Addressing to Achieve Security (and More)

(Abstract for the ASPLOS Wild and Crazy Ideas Session)

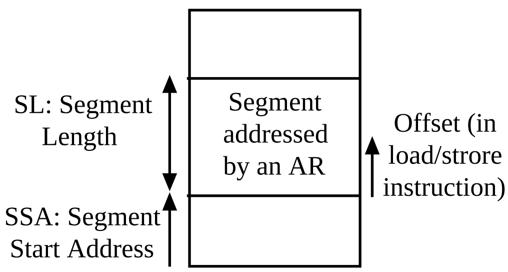
J. Leslie Keedy, University of Ulm, Germany

The computer security problem is increasing daily in importance, but architectural research in this area has stagnated since RISC systems killed research into capability designs. The main idea of this proposal is to rekindle this research area *without giving up the advantages of RISC*. The proposal is also relevant for other problem areas of memory addressing, including persistent virtual memory, distributed virtual memory and single address space parallel computers.

The basic idea is to use an addressing mechanism in which virtual addressing proceeds via special addressing registers (ARs) containing the following fields:

Basic Access Rights	ASI: Address Space Identifier	SSA: Segment Start Address (say 32 bits)	SL: Segment Length (say 32 bits)

The virtual space may be *very* large (and sharable by many computers). It is decomposed into a number of discrete address spaces. Each address space (max. length 2^{32} or more) is decomposed into logical segments. An AR defines the start of a segment (as an offset into an address space) and its length. A segment can be as large as an address space or just a few bytes long.

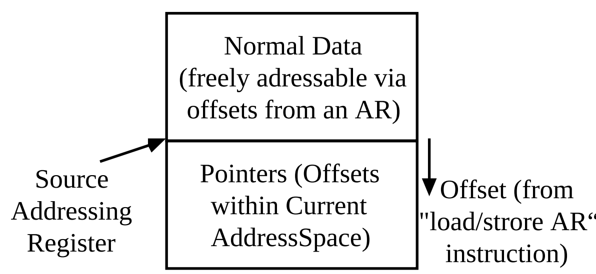


An Address SpaceIdentified by ASI

Machine instructions are basically as in a RISC, except that the load/store instructions specify an address which reduces to «AR, segment offset». To check segment bounds, load/store instructions compare the segment offset with SL. The access rights field is used to check basic access rights.

Paging and segmentation are orthogonal (i.e. the address derived from adding an offset in a load/store instruction to SSA is translated into a page frame number and offset in page). A segment may cross pages; a page may contain several segments.

ASI may be very large (e.g. 64 or more bits) in order to allow for unique addresses. This gives the advantages of capability addressing (cf. [1]) – both locally and over many nodes. ASI may *at the kernel level* be viewed as structured (e.g. into <node#, disk#, address space# in disk>).



This addressing technique allows the protection features of *all* of the earlier capability systems, such as [2, 3, 4], to be straightforwardly simulated in a kernel which has control over the loading of ARs *and* there is the added benefit that it is implemented in a RISC environment.

For example the kernel may support partitioned segments [5] by providing "load/store AR" operations in which an offset from an AR is treated as an index into the pointer section of a segment. These pointers need only be ca. 32 bits long, as the destination AR can copy the ASI (and usually the access rights) from the Source AR. SL can be taken from further admin. info. in the segment.

Depending on the computational/protection model, the ASI *need not be* a long unique identifier, since at a particular point in a computation only a few related address spaces are needed (e.g. parameters from the calling module). This allows a much smaller AR structure and creates possibilities for optimising the hardware implementation (for example using a TLB as in the DEC Alpha). Many other optimisations are also possible (e.g. parallel bounds checking and cache access, use of PALcode or similar for "load/ store AR" instructions, a separate memory management unit which also handles ARs). — Remember, the CDC 6600 (the spiritual father of RISC) had separate addressing registers!

In allowing unique virtual addressing over *all* the data of a system (even using a small ASI), the model provides a sound basis for designing a persistent virtual memory and hence for efficiently implementing persistent programming [6]. This in turn allows a simplified operating/file system and database environment. Similarly the model can be used as a basis for paging over a network and provides a suitable addressing environment for single address space parallel computers. But its most important property is that it allows the kind of experimentation with protection models which was destroyed in the early 1980s without forgoing the benefits of RISC.

References

1. R. S. Fabry "Capability Based Addressing", Comm. ACM, 17, 7, pp. 403-412, 1974.
2. W. A. Wulf, E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson and F. Pollack "HYDRA: The Kernel of a Multiprocessor Operating System", Comm. ACM, 17, 3, pp. 336-345, 1974.
3. M. V. Wilkes and R. M. Needham, The Cambridge CAP Computer and its Operating System, Oxford: North Holland, 1979.
4. B. W. Lampson and H. Sturgis "Reflections on an Operating System Design", Comm. ACM, 19, 5, pp. 251-265, 1976.
5. A. K. Jones "Capability Architecture Revisited", ACM Operating Systems Review, 14, 3, pp. 33-35, 1980.
6. M. P. Atkinson, P. Bailey, K. J. Chisholm, W. P. Cockshott and R. Morrison "An Approach to Persistent Programming", The Computer Journal, 26, 4, pp. 360-365, 1983.