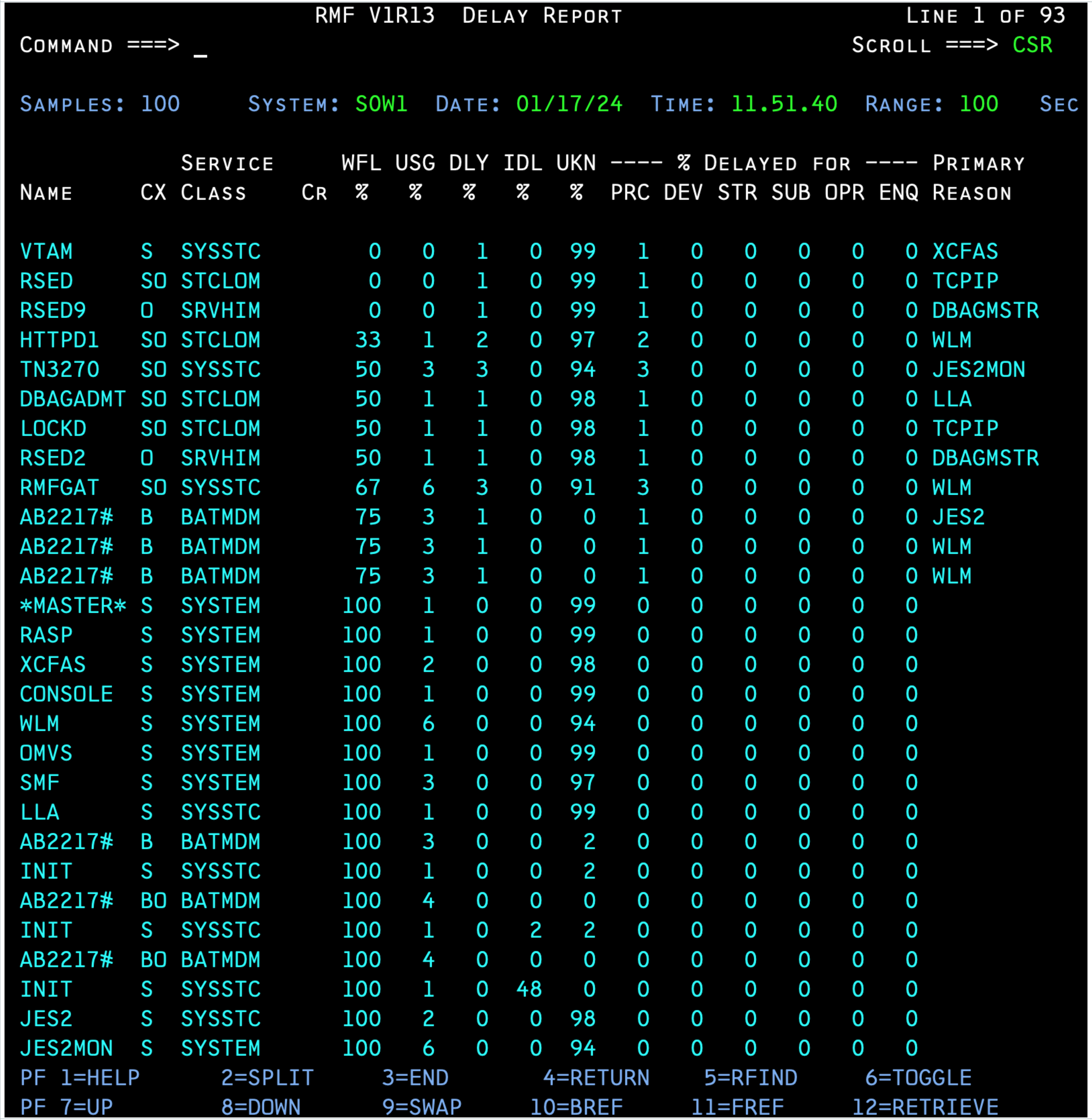


```
                RMF - SRCS CENTRAL STORAGE / PROCESSOR / SRM          LINE 1 OF 30
 COMMAND ===> _                                                    SCROLL ===> PAGE

                            CPU= 11/188 UIC= 65K PR=   0          SYSTEM= SOW1 TOTAL

                    HI  SQA  LPA  LPA CSA  L+C  PRI LSQA LSQA CPU  IN  OUT OUT  OUT
   TIME     AFC    UIC    F    F   FF   F   FF   FF  CSF  ESF UTL   Q  LOG  RQ   WQ

 11:45:54 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K          8  62   17   0   17
 11:45:54 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K          8  62   17   0   17
 11:45:55 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K          8  62   17   0   17
 11:45:55 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K          8  62   17   0   17
 11:45:55 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K          9  62   17   0   17
 11:45:55 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K          9  62   17   0   17
 11:45:55 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K          9  62   17   0   17
 11:46:01 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  62   17   0   17
 11:46:01 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  62   17   0   17
 11:46:01 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  62   17   0   17
 11:46:01 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  62   17   0   17
 11:46:02 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  62   17   0   17
 11:46:02 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  62   17   0   17
 11:46:02 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  62   17   0   17
 11:46:02 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  62   17   0   17
 11:46:02 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  62   17   0   17
 11:46:03 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:03 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:03 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:03 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:03 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:04 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:04 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:04 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:04 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:04 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:05 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         10  63   16   0   16
 11:46:05 1.7M  65K 4.5K 5.1K  76  11K  2K  11K  16K         11  62   17   0   17
 PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=RFIND     6=SORT
 PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

RMF III WFEX Workflow/Exceptions



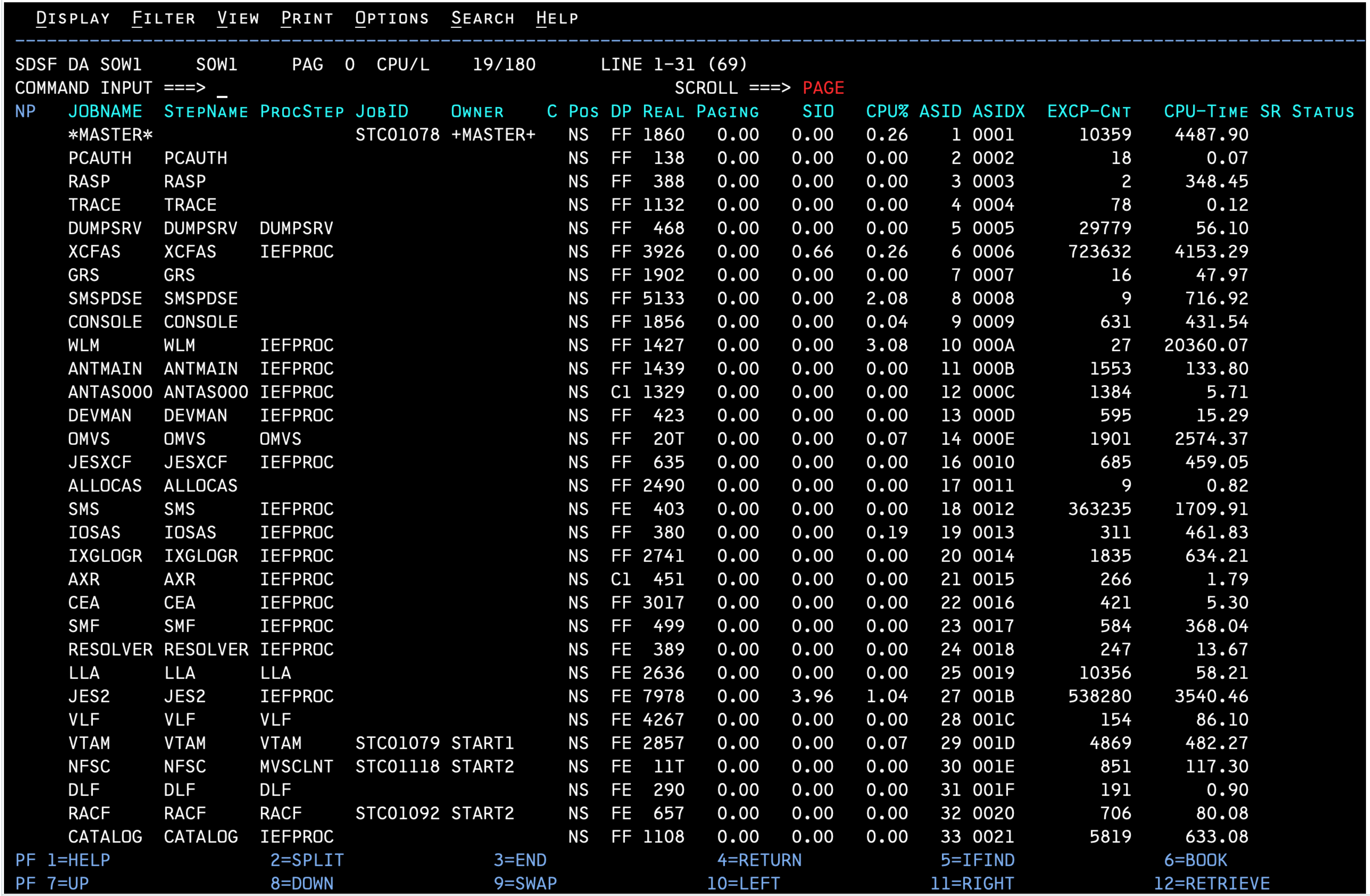
```
                        RMF V1R13  WORKFLOW/EXCEPTIONS             LINE 1 OF 1
COMMAND ===> _                                                SCROLL ===> CSR

SAMPLES: 100     SYSTEM: SOW1  DATE: 01/17/24  TIME: 11.48.20  RANGE: 100   SEC

--------------------------- SPEED (WORKFLOW) ---------------------------------
           SPEED OF 100 = MAXIMUM, 0 = STOPPED         AVERAGE CPU UTIL: 188 %
NAME       USERS ACTIVE       SPEED           NAME       USERS ACTIVE       SPEED
*SYSTEM       79      1          86           *DEV           1      0         100
ALL TSO        1      0         100           *MASTER*       1      0         100
ALL STC       73      0          81
ALL BATCH      0      0         100
ALL ASCH                  NOT AVAIL
ALL OMVS       5      0     NO WORK
*PROC         47      0          86

------------------------------ EXCEPTIONS -------------------------------------
NAME       REASON        CRITICAL VAL. POSSIBLE CAUSE OR ACTION
HSM        NOT AVAIL                   JOB HSM IS NOT RUNNING.












PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=RFIND     6=TOGGLE
PF 7=UP        8=DOWN      9=SWAP     10=BREF     11=FREF     12=RETRIEVE
```



```
                        RMF V1R13   Delay Report                 Line 1 of 93
Command ===> _                                                  Scroll ===> CSR

Samples: 100     System: SOW1  Date: 01/17/24  Time: 11.51.40  Range: 100   Sec

            Service     WFL USG DLY IDL UKN ---- % Delayed for ---- Primary
Name     CX Class    Cr  %   %   %   %   %  PRC DEV STR SUB OPR ENQ Reason

VTAM     S  SYSSTC        0   0   1   0  99   1   0   0   0   0   0 XCFAS
RSED     SO STCLOM        0   0   1   0  99   1   0   0   0   0   0 TCPIP
RSED9    O  SRVHIM        0   0   1   0  99   1   0   0   0   0   0 DBAGMSTR
HTTPD1   SO STCLOM       33   1   2   0  97   2   0   0   0   0   0 WLM
TN3270   SO SYSSTC       50   3   3   0  94   3   0   0   0   0   0 JES2MON
DBAGADMT SO STCLOM       50   1   1   0  98   1   0   0   0   0   0 LLA
LOCKD    SO STCLOM       50   1   1   0  98   1   0   0   0   0   0 TCPIP
RSED2    O  SRVHIM       50   1   1   0  98   1   0   0   0   0   0 DBAGMSTR
RMFGAT   SO SYSSTC       67   6   3   0  91   3   0   0   0   0   0 WLM
AB2217#  B  BATMDM       75   3   1   0   0   1   0   0   0   0   0 JES2
AB2217#  B  BATMDM       75   3   1   0   0   1   0   0   0   0   0 WLM
AB2217#  B  BATMDM       75   3   1   0   0   1   0   0   0   0   0 WLM
*MASTER* S  SYSTEM      100   1   0   0  99   0   0   0   0   0   0
RASP     S  SYSTEM      100   1   0   0  99   0   0   0   0   0   0
XCFAS    S  SYSTEM      100   2   0   0  98   0   0   0   0   0   0
CONSOLE  S  SYSTEM      100   1   0   0  99   0   0   0   0   0   0
WLM      S  SYSTEM      100   6   0   0  94   0   0   0   0   0   0
OMVS     S  SYSTEM      100   1   0   0  99   0   0   0   0   0   0
SMF      S  SYSTEM      100   3   0   0  97   0   0   0   0   0   0
LLA      S  SYSSTC      100   1   0   0  99   0   0   0   0   0   0
AB2217#  B  BATMDM      100   3   0   0   2   0   0   0   0   0   0
INIT     S  SYSSTC      100   1   0   0   2   0   0   0   0   0   0
AB2217#  BO BATMDM      100   4   0   0   0   0   0   0   0   0   0
INIT     S  SYSSTC      100   1   0   2   2   0   0   0   0   0   0
AB2217#  BO BATMDM      100   4   0   0   0   0   0   0   0   0   0
INIT     S  SYSSTC      100   1   0  48   0   0   0   0   0   0   0
JES2     S  SYSSTC      100   2   0   0  98   0   0   0   0   0   0
JES2MON  S  SYSTEM      100   6   0   0  94   0   0   0   0   0   0
PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=RFIND     6=TOGGLE
PF 7=UP        8=DOWN      9=SWAP     10=BREF     11=FREF     12=RETRIEVE
```

RMF III Delay Report

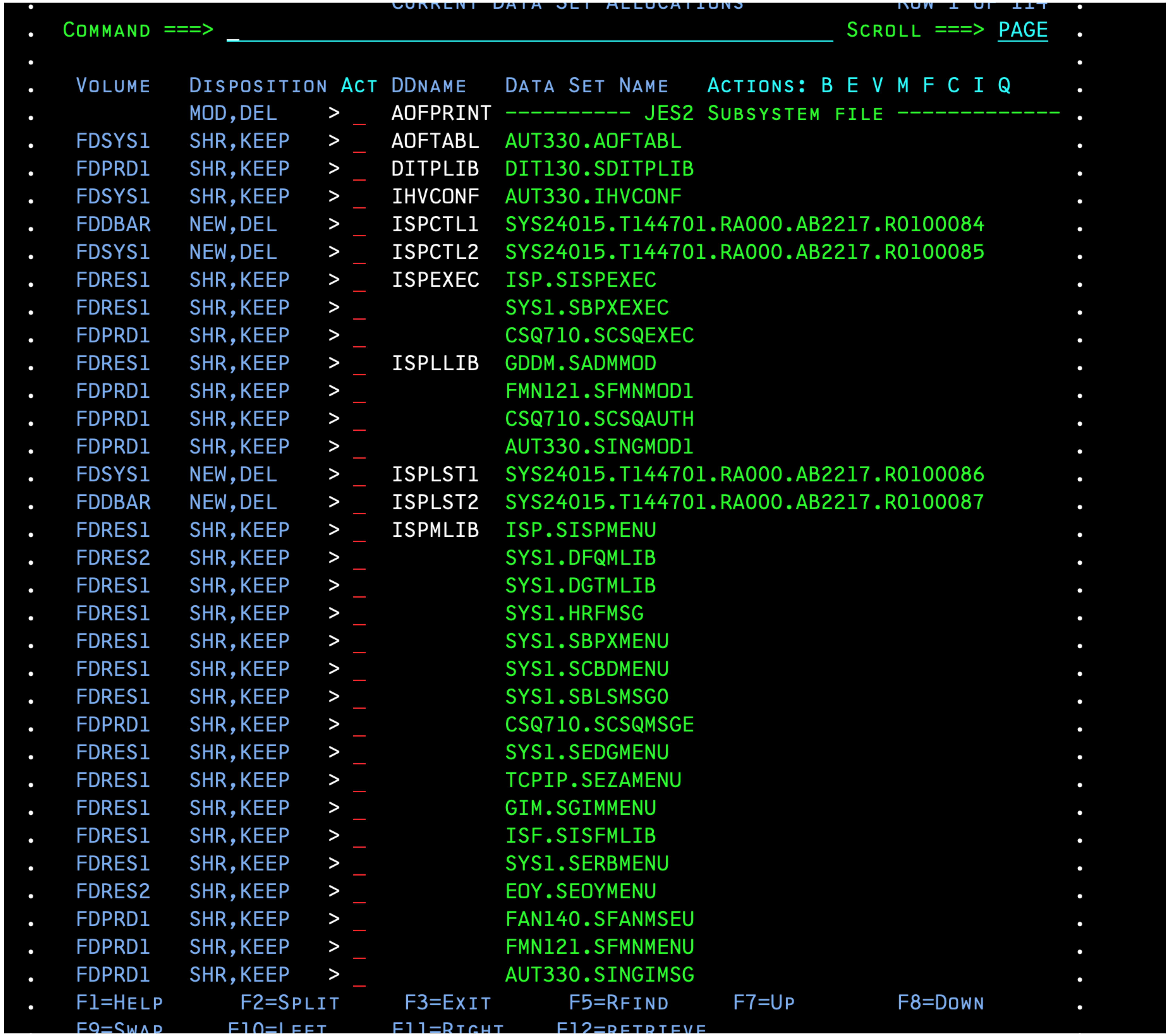
SDSF DA (ASCB+JES2 Control Blocks)



