On the IBM360

(Translation of EWD255)

E.W. Dijkstra*

A first series of objections concerns the functional specification of the organisation of channels. This is designed with the intention of the CPU submitting a chain of instructions to the channel for sequential execution. However, the hardware is poorly utilized because:

1. when a channel sends an interrupt to the CPU before an earlier interrupt from this channel has been processed, the previous interrupt is lost;

2. it is impossible to add reliably an instruction to a chain that is being executed by a channel: should the CPU attempt this, then there are circumstances under which it is not possible to determine reliably whether or not the channel has just completed execution or still has to do so. (This impossibility is a consequence of the fact that the length of the chain is indicated by marking its last element.)

3. if the execution of an instruction fails, then this is duly reported, but the channel continues after this failure with the next instruction in the chain as if no failure had occurred.

As a result of these shortcomings the instruction chain cannot be used in the intended manner and the operating system is faced with moral urgency situations[1] that would otherwise be avoidable. (To those not familiar with operating systems this probably seems to be a criticism of detail. I, on the contrary, feel that this defect is an alarming indication with respect to the competence of the design team: this is not a question of taste or style, things have been designed with sensible intent while simple reasoning demonstrates that the facilities offered are inadequate.)

A second series of objections consists of examples where the designers have left to the software the management of the machine's components instead of allowing the software to specify the process at a more abstract level, delegating the management of specific components to the system/machine. Examples of this are the following.

1. Peripherals can only be controlled by coupling them to a channel and by subsequently issuing instructions to this channel. And this while the channel has no logical significance; the peripheral does have one.

2. The arithmetic unit has a large number of registers of which it is indicated explicitly in the program text which ones are to be used. This has unpleasant consequences:

   (a) that compilers are confronted with the problem of "register allocation", which leads to an optimization process that is costly in terms of compilation time

   (b) that the kind of status change that occurs in subroutine calls and in multiprogramming becomes unavoidably costly because of obligations to save or restore reg-

*Translated by M.H. van Emden.

[1] Translator's note: I cannot make sense of this. It seems to me that "moral urgency situations" is the correct translation of the original's "morele haast-situaties". Is he talking about race conditions?

ister contents (whether these turn out to be necessary or not!). Here the design optimizes at the microscopic level, which must be paid for many times over at the macroscopic level:

re (2a) the fact that compilation is so time-consuming is responsible for the demand for "independently pre-compiled program components", the combination of which has created the need for a so-called linkage editor;

re (2b) in case time becomes an issue, subroutine calls have to be replaced by an adapted copy of the subroutine's text, which leads to extremely lengthy programs, so long, that their assembling becomes "a major processing task" (Asher Opler, IFIP 1965). IBM will, in that case, be happy to supply the required memory!

3. Instead of the program addressing *information*, it has to address *memory*, either in core or on disks, with the consequence that every program is responsible for its private organisation of "overlays" and transfers between slow and fast memory[2]. This has rather disastrous consequences:

   (a) Standard programs have to exist in different versions, depending on their core requirements. The desire to use programs of others (APT for the 360, for example) can force one to install at least that amount of core. Moreover: enlarging core does not seamlessly lead to increased efficiency, at least not without some reprogramming.

   (b) The only way in which distinct programs can use common subroutines that have only a single occurrence in core is to have these subroutines permanently present. Because one has to be economic with this, the need arises for "system generation", adapting the system to the problem mix, with all its consequent misery: it is a time-consuming process and, moreover, one does not want to do it, as the problem mix can change!

   (c) The fact that programs in execution occupy an immovable contiguous part of core memory makes scheduling harder to an unnecessary and disproportionate content; the resulting system remains inflexible in use.

[2]Emphasis added by translator, who finds the original obscure and gives it a sense in this way. The original reads: " In plaats van dat het programma informatie adresseert, moet het geheugen adresseren, hetzij in kernen, hetzij disks, tengevolge waarvan elk programma individueel belast is met privé organisatie van "overlay"'s en transporten tussen langzaam en snel geheugen. "

Summarizing: details of embedding, which could have been abstracted from by slightly more refined hardware, now require their explicit representation in programs. On the one hand this unnecessary explicitness makes program structure more difficult; on the other hand it is the case that this premature fixation detracts from the suppleness with which the installation can be used.

The unbelievable thing is that this machine is now being advertised with the "wonderful operating system" that relieves the programmer of so many tasks! This glosses over the facts that:

1. a large part of these tasks are necessitated by the hardware and would have been easier or non-existent in a better design;

2. this operating system implies an alarming overhead (but IBM is happy to supply a faster model from the family);

3. the tasks have not been obviated, but have been transferred to the computing centre's management, which has to dimension and manage the machine

4. the creation of this operating system turned out to be a programming task beyond the capabilities of the manufacturer;

5. that the system has become such a baroque monstrosity that no user can venture to adapt it.

Finally, just as the 709-series always struck me as a desparate attempt to use tapes as back-up storage, the 360-series strikes me as disk units with accompanying electronics. It would not surprise me if the somewhat disappointing performance of the faster models comes down to the fact that the faster model is waiting, say, four times faster for a disk arm to move. How they are going extract themselves from this vicious circle is not clear to me: to address this discrepancy by multiprogramming is not attractive from the point of view of the CPU; the advent of “one-head-per-track-disks” might well undercut the raison-d’être for the current operating system.