A review of the IBM 1620 Data Processing System.

It is a good custom that scientific books and articles are reviewed and that no publisher ever thinks about starting a lawsuit or any other measures of vengeance against the author of a very unfavourable review of one of his publications.

With this in mind it is somewhat curious that it is not customary to review digital computers. Reviews of these scientific instruments are in some respects much more important: it is a pity if you have bought the wrong book, but it is much, much worse if you have bought the wrong computer.

The idea that computer reviews ought to be written and published is with me for some time already. The final impuls to write such a review has been the discovery of two "Letters to the Editor" of the Communications of the A.C.M., both expressing praise and appreciation for the same computer, viz. the IBM 1620. They are the letter from Fred Gruenberger, C.A.C.M.,Vol 5, April 1962, pg 221 ("Besides being extremely ingenious....") and the letter from Daniel Herrick and Neal Butler, C.A.C.M., Vol 5, September 1962, pg 469 ("We wish to join Fred Gruenberger of the RAND Corporation in praising the variable word length IBM 1620....."). It is my considered opinion, however, that this machine embodies some very fundamental mistakes and certainly after the publication of the two letters mentioned above I regard it as my duty not to remain silent any longer. Manufacturers should be warned for these mistakes in order not to be tempted to incorporate them in their future designs, also machine users should be warned for these mistakes in order to help them in not chosing the wrong machine and in order to create a climate where machines will be judged more by their fundamental properties.

After the two praising letters I can be short about its virtues: I should like to add that as soon as the troubles of installation are over, it is an extremely reliable machine.

Before going on I should like to explain why I may have objections to "superfluous features". Suppose that a machine contains a certain feature and that I can show, for instance, that it is impossible to use it intelligently or that its use gives rise to undesirable programming conventions; suppose furthermore that the defender of the design agrees to my objections but defends the feature by pointing out that, if I do not like the feature, I do not need to use it, implying that no harm can be done by something "extra". In that stage of the

discussion I shall stress that the design would have been better without the feature under discussion. If it is impossible to use it intelligently every effort to do so is spoilt and the programmer would have been better of without it. If its use gives rise to undesirable programming conventions, also in that case the programmer had better ignore the feature completely.

The IBM 1620 contains a beautiful example of such a harmful "superfluous feature". Besides a general mechanism for calling subroutines, it has a special instruction ("Branch and Transmit") to call in subroutines, an instruction by which control is transferred to the subroutine after the return address has been saved; for the benefit of the end of such a subroutine the order code contains a complmentary instruction ("Branch Back") which takes care of the return jump. So far, so good, but the blunder is that the return address is saved in a special return address register, the contents of which are accessible only via the instruction "Branch Back" and in no other way! Before I got acquainted with the IBM 1620 I thought that in the mean time everybody knew that the most essential property of a general purpose information processing machine is that it allows you in principle to process any piece of information in whatever way you like. I was most astonished (and appaled) when I discovered that the design of the IBM 1620 Data Processing System has <u>not</u> remained faithful to this fundamental requirement: it is just impossible to write a program only inspecting the value of the return address register. As a result, the incorporation of this feature in a program containing many nested subroutine calls requires that the programmer decides, once and for all, on which level the feature will be used. The special feature has two so-called advantages: the calling sequence requires less memory space and the execution time of the call is reduced. Alas, the subroutine which is called in most frequently is, in general, not identical with the subroutine which is called in from the maximum number of different places in the program and the poor programmer who tries to decide intelligently on which level he shall make use of the special feature "Branch and Transmit" is faced with a fairly undecidable peoblem. Needless to say that, once the use of the feature has been incorporated in the program, program modifications become unnecessarily tricky or even impossible without rewriting major parts of the program.

There are more reasons why the machine is not worthy of the qualification "general purpose machine", in particular as far as its facilities for paper tape input and output are concerned. The tape reader has been equipped with a number of special properties which make it completely useless as soon as one wants to

process an arbitrarily punched tape, viz. special properties such as:

1) a built in parity check
2) automatic skipping of a specific punching (in IBM terminology identified by "TAPE FEED")
3) continuing to read until a specific punching is encountered (in IBM terminology identified by "END OF LINE").

The tape punch has similar pecularities. As a result the machine cannot be used to process tapes made by some automatic measuring apparatus or to produce tapes for some tape controled machinery, unless the tape conventions of these pieces of equipment do not violate the restrictions set by the IBM 1620.

Tape reading has another curious property. One can start tape reading and successive characters from the tape are stored on consecutive (pairs of) digit locations in the memory, starting at a known location. The reading process stops as soon as the character "END OF LINE" is met on the tape. This terminal character, however, is not stored in the memory, another character, named "Record Mark" is stored instead. After completion of the tape read operation the number of characters read (or the address of the location of the terminal character) is lost and if the machine wants to detect how many characters have been read from the tape, it must scan the memory. But now a curious problem arises: the terminal Record Mark which has been stored is indistinguishable from previous Record Marks which might have been read from the tape, and therefore the machine is faced with a problem that shows a striking resemblance to the protoype of an improper algorithm: a man asking the way and getting the answer "You go straight on and turn to the right just before the last steel bridge.". As a result one comes to the shocking conclusion that it is impossible to use the IBM 1620 Data Processing System for one of the most trivial jobs: the reproduction of a punched paper tape, impossible even when it is known that the characters on the tape all belong to the restricted set of the IBM 1620.