```
  Display  Filter  View  Print  Options  Search  Help
-------------------------------------------------------------------------------------------------------------------------------
SDSF DA SOW1     SOW1     PAG  0  CPU/L    19/180      LINE 1-31 (69)
COMMAND INPUT ===> _                                          SCROLL ===> PAGE
NP   JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO   CPU% ASID ASIDX  EXCP-Cnt   CPU-Time SR Status
     *MASTER*                   STC01078 +MASTER+   NS  FF 1860   0.00   0.00   0.26    1 0001      10359    4487.90
     PCAUTH   PCAUTH                                NS  FF  138   0.00   0.00   0.00    2 0002         18       0.07
     RASP     RASP                                  NS  FF  388   0.00   0.00   0.00    3 0003          2     348.45
     TRACE    TRACE                                 NS  FF 1132   0.00   0.00   0.00    4 0004         78       0.12
     DUMPSRV  DUMPSRV  DUMPSRV                      NS  FF  468   0.00   0.00   0.00    5 0005      29779      56.10
     XCFAS    XCFAS    IEFPROC                      NS  FF 3926   0.00   0.66   0.26    6 0006     723632    4153.29
     GRS      GRS                                   NS  FF 1902   0.00   0.00   0.00    7 0007         16      47.97
     SMSPDSE  SMSPDSE                               NS  FF 5133   0.00   0.00   2.08    8 0008          9     716.92
     CONSOLE  CONSOLE                               NS  FF 1856   0.00   0.00   0.04    9 0009        631     431.54
     WLM      WLM      IEFPROC                      NS  FF 1427   0.00   0.00   3.08   10 000A         27   20360.07
     ANTMAIN  ANTMAIN  IEFPROC                      NS  FF 1439   0.00   0.00   0.00   11 000B       1553     133.80
     ANTAS000 ANTAS000 IEFPROC                      NS  C1 1329   0.00   0.00   0.00   12 000C       1384       5.71
     DEVMAN   DEVMAN   IEFPROC                      NS  FF  423   0.00   0.00   0.00   13 000D        595      15.29
     OMVS     OMVS     OMVS                         NS  FF  20T   0.00   0.00   0.07   14 000E       1901    2574.37
     JESXCF   JESXCF   IEFPROC                      NS  FF  635   0.00   0.00   0.00   16 0010        685     459.05
     ALLOCAS  ALLOCAS                               NS  FF 2490   0.00   0.00   0.00   17 0011          9       0.82
     SMS      SMS      IEFPROC                      NS  FE  403   0.00   0.00   0.00   18 0012     363235    1709.91
     IOSAS    IOSAS    IEFPROC                      NS  FF  380   0.00   0.00   0.19   19 0013        311     461.83
     IXGLOGR  IXGLOGR  IEFPROC                      NS  FF 2741   0.00   0.00   0.00   20 0014       1835     634.21
     AXR      AXR      IEFPROC                      NS  C1  451   0.00   0.00   0.00   21 0015        266       1.79
     CEA      CEA      IEFPROC                      NS  FF 3017   0.00   0.00   0.00   22 0016        421       5.30
     SMF      SMF      IEFPROC                      NS  FF  499   0.00   0.00   0.00   23 0017        584     368.04
     RESOLVER RESOLVER IEFPROC                      NS  FE  389   0.00   0.00   0.00   24 0018        247      13.67
     LLA      LLA      LLA                          NS  FE 2636   0.00   0.00   0.00   25 0019      10356      58.21
     JES2     JES2     IEFPROC                      NS  FE 7978   0.00   3.96   1.04   27 001B     538280    3540.46
     VLF      VLF      VLF                          NS  FE 4267   0.00   0.00   0.00   28 001C        154      86.10
     VTAM     VTAM     VTAM     STC01079 START1     NS  FE 2857   0.00   0.00   0.07   29 001D       4869     482.27
     NFSC     NFSC     MVSCLNT  STC01118 START2     NS  FE  11T   0.00   0.00   0.00   30 001E        851     117.30
     DLF      DLF      DLF                          NS  FE  290   0.00   0.00   0.00   31 001F        191       0.90
     RACF     RACF     RACF     STC01092 START2     NS  FE  657   0.00   0.00   0.00   32 0020        706      80.08
     CATALOG  CATALOG  IEFPROC                      NS  FF 1108   0.00   0.00   0.00   33 0021       5819     633.08
PF 1=HELP               2=SPLIT              3=END                4=RETURN             5=IFIND              6=BOOK
PF 7=UP                 8=DOWN               9=SWAP              10=LEFT              11=RIGHT             12=RETRIEVE
```

Ministerie van Financiën

№ 4 View Control Blocks: ISRDDN

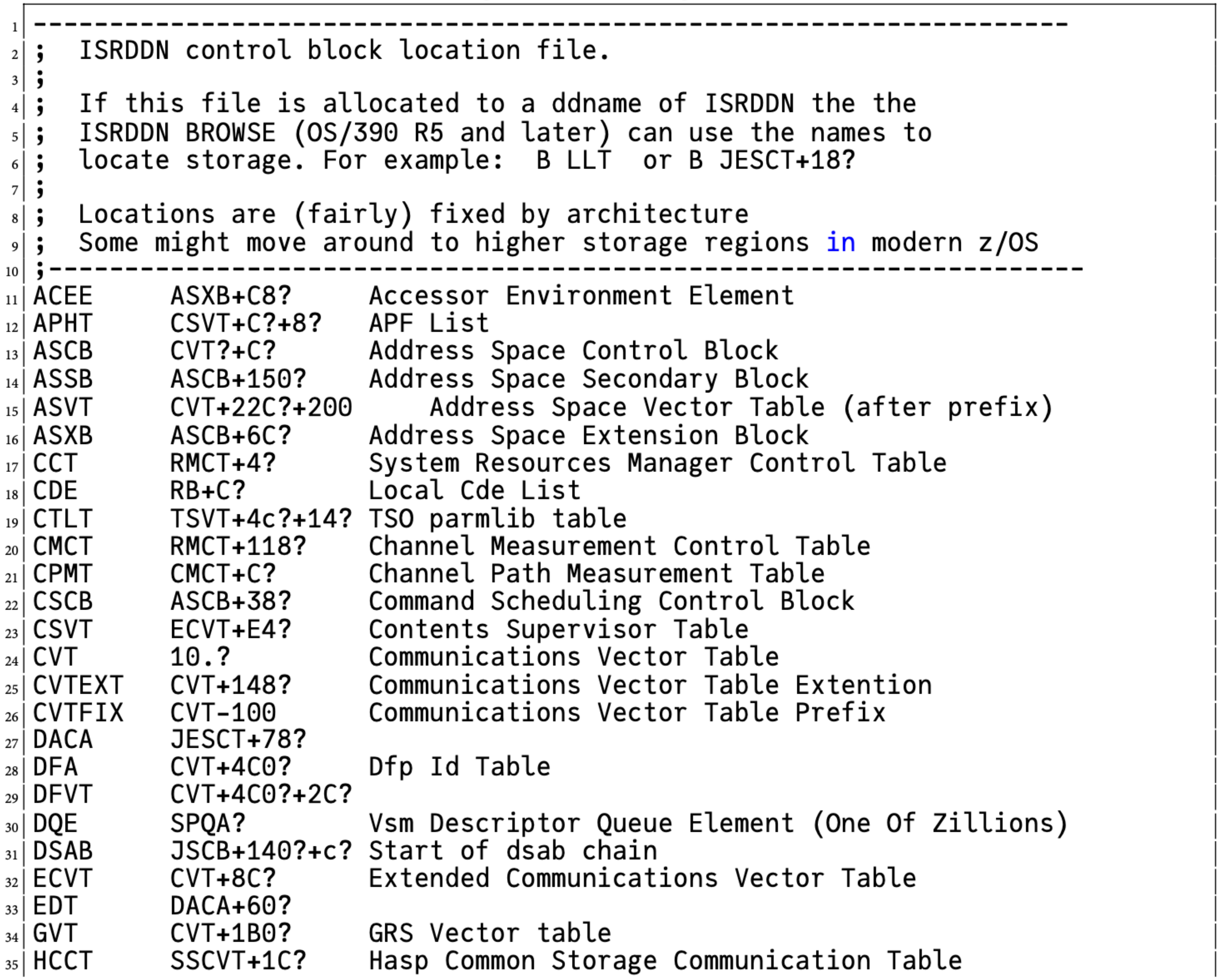


```
                       Current Data Set Allocations          Row 1 of 114
 Command ===> ____________________________________________ Scroll ===> PAGE

