#19 zTidBitz (V=R terminology)

Inquiry *on the Introduction to Mainframe: z/OS Basic (SG24-6366) publication from Marist College, Large Systems Education Department*
I'm not sure what the term V=R user means - sounds like VM terminology. How does DAT protect address spaces? Can you elaborate? If not you, who? Thanks for any help.

Response

Never recommended for user use, but in the past it was possible to run a job in only central storage with no I/O paging at all. The virtual addresses are equal to the central storage addresses. This is referred to as running in VIRTUAL=REAL or V=R. I recall (circa early ESA) some jobs used this to gain performance benefits or to analyze error conditions without having to page. If you go back a ways diagnostic tools used by SE's used this in order to analyze address translation problems. Also older devices had critical timing requirements (like optical readers) and programs accessing these devices might run as V=R. How jobs were assigned this was on the JOB or EXEC card with ADDRSPC=REAL.

Some background -
A real address identifies a location in real storage. When a real address is used for an access to main storage it is converted, by means of prefixing, to an absolute address. At any instant there is one real address-to-absolute address mapping for each CP in the configuration. When a real address is used by a CP to access main storage, it is converted to an absolute address by prefixing. Storage consisting of byte locations sequenced according to their real addresses is referred to as real storage.

Virtual address to main storage - when a virtual address is used by a CP to access main storage it is first converted, by means of dynamic address translation (DAT), to a real address, and then, by means of prefixing, to an absolute address. DAT may use from five to two levels of tables (region first table (R1 index), region second table, region third table, segment table, and page table) as transformation parameters. The designation (origin and length) of the highest-level table for a specific address space is called an address-space-control element, and it is found for use by DAT in a control register or as specified by an access register. Alternatively, the address-space-control element for an
address space may be a real space designation, which indicates that DAT is to translate the virtual address simply by treating it as a real address and without using any tables.

A reference to a given virtual storage address will cause dynamic translation to access the Segment Table (Region Table if 64-bit) , obtain the page table address and then get a central storage address of the page frame. DAT provides the ability to interrupt the execution of a program at an arbitrary moment, record it and its data

on auxiliary storage, and at a later time return the program and the data to different main-storage locations for resumption of execution.

Four protection facilities are provided to protect the contents of main storage (inc. address spaces) from destruction or misuse by programs that contain errors or are unauthorized: key-controlled protection, access-list-controlled protection, page protection, and low-address protection. Each protection facility is applied independently.

In general, storage protection imposes limits and a task running in an address space is only able to access (for read or write) the central storage locations with its own data and programs, or, if specifically allowed, to read areas from other tasks. Any violation of this rule causes the CPU to generate a program interrupt 0004 (protection exception), that makes z/OS ABEND that task.

All real addresses manipulated by CPUs (or channels for that matter) go through the storage protection verification before being used as an argument to access the contents of central storage. The input of storage protection is a real storage address and if the output is 'OK' it performs the instruction, if not a program interrupt will occur.

For each 4 KB block of central storage, there is a 7-bit control field, called a *storage key.* There are access control bits, bits 0-3 are matched against the 4-bit protection key in the program status word (PSW) whenever information is stored, or whenever information is fetched from a location that is protected against fetching.

There are protection keys provided by the PSW, and they are matched against the access control bits in the storage key. The storage protection implementation formats central (real) storage into 4 KB frames. For each 4 KB frame, there is a 7-bit storage key. Each storage key has: access control bits, fetch bit,reference bit, and change bit.

When key-controlled protection applies to a storage being accessed, a store is permitted only when the storage key matches the access key associated with the request for storage access. A fetch is permitted when the keys match or when the fetch-protection bit of the storage key is zero. The keys are said to match when the four access control bits of the storage key are equal to the access key, or when the access key is zero.

When the access to storage is initiated by the CPU and key-controlled protection applies, the PSW key becomes the access key. Bit 5 is associated with dynamic address translation (DAT). It is normally set to one whenever a location in the related 4 KB storage block is referred to for either storing or fetching of

information.Bit 6 is also associated with DAT. It is set to one each time that information is stored into the corresponding 4 KB block of storage.

The PSW key field (bits 8 to 11) is compared to the access control bits for the CSTOR frame being referenced for storage access. z/OS exploits storage protection by managing frame storage key values and running the program PSW key field in the current PSW; for example, several z/OS routines run with PSW key zero, and others run with PSW key one. Note all application code has PSW key of eight.