On the remark in the Referance Manual "The IBM 1620 is a variable field length computer in the complete sense of the term." I should like to give two comments. I agree that variable field length (plus a non-negligible additional burden on the programmer) in principle admits a more economic storage utilization. But the design of the IBM 1620 has turned out in such a way that it is very doubtful whether this gain can be attained. The store is divided into locations of five bits, each location requiring six magnetic cores, because every location has its own parity bit. To store numerical data one needs one location, i.e. 6 cores, for

every decimal digit, which compares rather unfavourably with the 3.4 cores needed in the case of binary words. It seems to me very doubtful whether cunning exploitation of the variable length feature will be able to compensate this initial loss, in particular because the gain must come from the way in which the data are stored: the program is stored in fixed length two address instructions each requiring the considerable amount of 72 cores! The full waste of core storage is certainly attained in the processing of FORTRAN programs, where fixed and floating point variables have their fixed length of 4 and 10 locations respectively. My second remark on the variable field length pertains to the way chosen to specify the field length, a way which I regard the more objectionable, the more I think about it. One of the bits of each location has the special function of "a flag bit"; a variable field length operation starts by an addressed location and processes consecutive locations from there onwards until a location has been processed with its flag bit = 1 . An alternative solution (as e.g. in the Philco 1000) would have been to state separately and in advance, somewhere else, the length of the field in question. First of all, the method of the flag bit tends to be rather uneconomic as far as number of cores is concerned, because

1) the field length is effectively given in a "unary number system"

2) the field length is stored all over again with every field, which is apparently a waste as soon as a great number of fields of known equal length are to be stored.

The next objection to the flag bit, which is of a more fundamental nature, also applies to the special function of the Record Mark, a character which -one level higher- may be used to end the operation "Transmit Record". Information stored in the memory of a machine shows a certain hierarchical structure: on the one hand we have -on a certain level- "the information proper", on the other hand we have "the addressing information" specifying place and/or extent of the information proper. The flagbits and the Record Marks play with respect to the fields and records the role of "addressing information" In the IBM 1620 (as in some other machines) the correspondence between the information proper and some of the addressing information is established by a fixed convention in terms of relative position. This would be all right if the information proper were only regarded "from one single point of view", but a basic property of a general information processing process is that information can and will be regarded from many different points of view. If addressing information has to be stored in a position uniquely fixed relative to the information proper, such a change of "point of view" will, in general, imply extensive, clumsy manipulations on the addressing information, in this case on the flag bits or on the Record Marks. If somebody tries to make a program shifting the arbitrary contents of a given sequence of memory locations

to another place in the memory, he will realize the full impact of my objection.

The IBM 1620 is not the only machine on the market, where (part of) the addressing information and the corresponding information proper are positionally dependent. Because such machines can scarcely be used whenever the way in which the information proper is to be addressed changes frequently, their field of application is virtually restricted to rather straightforward jobs; they are, for instance, no longer qualified to take the place of a university computer. I always wonder whether the designers of such machines have been aware of the restrictive consequences of the technique in question; if so, it is hard to respect their conscious decision to stick to it, if not, are they the people that should have been designing machines? I always wonder.....

From the letter from F.Gruenberger I quoted "Besides being extremely ingenious...". In contrast to this opinion I should like to state that, in more than one respect, I regard the design as rather crude. I shall give two examples.

Addition of two numbers is performed serially from right to left. If a short number (source) is added to a longer one (destination) the addition needs only to be continued to the left of the most significant digit of the sourde so long as the carry is unequal to zero. In the IBM 1620, however, the process is always continued to the bitter end of the destination.

The instruction for the punching of a piece of paper tape always ends by punching a character "END OF LINE". To punch a sequence of characters unequal to "END OF LINE" one is therefore forced first to build up the sequence on consecutive locations in the memory and then to punch out this sequence by means of a single punch instruction. But when this punching takes place (at the conservative speed of 7 char./sec!) the machine itself stands idle.

With this last example in mind one must come to the conclusion that the software provided by the manufacturer is just ridiculous. Even the processing of SPS (standing for "Symbolic Programming System", although it is hardly more than just machine code with mnemonic letter combinations for the different instructions) implies the punching out of the "translated" program. One of the SPS-processors punches the translated program in portions of a fixed length of 80 charcters, although the last 8 are never used (and sometimes not even one or more preceding multiples of 12).

Concluding remarks.

As the reader will understand, my recent study of the IBM 1620 has been a shocking experience: I knew that it was a rather small machine but I had never suspected that it would embody so many basic blunders. Personally, I cannot undergo such an experience without asking myself what its morals are.

One of the facts we have to face is that this machine, despite of its poor qualities, has been bought or rented. Either the customer is incompetent to judge what he is buying, or the contracts are signed by the wrong persons; in both cases the conclusion is that the fact, that other people have chosen a particular machine, is no guarantee whatsoever as far as its quality is concerned. Well, this is hardly surprising. (For this fact one could try the explanation that the machine, although limited gives "value for its money". But this is belied by the machine itself, for even a modest expert can see, that it also gives "nuisance for tis money": for instance, without the additional "END OF LINE" character at the end of the punch instruction the machine would have been ceaper and more sound at the same time.)

The next fact that we have to face is that this machine, despite of its poor qualities, has been produced, in this case even by a [illegible] big firm with a long and considerable experience. The straightforward conclusion is, that nor the size nor the experience is a guarantee as far as the quality of the product is concerned. Well, we can think of various explanations for this apparent inconsistency, but the most obvious explanation predicts still more blunders in the more ambitious and more complicated products of the manufacturer in question.

Januari 1963

E.W.Dijkstra
Technological University
Department of Mathematics
Postbox 513
E I N D H O V E N
The Netherlands.

Literature: IBM Reference Manual 1620 Data Processing System, printed in the Netherlands A 26-4500-1.