 Volume    Disposition Act DDname    Data Set Name    Actions: B E V M F C I Q
           MOD,DEL     > _ AOFPRINT ---------- JES2 Subsystem file -------------
 FDSYS1    SHR,KEEP    > _ AOFTABL  AUT330.AOFTABL
 FDPRD1    SHR,KEEP    > _ DITPLIB  DIT130.SDITPLIB
 FDSYS1    SHR,KEEP    > _ IHVCONF  AUT330.IHVCONF
 FDDBAR    NEW,DEL     > _ ISPCTL1  SYS24015.T144701.RA000.AB2217.R0100084
 FDSYS1    NEW,DEL     > _ ISPCTL2  SYS24015.T144701.RA000.AB2217.R0100085
 FDRES1    SHR,KEEP    > _ ISPEXEC  ISP.SISPEXEC
 FDRES1    SHR,KEEP    > _          SYS1.SBPXEXEC
 FDPRD1    SHR,KEEP    > _          CSQ710.SCSQEXEC
 FDRES1    SHR,KEEP    > _ ISPLLIB  GDDM.SADMMOD
 FDPRD1    SHR,KEEP    > _          FMN121.SFMNMOD1
 FDPRD1    SHR,KEEP    > _          CSQ710.SCSQAUTH
 FDPRD1    SHR,KEEP    > _          AUT330.SINGMOD1
 FDSYS1    NEW,DEL     > _ ISPLST1  SYS24015.T144701.RA000.AB2217.R0100086
 FDDBAR    NEW,DEL     > _ ISPLST2  SYS24015.T144701.RA000.AB2217.R0100087
 FDRES1    SHR,KEEP    > _ ISPMLIB  ISP.SISPMENU
 FDRES2    SHR,KEEP    > _          SYS1.DFQMLIB
 FDRES1    SHR,KEEP    > _          SYS1.DGTMLIB
 FDRES1    SHR,KEEP    > _          SYS1.HRFMSG
 FDRES1    SHR,KEEP    > _          SYS1.SBPXMENU
 FDRES1    SHR,KEEP    > _          SYS1.SCBDMENU
 FDRES1    SHR,KEEP    > _          SYS1.SBLSMSGO
 FDPRD1    SHR,KEEP    > _          CSQ710.SCSQMSGE
 FDRES1    SHR,KEEP    > _          SYS1.SEDGMENU
 FDRES1    SHR,KEEP    > _          TCPIP.SEZAMENU
 FDRES1    SHR,KEEP    > _          GIM.SGIMMENU
 FDRES1    SHR,KEEP    > _          ISF.SISFMLIB
 FDRES1    SHR,KEEP    > _          SYS1.SERBMENU
 FDRES2    SHR,KEEP    > _          EOY.SEOYMENU
 FDPRD1    SHR,KEEP    > _          FAN140.SFANMSEU
 FDPRD1    SHR,KEEP    > _          FMN121.SFMNMENU
 FDPRD1    SHR,KEEP    > _          AUT330.SINGIMSG
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F7=Up        F8=Down
 F9=Swap     F10=Left     F11=Right    F12=Retrieve
```

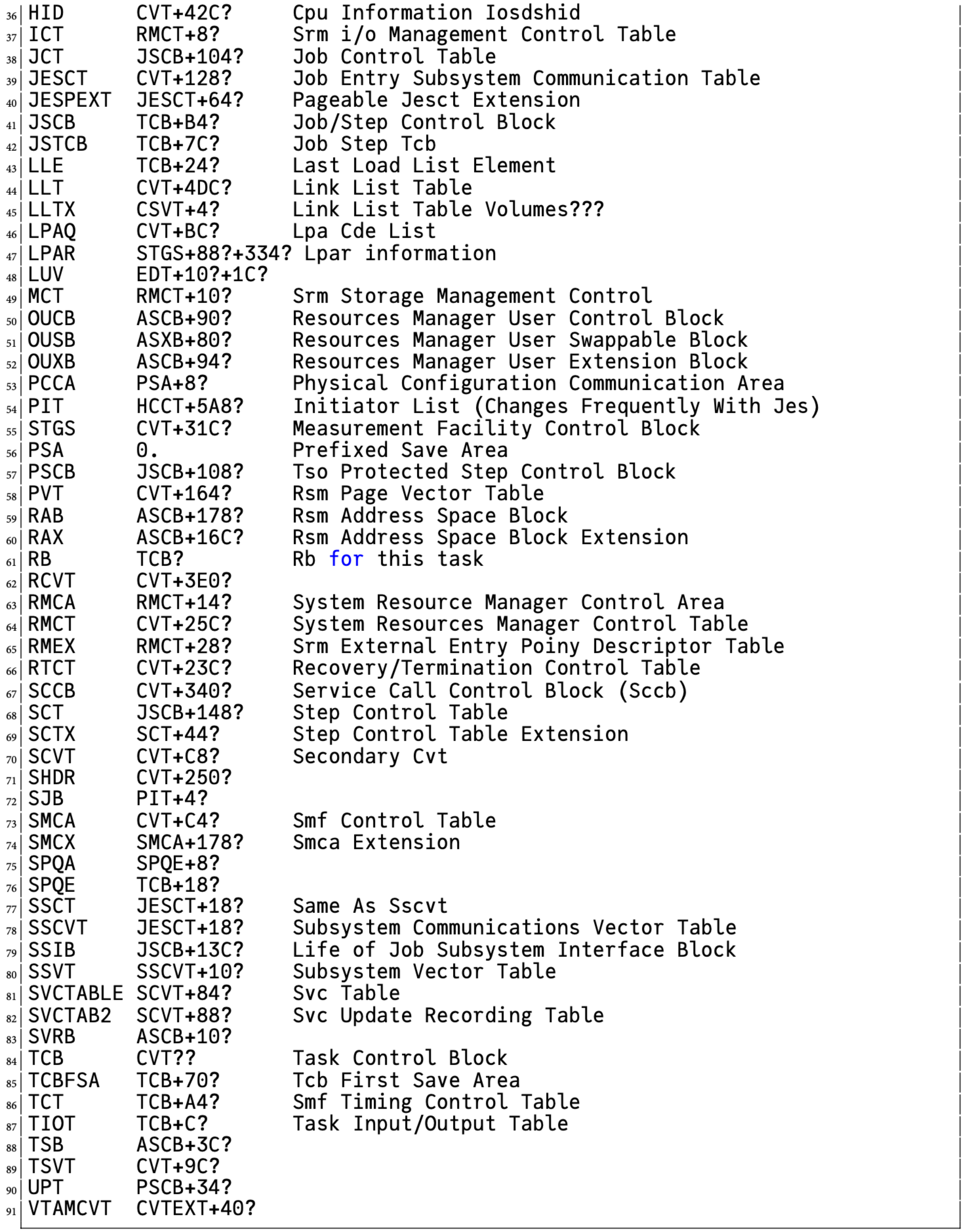
TSO ISRDDN
or, DDLIST

The default view shows allocated files (ddnames and datastes)

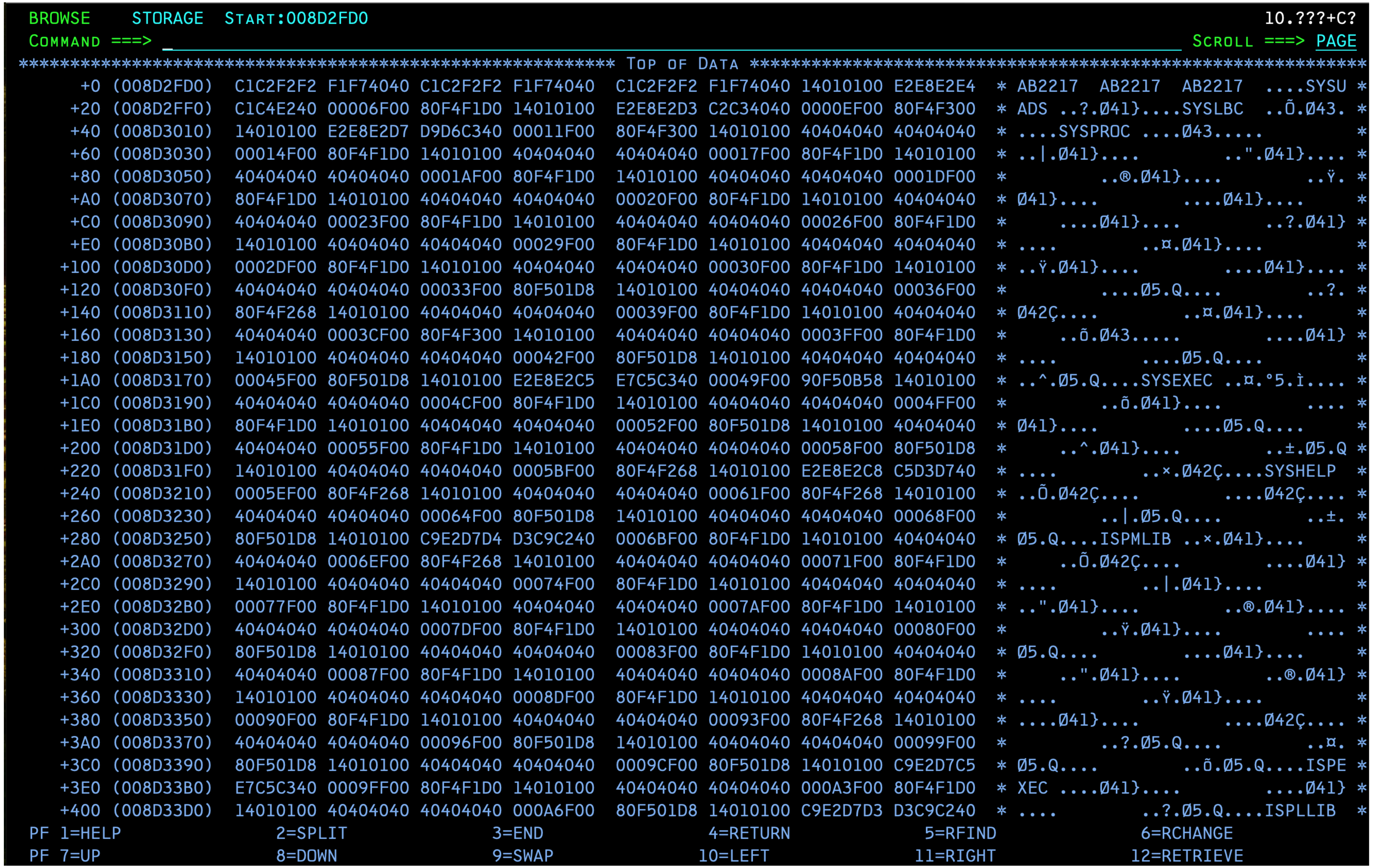
First, allocate a mapping file to DDNAME ISRDDN



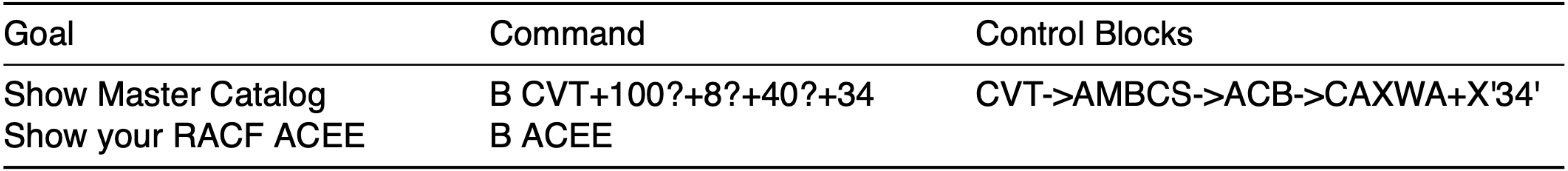
```
--------------------------------------------------------------------
;  ISRDDN control block location file.
;
;  If this file is allocated to a ddname of ISRDDN the the
;  ISRDDN BROWSE (OS/390 R5 and later) can use the names to
;  locate storage. For example:  B LLT  or B JESCT+18?
;
;  Locations are (fairly) fixed by architecture
;  Some might move around to higher storage regions in modern z/OS
;--------------------------------------------------------------------
ACEE     ASXB+C8?      Accessor Environment Element
APHT     CSVT+C?+8?    APF List
ASCB     CVT?+C?       Address Space Control Block
ASSB     ASCB+150?     Address Space Secondary Block
ASVT     CVT+22C?+200      Address Space Vector Table (after prefix)
ASXB     ASCB+6C?      Address Space Extension Block
CCT      RMCT+4?       System Resources Manager Control Table
CDE      RB+C?         Local Cde List
CTLT     TSVT+4c?+14?  TSO parmlib table
CMCT     RMCT+118?     Channel Measurement Control Table
CPMT     CMCT+C?       Channel Path Measurement Table
CSCB     ASCB+38?      Command Scheduling Control Block
CSVT     ECVT+E4?      Contents Supervisor Table
CVT      10.?          Communications Vector Table
CVTEXT   CVT+148?      Communications Vector Table Extention
CVTFIX   CVT-100       Communications Vector Table Prefix
DACA     JESCT+78?
DFA      CVT+4C0?      Dfp Id Table
DFVT     CVT+4C0?+2C?
DQE      SPQA?         Vsm Descriptor Queue Element (One Of Zillions)
DSAB     JSCB+140?+c?  Start of dsab chain
ECVT     CVT+8C?       Extended Communications Vector Table
EDT      DACA+60?
GVT      CVT+1B0?      GRS Vector table
HCCT     SSCVT+1C?     Hasp Common Storage Communication Table
```



```
HID      CVT+42C?      Cpu Information Iosdshid
ICT      RMCT+8?       Srm i/o Management Control Table
JCT      JSCB+104?     Job Control Table
JESCT    CVT+128?      Job Entry Subsystem Communication Table
JESPEXT  JESCT+64?     Pageable Jesct Extension
JSCB     TCB+B4?       Job/Step Control Block
JSTCB    TCB+7C?       Job Step Tcb
LLE      TCB+24?       Last Load List Element
LLT      CVT+4DC?      Link List Table
LLTX     CSVT+4?       Link List Table Volumes???
LPAQ     CVT+BC?       Lpa Cde List
LPAR     STGS+88?+334? Lpar information
LUV      EDT+10?+1C?
MCT      RMCT+10?      Srm Storage Management Control
OUCB     ASCB+90?      Resources Manager User Control Block
OUSB     ASXB+80?      Resources Manager User Swappable Block
OUXB     ASCB+94?      Resources Manager User Extension Block
PCCA     PSA+8?        Physical Configuration Communication Area
PIT      HCCT+5A8?     Initiator List (Changes Frequently With Jes)
STGS     CVT+31C?      Measurement Facility Control Block
PSA      0.            Prefixed Save Area
PSCB     JSCB+108?     Tso Protected Step Control Block
PVT      CVT+164?      Rsm Page Vector Table
RAB      ASCB+178?     Rsm Address Space Block
RAX      ASCB+16C?     Rsm Address Space Block Extension
RB       TCB?          Rb for this task
RCVT     CVT+3E0?
RMCA     RMCT+14?      System Resource Manager Control Area
RMCT     CVT+25C?      System Resources Manager Control Table
RMEX     RMCT+28?      Srm External Entry Poiny Descriptor Table
RTCT     CVT+23C?      Recovery/Termination Control Table
SCCB     CVT+340?      Service Call Control Block (Sccb)
SCT      JSCB+148?     Step Control Table
SCTX     SCT+44?       Step Control Table Extension
SCVT     CVT+C8?       Secondary Cvt
SHDR     CVT+250?
SJB      PIT+4?
SMCA     CVT+C4?       Smf Control Table
SMCX     SMCA+178?     Smca Extension
SPQA     SPQE+8?
SPQE     TCB+18?
SSCT     JESCT+18?     Same As Sscvt
SSCVT    JESCT+18?     Subsystem Communications Vector Table
SSIB     JSCB+13C?     Life of Job Subsystem Interface Block
SSVT     SSCVT+10?     Subsystem Vector Table
SVCTABLE SCVT+84?      Svc Table
SVCTAB2  SCVT+88?      Svc Update Recording Table
SVRB     ASCB+10?
TCB      CVT??         Task Control Block
TCBFSA   TCB+70?       Tcb First Save Area
TCT      TCB+A4?       Smf Timing Control Table
TIOT     TCB+C?        Task Input/Output Table
TSB      ASCB+3C?
TSVT     CVT+9C?
UPT      PSCB+34?
VTAMCVT  CVTEXT+40?
```



