Solution notes

Operating Systems (Part IA) 2005 – Paper 1 Question 4 (SMH)

This concerns memory management and, in particular, one of the motivations for virtual addressing. The "address binding problem" is explicitly mentioned in the notes. The answers below give more detail than I would expect from the students.

The address binding problem

This is the problem of coordinating the memory addresses issued by a program (e.g. accessing variables, making function calls) with the correct addresses to be used at run time.

Solutions to the address binding problem

One mark for the solution, one mark for an advantage, and one for a disadvantage.

We can solve this at compile time by enforcing a load address a priori (e.g. as for DOS .com files)[1]. An advantage is that this is very simple (and requires no load time or run time support). Major disadvantage is inflexiblity; it is not possible to load or run a program unless sufficient memroy at the required location is available.

We can solve this at load time by *relocating* – i.e. the loader patches up addresses once it knows the base address at which the program is to be loaded. An advantage is that we now have more flexibility than the static compile time option and can use any free (contiguous) memory. Disadvantages include the additional time required for performing relocation, the fact that this must be done on every load (including swap in), and that we still require contiguous memory.

We can solve this at run time by using *virtual addresses*; i.e. dynamically mapping from "program addresses" into "read addresses". Usually this means that the processor will have either paging or segmentation hardware. Advantages include greater flexibility (e.g. with paging can now use any frame of physical memory for any page of a process's image). Disadvantages are the additional hardware and run-time costs (and complexity).

[1] An answer which refers to position independent code is also fine.