```
 BROWSE    STORAGE   START:008D2FD0                                                                                   10.???+C?
 COMMAND ===> _                                                                                               SCROLL ===> PAGE
*********************************************************** TOP OF DATA ************************************************************
      +0 (008D2FD0)  C1C2F2F2 F1F74040 C1C2F2F2 F1F74040  C1C2F2F2 F1F74040 14010100 E2E8E2E4  * AB2217  AB2217  AB2217  ....SYSU *
     +20 (008D2FF0)  C1C4E240 00006F00 80F4F1D0 14010100  E2E8E2D3 C2C34040 0000EF00 80F4F300  * ADS ..?.Ø41}....SYSLBC  ..Õ.Ø43. *
     +40 (008D3010)  14010100 E2E8E2D7 D9D6C340 00011F00  80F4F300 14010100 40404040 40404040  * ....SYSPROC ....Ø43.....         *
     +60 (008D3030)  00014F00 80F4F1D0 14010100 40404040  40404040 00017F00 80F4F1D0 14010100  * ..|.Ø41}....        ..".Ø41}.... *
     +80 (008D3050)  40404040 40404040 0001AF00 80F4F1D0  14010100 40404040 40404040 0001DF00  *         ..®.Ø41}....        ..Ÿ. *
     +A0 (008D3070)  80F4F1D0 14010100 40404040 40404040  00020F00 80F4F1D0 14010100 40404040  * Ø41}....        ....Ø41}....     *
     +C0 (008D3090)  40404040 00023F00 80F4F1D0 14010100  40404040 40404040 00026F00 80F4F1D0  *     ....Ø41}....        ..?.Ø41} *
     +E0 (008D30B0)  14010100 40404040 40404040 00029F00  80F4F1D0 14010100 40404040 40404040  * ....        ..¤.Ø41}....         *
    +100 (008D30D0)  0002DF00 80F4F1D0 14010100 40404040  40404040 00030F00 80F4F1D0 14010100  * ..Ÿ.Ø41}....        ....Ø41}.... *
    +120 (008D30F0)  40404040 40404040 00033F00 80F501D8  14010100 40404040 40404040 00036F00  *         ....Ø5.Q....        ..?. *
    +140 (008D3110)  80F4F268 14010100 40404040 40404040  00039F00 80F4F1D0 14010100 40404040  * Ø42Ç....        ..¤.Ø41}....     *
    +160 (008D3130)  40404040 0003CF00 80F4F300 14010100  40404040 40404040 0003FF00 80F4F1D0  *     ..õ.Ø43.....        ....Ø41} *
    +180 (008D3150)  14010100 40404040 40404040 00042F00  80F501D8 14010100 40404040 40404040  * ....        ....Ø5.Q....         *
    +1A0 (008D3170)  00045F00 80F501D8 14010100 E2E8E2C5  E7C5C340 00049F00 90F50B58 14010100  * ..^.Ø5.Q....SYSEXEC ..¤.°5.ì.... *
    +1C0 (008D3190)  40404040 40404040 0004CF00 80F4F1D0  14010100 40404040 40404040 0004FF00  *         ..õ.Ø41}....        .... *
    +1E0 (008D31B0)  80F4F1D0 14010100 40404040 40404040  00052F00 80F501D8 14010100 40404040  * Ø41}....        ....Ø5.Q....     *
    +200 (008D31D0)  40404040 00055F00 80F4F1D0 14010100  40404040 40404040 00058F00 80F501D8  *     ..^.Ø41}....        ..±.Ø5.Q *
    +220 (008D31F0)  14010100 40404040 40404040 0005BF00  80F4F268 14010100 E2E8E2C8 C5D3D740  * ....        ..×.Ø42Ç....SYSHELP  *
    +240 (008D3210)  0005EF00 80F4F268 14010100 40404040  40404040 00061F00 80F4F268 14010100  * ..Õ.Ø42Ç....        ....Ø42Ç.... *
    +260 (008D3230)  40404040 40404040 00064F00 80F501D8  14010100 40404040 40404040 00068F00  *         ..|.Ø5.Q....        ..±. *
    +280 (008D3250)  80F501D8 14010100 C9E2D7D4 D3C9C240  0006BF00 80F4F1D0 14010100 40404040  * Ø5.Q....ISPMLIB ..×.Ø41}....     *
    +2A0 (008D3270)  40404040 0006EF00 80F4F268 14010100  40404040 40404040 00071F00 80F4F1D0  *     ..Õ.Ø42Ç....        ....Ø41} *
    +2C0 (008D3290)  14010100 40404040 40404040 00074F00  80F4F1D0 14010100 40404040 40404040  * ....        ..|.Ø41}....         *
    +2E0 (008D32B0)  00077F00 80F4F1D0 14010100 40404040  40404040 0007AF00 80F4F1D0 14010100  * ..".Ø41}....        ..®.Ø41}.... *
    +300 (008D32D0)  40404040 40404040 0007DF00 80F4F1D0  14010100 40404040 40404040 00080F00  *         ..Ÿ.Ø41}....        .... *
    +320 (008D32F0)  80F501D8 14010100 40404040 40404040  00083F00 80F4F1D0 14010100 40404040  * Ø5.Q....        ....Ø41}....     *
    +340 (008D3310)  40404040 00087F00 80F4F1D0 14010100  40404040 40404040 0008AF00 80F4F1D0  *     ..".Ø41}....        ..®.Ø41} *
    +360 (008D3330)  14010100 40404040 40404040 0008DF00  80F4F1D0 14010100 40404040 40404040  * ....        ..Ÿ.Ø41}....         *
    +380 (008D3350)  00090F00 80F4F1D0 14010100 40404040  40404040 00093F00 80F4F268 14010100  * ....Ø41}....        ....Ø42Ç.... *
    +3A0 (008D3370)  40404040 40404040 00096F00 80F501D8  14010100 40404040 40404040 00099F00  *         ..?.Ø5.Q....        ..¤. *
    +3C0 (008D3390)  80F501D8 14010100 40404040 40404040  0009CF00 80F501D8 14010100 C9E2D7C5  * Ø5.Q....        ..õ.Ø5.Q....ISPE *
    +3E0 (008D33B0)  E7C5C340 0009FF00 80F4F1D0 14010100  40404040 40404040 000A3F00 80F4F1D0  * XEC ....Ø41}....        ....Ø41} *
    +400 (008D33D0)  14010100 40404040 40404040 000A6F00  80F501D8 14010100 C9E2D7D3 D3C9C240  * ....        ..?.Ø5.Q....ISPLLIB  *
 PF 1=HELP                2=SPLIT               3=END                4=RETURN              5=RFIND               6=RCHANGE
 PF 7=UP                  8=DOWN                9=SWAP              10=LEFT               11=RIGHT              12=RETRIEVE
```

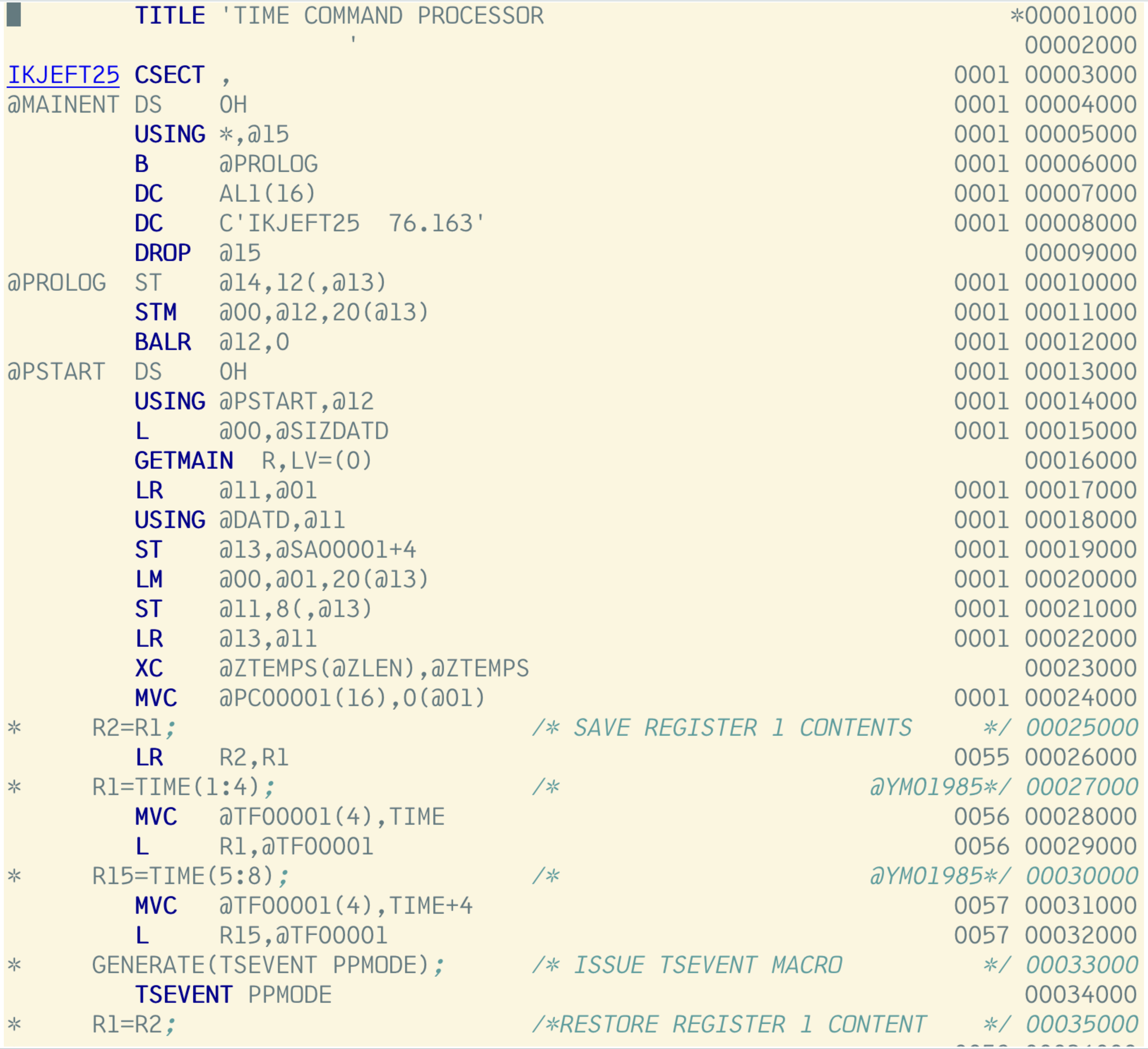


Goal	Command	Control Blocks
Show Master Catalog	B CVT+100?+8?+40?+34	CVT->AMBCS->ACB->CAXWA+X'34'
Show your RACF ACEE	B ACEE	

Look for some info (that is hard to get elsewhere)

Also modules; Zoom in on IKJEFT25

- IKJEFT25, the TSO TIME command
- Relevant for performance because it gives you spent service units
  - A service unit is a cpu-independent measure of resource usage
- browse IKJEFT25
- disasm



```
         TITLE 'TIME COMMAND PROCESSOR                                 *00001000
                        '                                               00002000
IKJEFT25 CSECT ,                                                  0001 00003000
@MAINENT DS    0H                                                 0001 00004000
         USING *,@15                                              0001 00005000
         B     @PROLOG                                            0001 00006000
         DC    AL1(16)                                            0001 00007000
         DC    C'IKJEFT25  76.163'                                0001 00008000
         DROP  @15                                                     00009000
@PROLOG  ST    @14,12(,@13)                                       0001 00010000
         STM   @00,@12,20(@13)                                    0001 00011000
         BALR  @12,0                                              0001 00012000
@PSTART  DS    0H                                                 0001 00013000
         USING @PSTART,@12                                        0001 00014000
         L     @00,@SIZDATD                                       0001 00015000
         GETMAIN  R,LV=(0)                                             00016000
         LR    @11,@01                                            0001 00017000
         USING @DATD,@11                                          0001 00018000
         ST    @13,@SA00001+4                                     0001 00019000
         LM    @00,@01,20(@13)                                    0001 00020000
         ST    @11,8(,@13)                                        0001 00021000
         LR    @13,@11                                            0001 00022000
         XC    @ZTEMPS(@ZLEN),@ZTEMPS                                  00023000
         MVC   @PC00001(16),0(@01)                                0001 00024000
*     R2=R1;                         /* SAVE REGISTER 1 CONTENTS     */ 00025000
         LR    R2,R1                                              0055 00026000
*     R1=TIME(1:4);                  /*                      @YM01985*/ 00027000
         MVC   @TF00001(4),TIME                                   0056 00028000
         L     R1,@TF00001                                        0056 00029000
*     R15=TIME(5:8);                 /*                      @YM01985*/ 00030000
         MVC   @TF00001(4),TIME+4                                 0057 00031000
         L     R15,@TF00001                                       0057 00032000
*     GENERATE(TSEVENT PPMODE);      /* ISSUE TSEVENT MACRO          */ 00033000
         TSEVENT PPMODE                                                00034000
*     R1=R2;                         /*RESTORE REGISTER 1 CONTENT    */ 00035000
```

Fortunately, we have the source of an older version

We can see:

- It is written in PL/S
- The eyecatcher says 76.163
- It is reenterable
- Register equates with @

In SYS1.LINKLIB we see that it has the attributes RF RE RU

Ministerie van Financiën

№ 5 View Control Blocks: IPCS

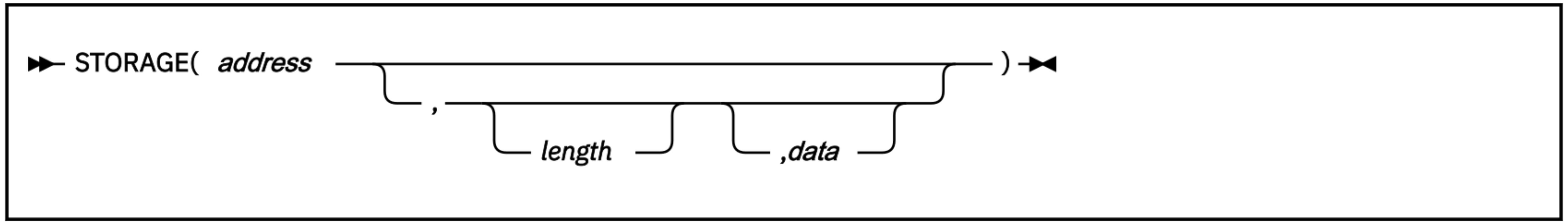
IPCS is the built-on dump analyzer of z/OS; it can also regard active memory as a dump dataset and format control blocks - and lots of other things, like running chains and catching ECB (POST/WAIT) problems with thread (TCB) locking. It has a great relevance for debugging this type of performance problem. On the other hand, nobody knows how to use it anymore and it is relegated to being a tool for the IBM CE; with sites that are read-protecting SYS1.PARMLIB you are out of luck because it needs to read its configuration from there. Already present in the first releases of MVS, and before those

Ministerie van Financiën

№ 6

Make Rexx programs with the STORAGE built-in function

STORAGE



STORAGE returns *length* bytes of data from the specified *address* in storage. The *address* is a character string containing the hexadecimal representation of the storage address from which data is retrieved.

The address can be a 31-bit address, represented by 1 to 8 hexadecimal characters. The address can also be a 64-bit address represented by 9 to 17 characters which consists of 8 to 16 hexadecimal characters plus an optional underscore ("_") character separating the high order half and low order half of the 64-bit address. If an "_" is part of the 64-bit address, it must be followed by exactly 8 hexadecimal digits in the low order (or right) half of the 64-bit address.

Optionally, you can specify *length*, which is the decimal number of bytes to be retrieved from *address*. The default *length* is one byte. When *length* is 0, STORAGE returns a null character string.

If you specify *data*, STORAGE returns the information from *address* and then overwrites the storage starting at *address* with *data* you specified on the function call. The *data* is the character string to be stored at *address*. The *length* argument has no effect on how much storage is overwritten; the entire *data* is written.

If the REXX environment under which STORAGE is executing is configured to allow STORAGE to run in read-only mode, then the STORAGE function can be used to read but not alter storage. In this case, do not specify a *data* argument. If you do specify a new value in the third argument while executing in read-only mode, error message IRX0241I will be issued and the STORAGE function will end in error.

You can use the STORAGE function in REXX execs that run in any MVS address space (TSO/E and non-TSO/E).

Examples:

The following are some examples of using STORAGE:

1. To retrieve 25 bytes of data from address 000AAE35, use the STORAGE function as follows:

```
storret = STORAGE(000AAE35,25)
```

2. To replace the data at address 0035D41F with ‘TSO/E REXX’, use the following STORAGE function:

```
storrep = STORAGE(0035D41F,,'TSO/E REXX')
```

   This example first returns one byte of information found at address 0035D41F and then replaces the data beginning at address 0035D41F with the characters ‘TSO/E REXX’.

   Note : Information is retrieved before it is replaced.

3. Some areas may be accessible to be fetched but not written. That storage can be read as the actual hex data. You can then use the X2D function to then display that hex data in displaceable character format.

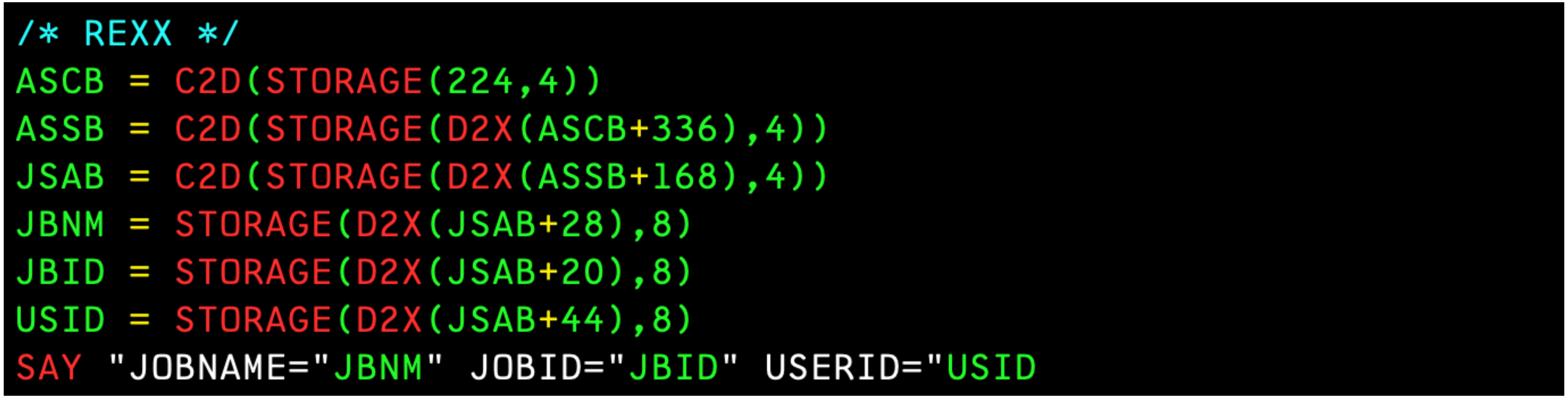
```
say '<'C2X(STORAGE(10,4))'>'         /* Returns <00FDC248>, perhaps.  This
                                     area in PSA is update protected, but
                                     not fetch protected.  The CVT addr.*/
```

   Trying to update this same area will fail because address x'10' is a write protected area in PSA at PSA +x'10'.

```
say '<'C2X<STORAGE(10,4,'XXXX'))'>'  /* Returns <> (a null string)
                                     because the storage at x'10' is at
                                     PSA+x'10' and is write protected and
                                     cannot be overwritten by STORAGE  */
```

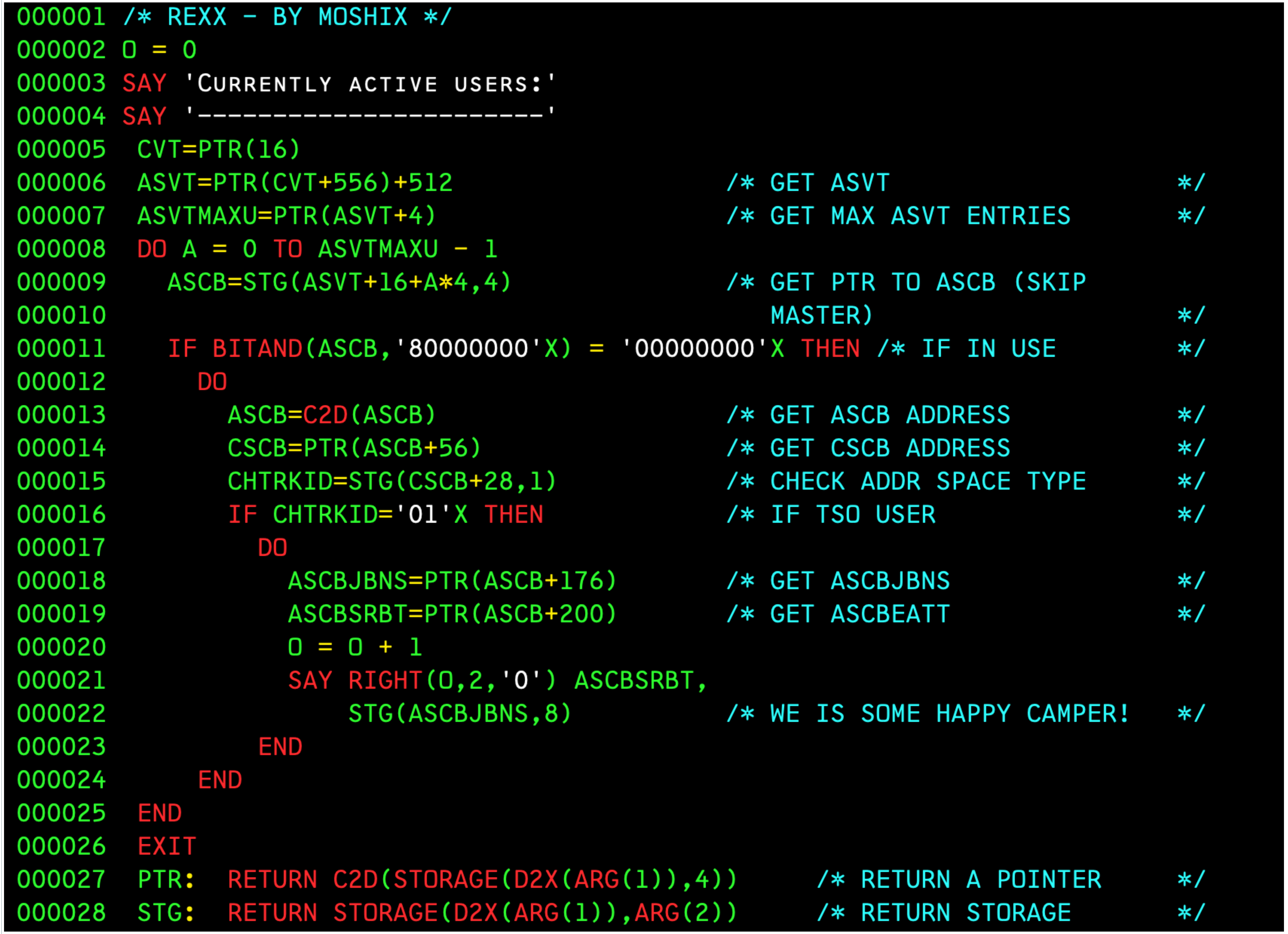
4. STORAGE can access 31-bit storage (including 24-bit areas), as well as 64-bit storage. The following shows some possible STORAGE addresses, and the resulting binary addresses that is actually accessed by the STORAGE function.

Simple Job Name exec (works on modern z/OS)



```
/* REXX */
ASCB = C2D(STORAGE(224,4))
ASSB = C2D(STORAGE(D2X(ASCB+336),4))
JSAB = C2D(STORAGE(D2X(ASSB+168),4))
JBNM = STORAGE(D2X(JSAB+28),8)
JBID = STORAGE(D2X(JSAB+20),8)
USID = STORAGE(D2X(JSAB+44),8)
SAY "JOBNAME="JBNM" JOBID="JBID" USERID="USID
```

There is more than one way that leads to Rome - this works on all known releases of MVS, OS/390 and z/OS



```
/* REXX - BY MOSHIX */
O = 0
SAY 'Currently active users:'
SAY '-----------------------'
 CVT=PTR(16)
 ASVT=PTR(CVT+556)+512                  /* GET ASVT                   */
 ASVTMAXU=PTR(ASVT+4)                   /* GET MAX ASVT ENTRIES       */
 DO A = 0 TO ASVTMAXU - 1
   ASCB=STG(ASVT+16+A*4,4)              /* GET PTR TO ASCB (SKIP
                                           MASTER)                    */
   IF BITAND(ASCB,'80000000'X) = '00000000'X THEN /* IF IN USE        */
     DO
       ASCB=C2D(ASCB)                   /* GET ASCB ADDRESS           */
       CSCB=PTR(ASCB+56)                /* GET CSCB ADDRESS           */
       CHTRKID=STG(CSCB+28,1)           /* CHECK ADDR SPACE TYPE      */
       IF CHTRKID='01'X THEN            /* IF TSO USER                */
         DO
           ASCBJBNS=PTR(ASCB+176)       /* GET ASCBJBNS               */
           ASCBSRBT=PTR(ASCB+200)       /* GET ASCBEATT               */
           O = O + 1
           SAY RIGHT(O,2,'0') ASCBSRBT,
               STG(ASCBJBNS,8)          /* WE IS SOME HAPPY CAMPER!   */
         END
     END
 END
 EXIT
 PTR:  RETURN C2D(STORAGE(D2X(ARG(1)),4))    /* RETURN A POINTER      */
 STG:  RETURN STORAGE(D2X(ARG(1)),ARG(2))    /* RETURN STORAGE        */
```

This lists all the active TSO users on the system (all address spaces where CSCB+28 contains a 01

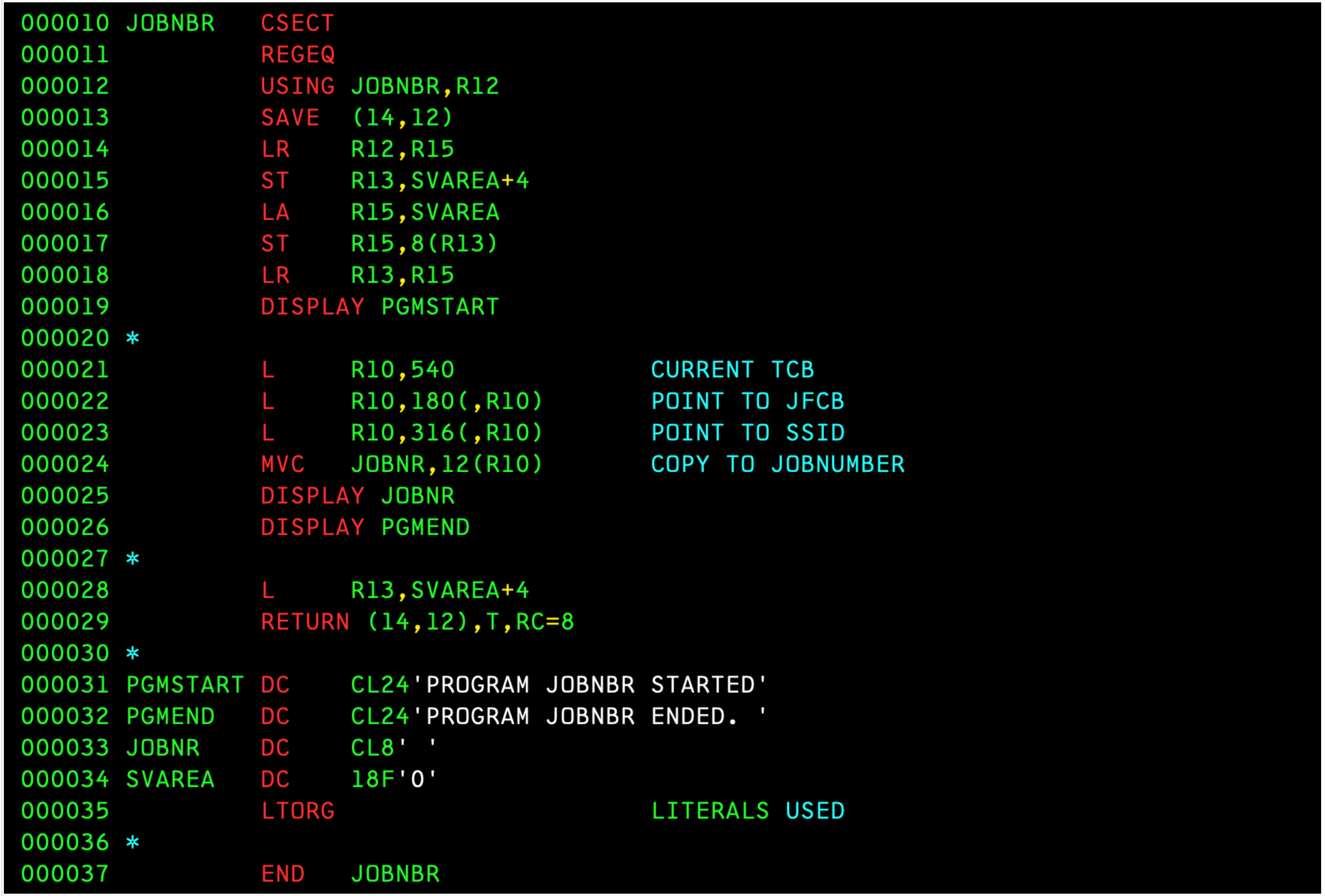
You can run that from USS also

- It's the same Rexx interpreter, with added functions in the ADDRESS SYSTEM environment

Ministerie van Financiën

№ 7 Macro mappings and Assembler

Assembler, plain - gets the current job number

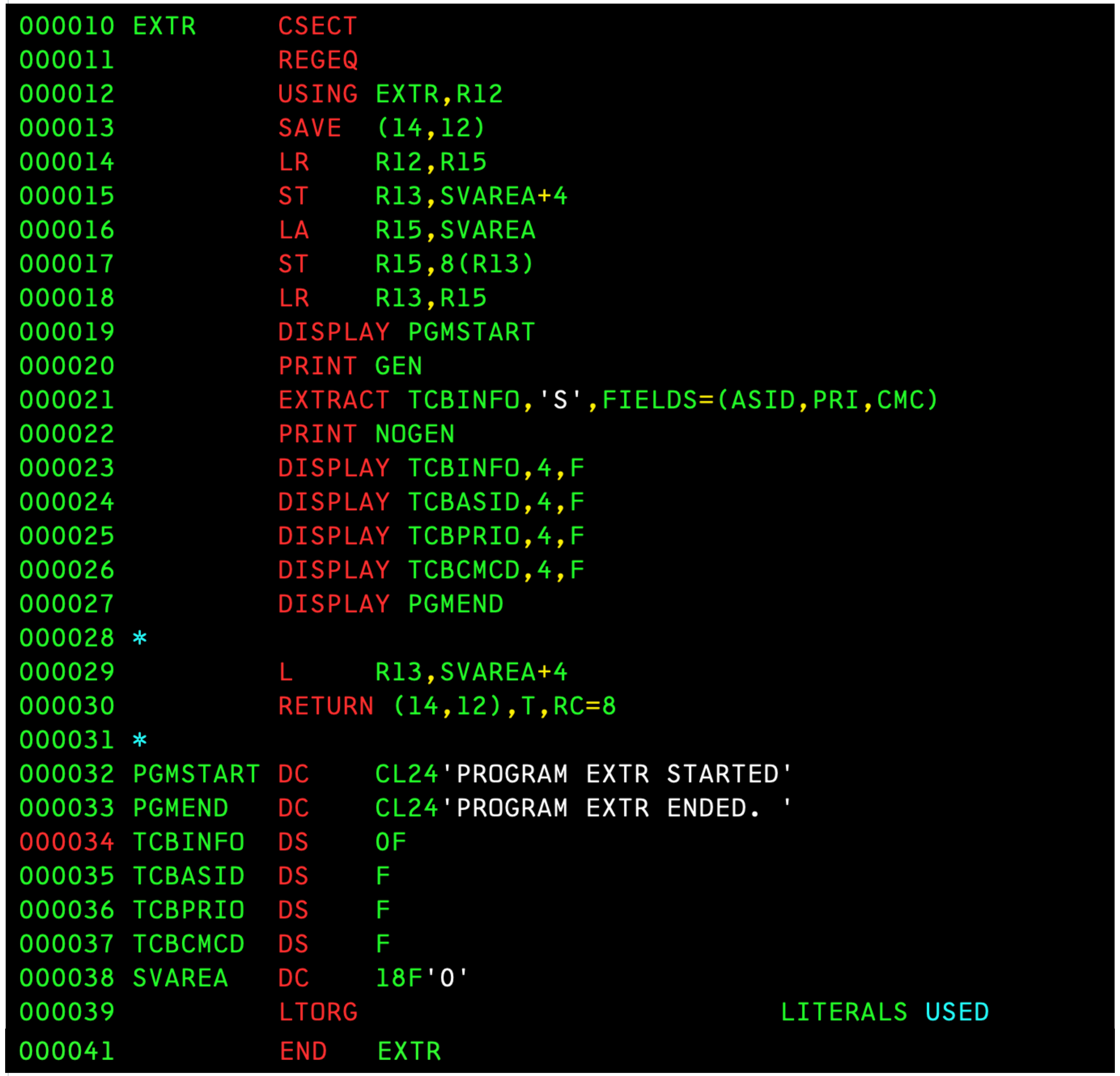


```
JOBNBR   CSECT
         REGEQ
         USING JOBNBR,R12
         SAVE  (14,12)
         LR    R12,R15
         ST    R13,SVAREA+4
         LA    R15,SVAREA
         ST    R15,8(R13)
         LR    R13,R15
         DISPLAY PGMSTART
*
         L     R10,540             CURRENT TCB
         L     R10,180(,R10)       POINT TO JFCB
         L     R10,316(,R10)       POINT TO SSID
         MVC   JOBNR,12(R10)       COPY TO JOBNUMBER
         DISPLAY JOBNR
         DISPLAY PGMEND
*
         L     R13,SVAREA+4
         RETURN (14,12),T,RC=8
*
PGMSTART DC    CL24'PROGRAM JOBNBR STARTED'
PGMEND   DC    CL24'PROGRAM JOBNBR ENDED. '
JOBNR    DC    CL8' '
SVAREA   DC    18F'0'
         LTORG                     LITERALS USED
*
         END   JOBNBR
```

Fully automated using an OS macro

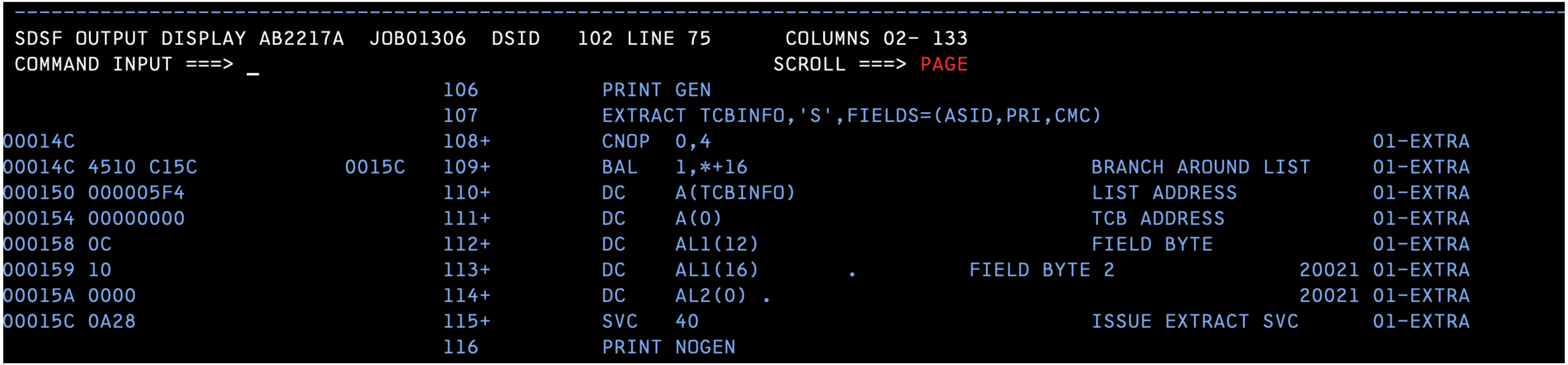
```
*
*
          EXTRACT TCBINFO,STCB1ADR,FIELDS=(ASID,PRI,CMC)
          ...
STCB1ADR DS    F
TCBINFO  DS    0F
TCBASID  DS    F
TCBPRIO  DS    F
TCBCMCD  DS    F
```

Assembler, using EXTRACT macro



```
EXTR     CSECT
         REGEQ
         USING EXTR,R12
         SAVE  (14,12)
         LR    R12,R15
         ST    R13,SVAREA+4
         LA    R15,SVAREA
         ST    R15,8(R13)
         LR    R13,R15
         DISPLAY PGMSTART
         PRINT GEN
         EXTRACT TCBINFO,'S',FIELDS=(ASID,PRI,CMC)
         PRINT NOGEN
         DISPLAY TCBINFO,4,F
         DISPLAY TCBASID,4,F
         DISPLAY TCBPRIO,4,F
         DISPLAY TCBCMCD,4,F
         DISPLAY PGMEND
*
         L     R13,SVAREA+4
         RETURN (14,12),T,RC=8
*
PGMSTART DC    CL24'PROGRAM EXTR STARTED'
PGMEND   DC    CL24'PROGRAM EXTR ENDED. '
TCBINFO  DS    0F
TCBASID  DS    F
TCBPRIO  DS    F
TCBCMCD  DS    F
SVAREA   DC    18F'0'
         LTORG                         LITERALS USED
         END   EXTR
```

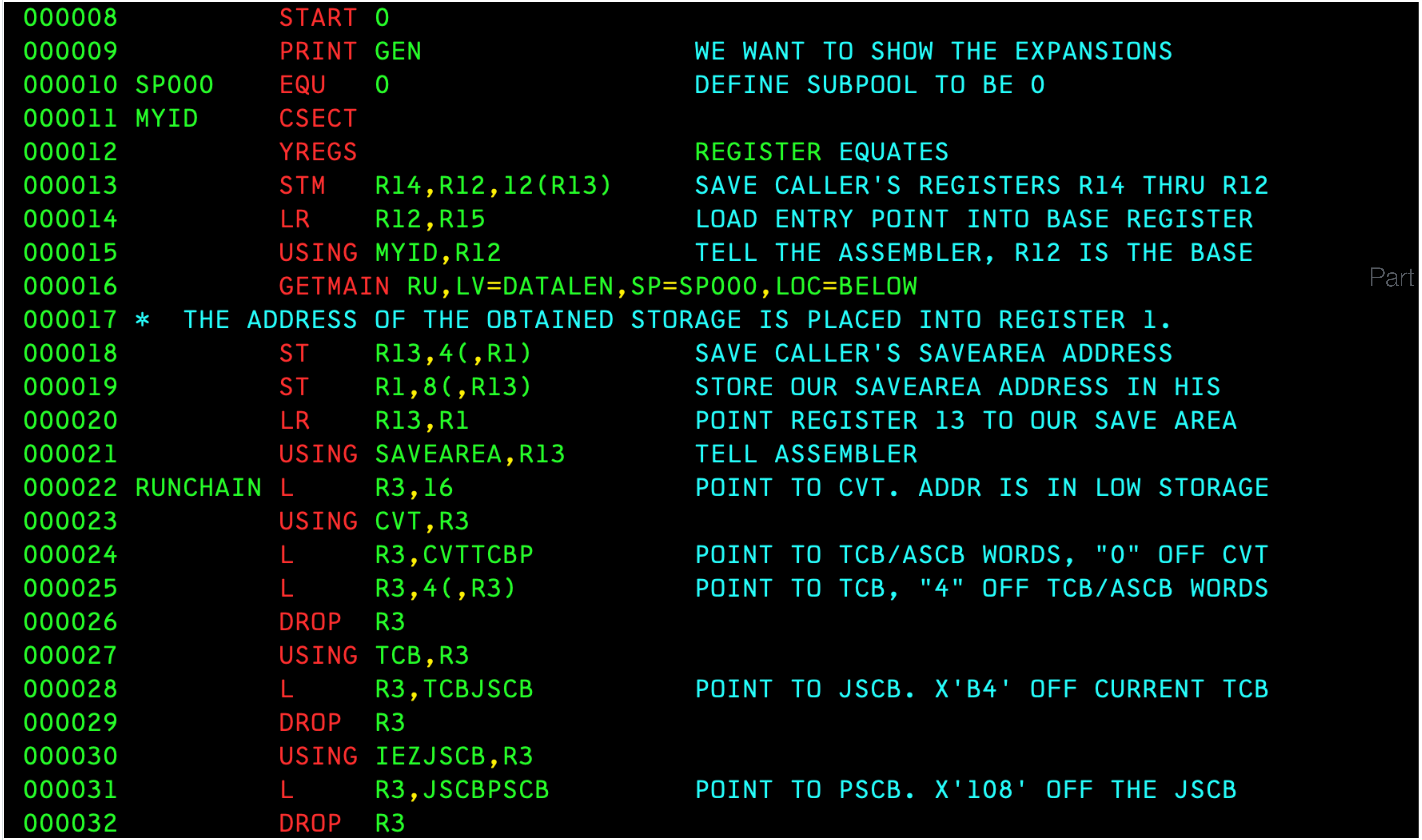
Assembler, EXTRACT macro expansion (SVC 40)



```
 -------------------------------------------------------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY AB2217A  JOB01306  DSID   102 LINE 75      COLUMNS 02- 133
 COMMAND INPUT ===> _                                          SCROLL ===> PAGE
                                    106          PRINT GEN
                                    107          EXTRACT TCBINFO,'S',FIELDS=(ASID,PRI,CMC)
00014C                              108+         CNOP  0,4                                                      01-EXTRA
00014C 4510 C15C            0015C   109+         BAL   1,*+16                            BRANCH AROUND LIST     01-EXTRA
000150 000005F4                     110+         DC    A(TCBINFO)                        LIST ADDRESS           01-EXTRA
000154 00000000                     111+         DC    A(0)                              TCB ADDRESS            01-EXTRA
000158 0C                           112+         DC    AL1(12)                           FIELD BYTE             01-EXTRA
000159 10                           113+         DC    AL1(16)       .         FIELD BYTE 2               20021 01-EXTRA
00015A 0000                         114+         DC    AL2(0) .                                           20021 01-EXTRA
00015C 0A28                         115+         SVC   40                                ISSUE EXTRACT SVC      01-EXTRA
                                    116          PRINT NOGEN
```

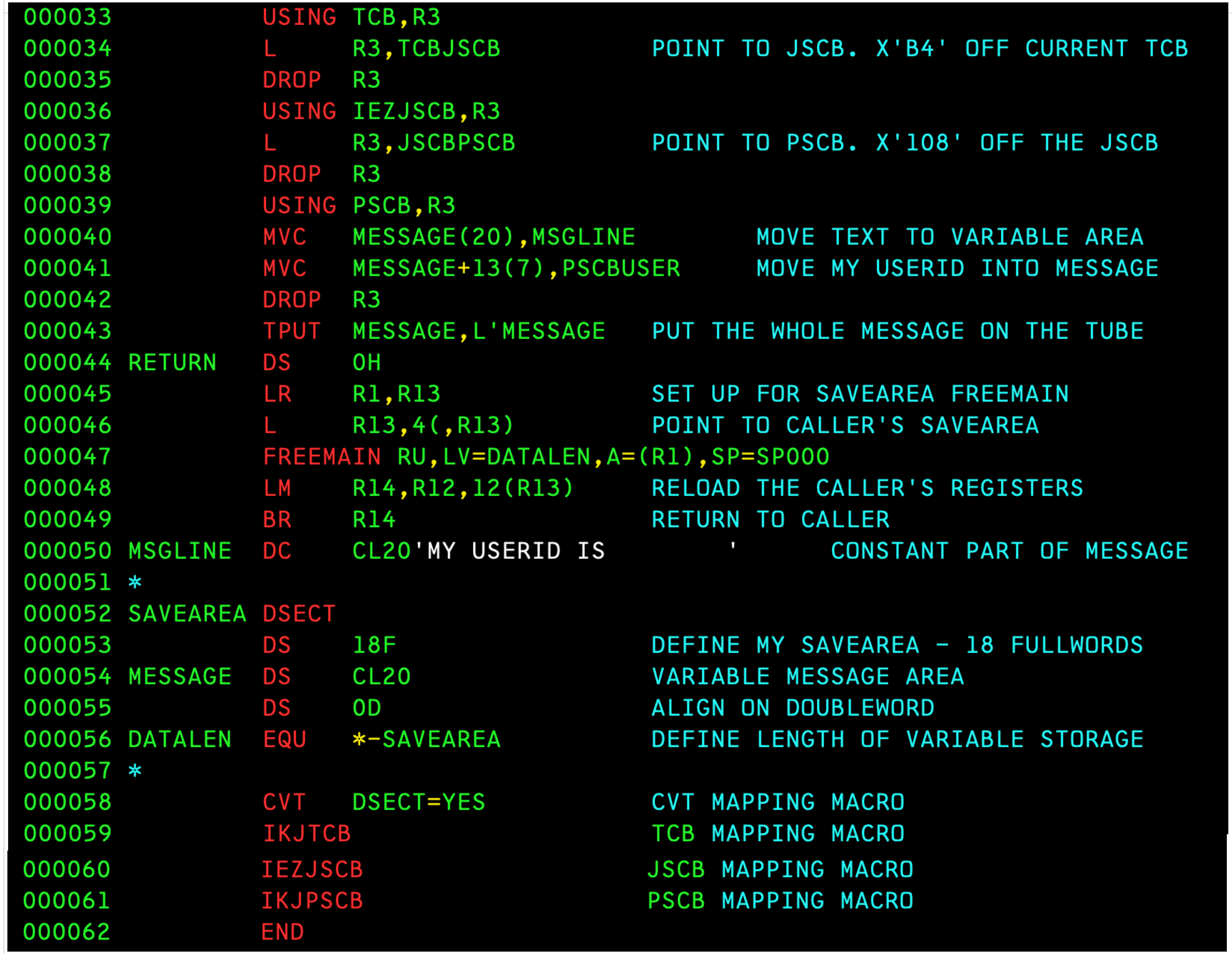
The super-duper macro version

- The next slide has the best, most stable version
    - It uses IBM provided macros and mapping
    - So the blocks and offsets might change, but the program keeps working
    - There is not a lot of counting or manual mapping involved
        - Lazy is always better
    - This program is exclusively for TSO (or TSO in Batch) due to the use of the TPUT macro for terminal I/O



```
         START 0
         PRINT GEN                 WE WANT TO SHOW THE EXPANSIONS
SP000    EQU   0                   DEFINE SUBPOOL TO BE 0
MYID     CSECT
         YREGS                     REGISTER EQUATES
         STM   R14,R12,12(R13)     SAVE CALLER'S REGISTERS R14 THRU R12
         LR    R12,R15             LOAD ENTRY POINT INTO BASE REGISTER
         USING MYID,R12            TELL THE ASSEMBLER, R12 IS THE BASE
         GETMAIN RU,LV=DATALEN,SP=SP000,LOC=BELOW
*  THE ADDRESS OF THE OBTAINED STORAGE IS PLACED INTO REGISTER 1.
         ST    R13,4(,R1)          SAVE CALLER'S SAVEAREA ADDRESS
         ST    R1,8(,R13)          STORE OUR SAVEAREA ADDRESS IN HIS
         LR    R13,R1              POINT REGISTER 13 TO OUR SAVE AREA
         USING SAVEAREA,R13        TELL ASSEMBLER
RUNCHAIN L     R3,16               POINT TO CVT. ADDR IS IN LOW STORAGE
         USING CVT,R3
         L     R3,CVTTCBP          POINT TO TCB/ASCB WORDS, "0" OFF CVT
         L     R3,4(,R3)           POINT TO TCB, "4" OFF TCB/ASCB WORDS
         DROP  R3
         USING TCB,R3
         L     R3,TCBJSCB          POINT TO JSCB. X'B4' OFF CURRENT TCB
         DROP  R3
         USING IEZJSCB,R3
         L     R3,JSCBPSCB         POINT TO PSCB. X'108' OFF THE JSCB
         DROP  R3
```

Part 1

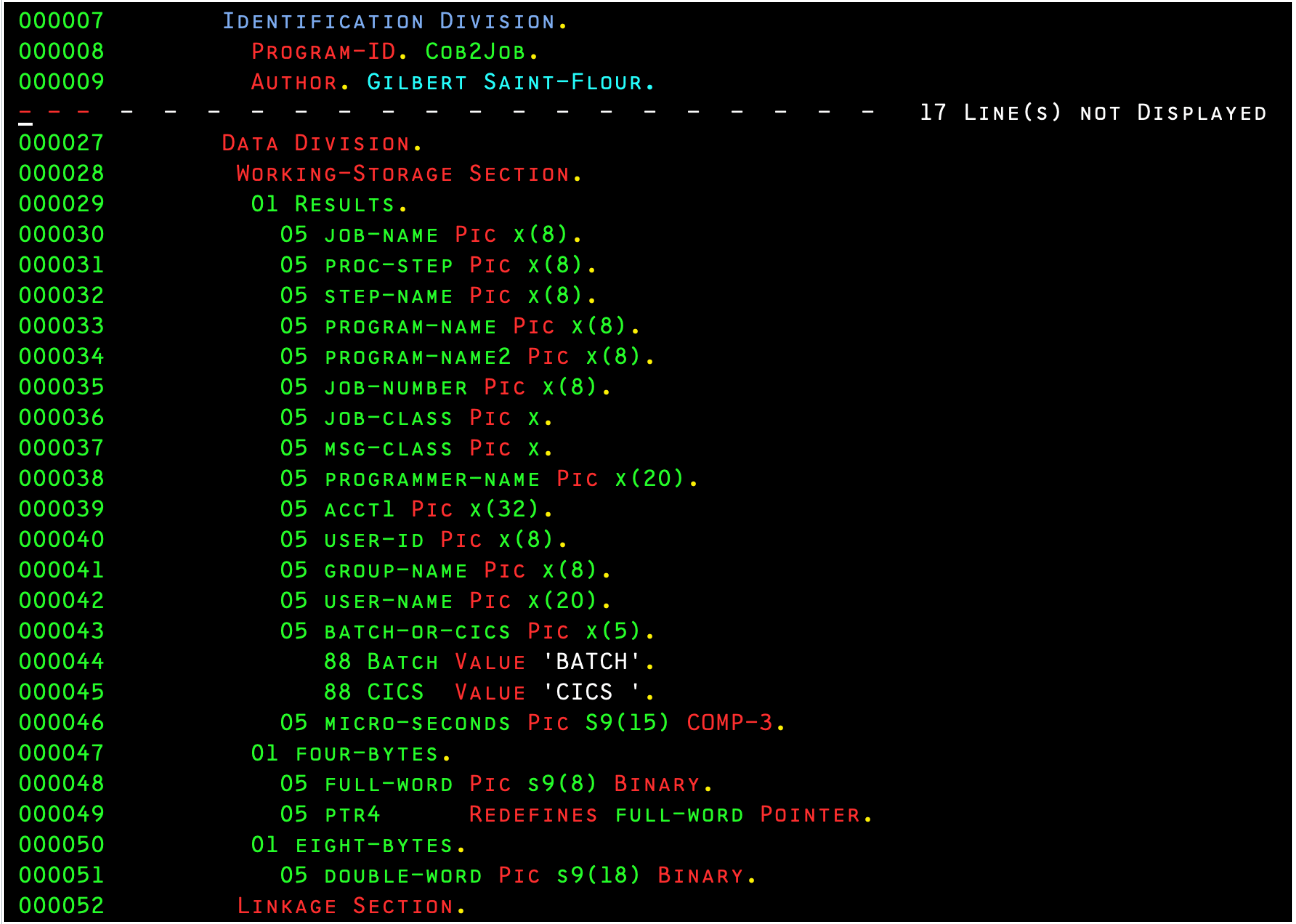


```
         USING TCB,R3
         L     R3,TCBJSCB          POINT TO JSCB. X'B4' OFF CURRENT TCB
         DROP  R3
         USING IEZJSCB,R3
         L     R3,JSCBPSCB         POINT TO PSCB. X'108' OFF THE JSCB
         DROP  R3
         USING PSCB,R3
         MVC   MESSAGE(20),MSGLINE        MOVE TEXT TO VARIABLE AREA
         MVC   MESSAGE+13(7),PSCBUSER     MOVE MY USERID INTO MESSAGE
         DROP  R3
         TPUT  MESSAGE,L'MESSAGE   PUT THE WHOLE MESSAGE ON THE TUBE
RETURN   DS    0H
         LR    R1,R13              SET UP FOR SAVEAREA FREEMAIN
         L     R13,4(,R13)         POINT TO CALLER'S SAVEAREA
         FREEMAIN RU,LV=DATALEN,A=(R1),SP=SP000
         LM    R14,R12,12(R13)     RELOAD THE CALLER'S REGISTERS
         BR    R14                 RETURN TO CALLER
MSGLINE  DC    CL20'MY USERID IS        '      CONSTANT PART OF MESSAGE
*
SAVEAREA DSECT
         DS    18F                 DEFINE MY SAVEAREA - 18 FULLWORDS
MESSAGE  DS    CL20                VARIABLE MESSAGE AREA
         DS    0D                  ALIGN ON DOUBLEWORD
DATALEN  EQU   *-SAVEAREA          DEFINE LENGTH OF VARIABLE STORAGE
*
         CVT   DSECT=YES           CVT MAPPING MACRO
         IKJTCB                    TCB MAPPING MACRO
         IEZJSCB                   JSCB MAPPING MACRO
         IKJPSCB                   PSCB MAPPING MACRO
         END
```

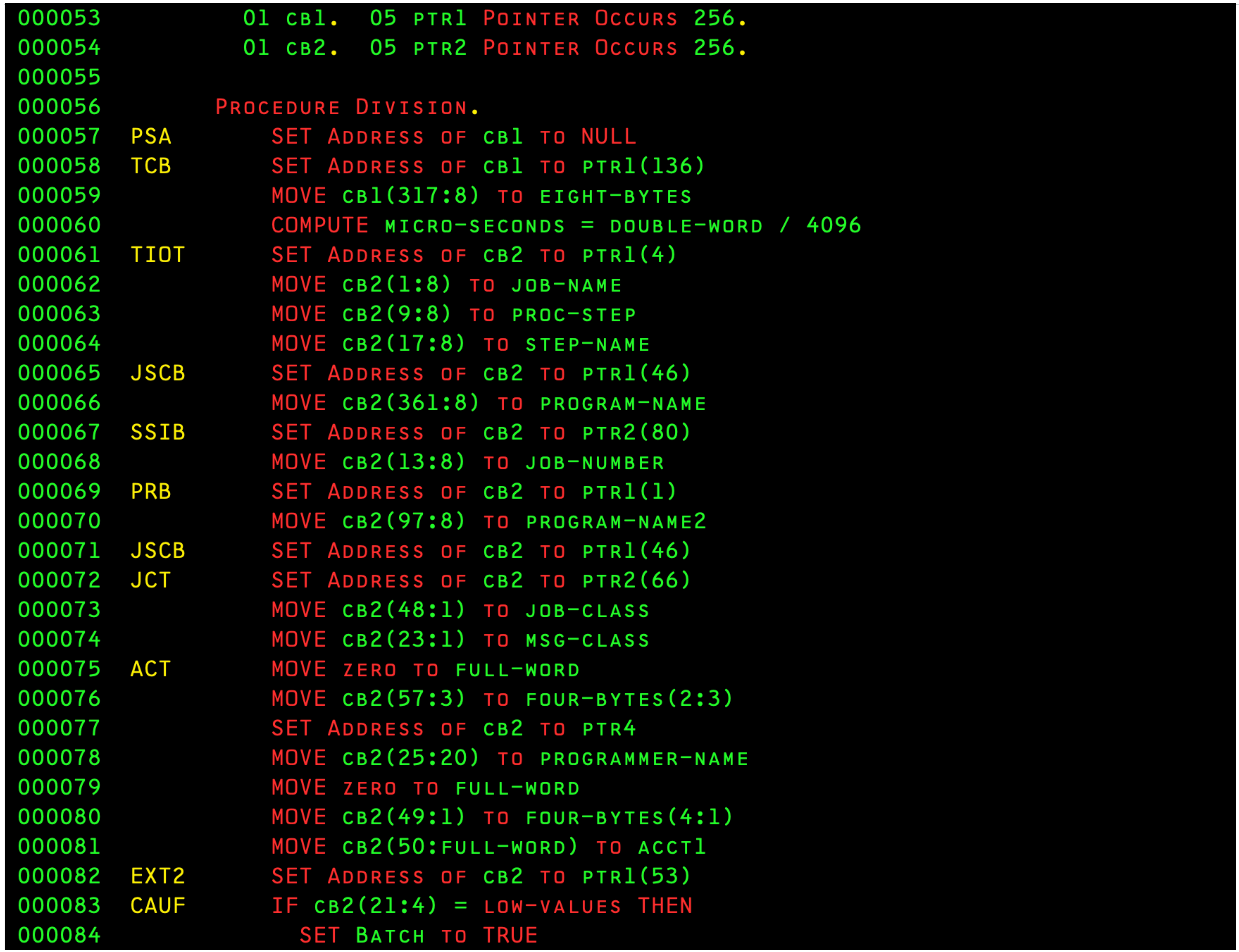
Part 2

Ministerie van Financiën

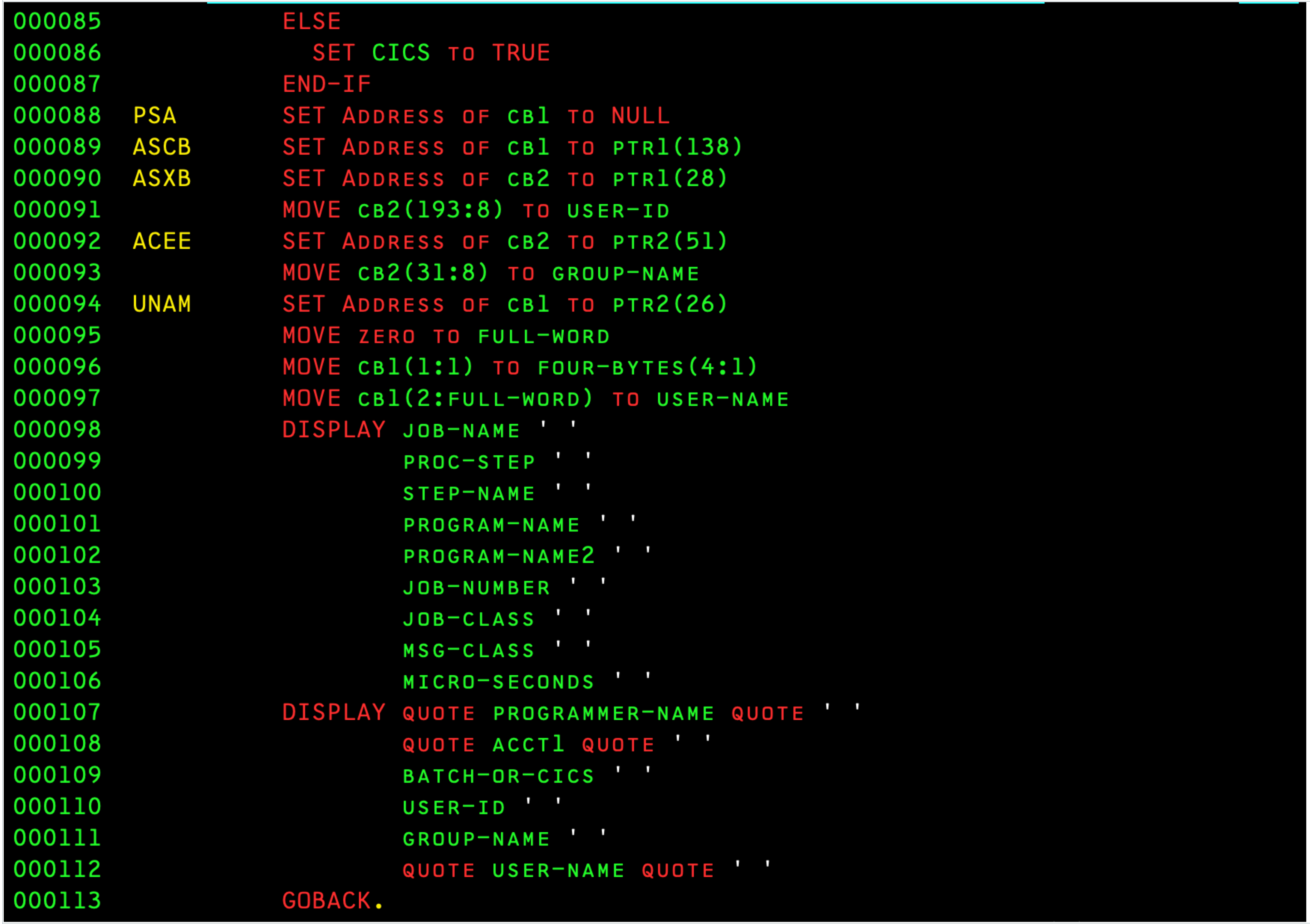
№ 8 … or COBOL



```
000007        Identification Division.
000008          Program-ID. Cob2Job.
000009          Author. Gilbert Saint-Flour.
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -   17 Line(s) not Displayed
000027        Data Division.
000028         Working-Storage Section.
000029          01 Results.
000030            05 job-name Pic x(8).
000031            05 proc-step Pic x(8).
000032            05 step-name Pic x(8).
000033            05 program-name Pic x(8).
000034            05 program-name2 Pic x(8).
000035            05 job-number Pic x(8).
000036            05 job-class Pic x.
000037            05 msg-class Pic x.
000038            05 programmer-name Pic x(20).
000039            05 acct1 Pic x(32).
000040            05 user-id Pic x(8).
000041            05 group-name Pic x(8).
000042            05 user-name Pic x(20).
000043            05 batch-or-cics Pic x(5).
000044               88 Batch Value 'BATCH'.
000045               88 CICS  Value 'CICS '.
000046            05 micro-seconds Pic S9(15) COMP-3.
000047          01 four-bytes.
000048            05 full-word Pic s9(8) Binary.
000049            05 ptr4      Redefines full-word Pointer.
000050          01 eight-bytes.
000051            05 double-word Pic s9(18) Binary.
000052         Linkage Section.
```



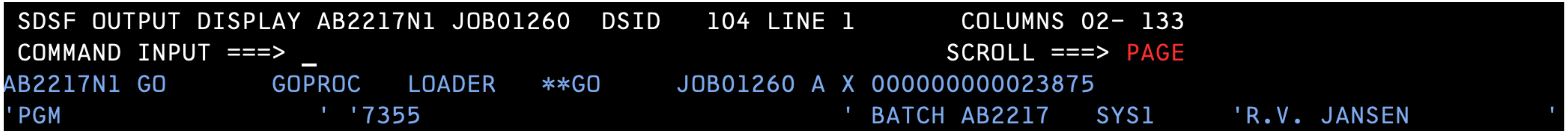
```
000053                01 cb1.  05 ptr1 Pointer Occurs 256.
000054                01 cb2.  05 ptr2 Pointer Occurs 256.
000055
000056              Procedure Division.
000057  PSA             SET Address of cb1 to NULL
000058  TCB             SET Address of cb1 to ptr1(136)
000059                  MOVE cb1(317:8) to eight-bytes
000060                  COMPUTE micro-seconds = double-word / 4096
000061  TIOT            SET Address of cb2 to ptr1(4)
000062                  MOVE cb2(1:8) to job-name
000063                  MOVE cb2(9:8) to proc-step
000064                  MOVE cb2(17:8) to step-name
000065  JSCB            SET Address of cb2 to ptr1(46)
000066                  MOVE cb2(361:8) to program-name
000067  SSIB            SET Address of cb2 to ptr2(80)
000068                  MOVE cb2(13:8) to job-number
000069  PRB             SET Address of cb2 to ptr1(1)
000070                  MOVE cb2(97:8) to program-name2
000071  JSCB            SET Address of cb2 to ptr1(46)
000072  JCT             SET Address of cb2 to ptr2(66)
000073                  MOVE cb2(48:1) to job-class
000074                  MOVE cb2(23:1) to msg-class
000075  ACT             MOVE zero to full-word
000076                  MOVE cb2(57:3) to four-bytes(2:3)
000077                  SET Address of cb2 to ptr4
000078                  MOVE cb2(25:20) to programmer-name
000079                  MOVE zero to full-word
000080                  MOVE cb2(49:1) to four-bytes(4:1)
000081                  MOVE cb2(50:full-word) to acct1
000082  EXT2            SET Address of cb2 to ptr1(53)
000083  CAUF            IF cb2(21:4) = low-values THEN
000084                    SET Batch to TRUE
```



```
          ELSE
            SET CICS TO TRUE
          END-IF
PSA       SET ADDRESS OF CB1 TO NULL
ASCB      SET ADDRESS OF CB1 TO PTR1(138)
ASXB      SET ADDRESS OF CB2 TO PTR1(28)
          MOVE CB2(193:8) TO USER-ID
ACEE      SET ADDRESS OF CB2 TO PTR2(51)
          MOVE CB2(31:8) TO GROUP-NAME
UNAM      SET ADDRESS OF CB1 TO PTR2(26)
          MOVE ZERO TO FULL-WORD
          MOVE CB1(1:1) TO FOUR-BYTES(4:1)
          MOVE CB1(2:FULL-WORD) TO USER-NAME
          DISPLAY JOB-NAME ' '
                  PROC-STEP ' '
                  STEP-NAME ' '
                  PROGRAM-NAME ' '
                  PROGRAM-NAME2 ' '
                  JOB-NUMBER ' '
                  JOB-CLASS ' '
                  MSG-CLASS ' '
                  MICRO-SECONDS ' '
          DISPLAY QUOTE PROGRAMMER-NAME QUOTE ' '
                  QUOTE ACCT1 QUOTE ' '
                  BATCH-OR-CICS ' '
                  USER-ID ' '
                  GROUP-NAME ' '
                  QUOTE USER-NAME QUOTE ' '
          GOBACK.
```

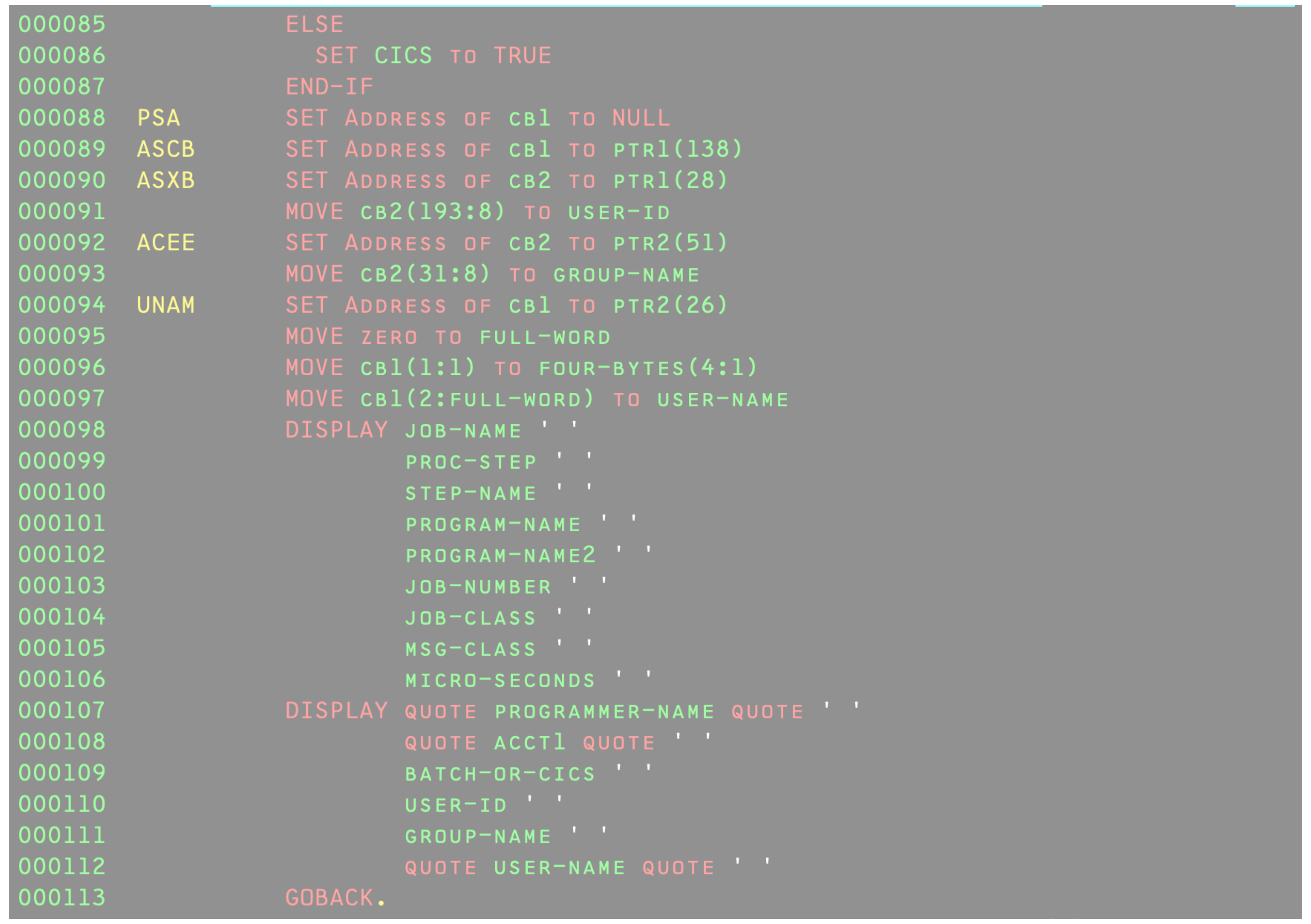
CICS or Batch

Output



```
 SDSF OUTPUT DISPLAY AB2217N1 JOB01260  DSID   104 LINE 1        COLUMNS 02- 133
 COMMAND INPUT ===> _                                           SCROLL ===> PAGE
AB2217N1 GO        GOPROC   LOADER   **GO     JOB01260 A X 000000000023875
'PGM                     ' '7355                          ' BATCH AB2217   SYS1     'R.V. JANSEN          '
```

Which, like always, is a lot of source code for one line of output. But that is the charm of COBOL: no documentation needed.



```
              ELSE
                SET CICS TO TRUE
              END-IF
       PSA    SET ADDRESS OF CB1 TO NULL
       ASCB   SET ADDRESS OF CB1 TO PTR1(138)
       ASXB   SET ADDRESS OF CB2 TO PTR1(28)
              MOVE CB2(193:8) TO USER-ID
       ACEE   SET ADDRESS OF CB2 TO PTR2(51)
              MOVE CB2(31:8) TO GROUP-NAME
       UNAM   SET ADDRESS OF CB1 TO PTR2(26)
              MOVE ZERO TO FULL-WORD
              MOVE CB1(1:1) TO FOUR-BYTES(4:1)
              MOVE CB1(2:FULL-WORD) TO USER-NAME
              DISPLAY JOB-NAME ' '
                      PROC-STEP ' '
                      STEP-NAME ' '
                      PROGRAM-NAME ' '
                      PROGRAM-NAME2 ' '
                      JOB-NUMBER ' '
                      JOB-CLASS ' '
                      MSG-CLASS ' '
                      MICRO-SECONDS ' '
              DISPLAY QUOTE PROGRAMMER-NAME QUOTE ' '
                      QUOTE ACCT1 QUOTE ' '
                      BATCH-OR-CICS ' '
                      USER-ID ' '
                      GROUP-NAME ' '
                      QUOTE USER-NAME QUOTE ' '
              GOBACK.
```

The end.

Q?: rv.jansen@xs4all.nl