The Multiprogramming System for the EL X8 THE.

In the following I shall describe our proposed multiprogramming system. While starting to do so I am aware that I shall be content with a system satisfying our needs as we see them; I am also aware of the fact that the presence of the drum and the way in which we intend to effectuate the so-called "Segment Control" have had a major influence on our approach. For a detailed description of Segment Control we refer to EWD113.

Sequential processes, embedded in the machine.

Sequential processes, to be performed by the central computer under program control have been called "abstract Machines". They fall into two groups, the programmable machines, called for short the PM's and the constant machines (origibally "de stamgasten"), called the CM's for short.

Each PM represents the possibility to execute a user's program; the number of PM's will be fixed, say 4.

Each CM describes a process to be performed synchronized with the interrupt signals sent by one of the pieces of peripheral equipment. We shall have a CM

1) for each tape reader
2) for each tape punch
3) for the line printer
4) for the plotter
5) for the drum (the so-called "Segment Controller")
6) for the console teleprinter

(We regard the console teleprinter as a single piece of equipment: when the keyboard is used the printer is used by the X8 in complete synchronisme, and therefore the teleprinter gives rise to only one CM.)

The 12 abstract machines, thus embedded in the installation have to cooperate with each other. Originally we thought that one needed a 13th program, say "a coordinator" to regulate this cooperation. This now strikes me as organizing a society out of 12 unwilling individuals by appointing a policeman. Our present solution is more aptly described by making 12 mutually well-behaved individuals. There is still some coordinator left (viz. the instructions performed between transition from one abstract machine to another) but this is so meagre, that I doubt, whether any attention will be paid to it in this report at all.

Information streams.

The different abstract machines can communicate with each other in the sense that one can describe its "status" and that another can inspect it. This section, however, deals with bulk information transport between an input CM and a PM or between a PM and an output CM. In other words with the transportation of information which must be regarded as subject matter of the user's program. The following discussion does not apply to the drum, nor to the teleprinter. Not to the drum, because the activities of the segment controler take place behind the scenes, segment control is in fact, integrated with the addressing facilities available to each abstract machine; not to the teleprinter because here we shall only allow for very limited buffering and it is exactly the task of the information streams to cater for that.

Each PM has the same fixed set of information streams to its private disposal: the PM's are identical and there is no question of "stream sharing" between the different PM's.This set will comprise two tape reader streams, two tape punch streams, one plotter stream and one line printer stream. The local terminology with which each PM can identify its various information streams is the same for each PM.

When an input CM is engaged in the production of an information stream, it must know which information stream it is building up. We may think of a local variable "current information stream" which can take 8 possible values (8 tape reader information streams being present in the installation).

Similarly a tape punch CM must know, when engaged, which of the 8 tape punch information streams it is consuming, the plotter CM and the line printer CM must know which of the four information streams, destined for them, they are processing.

An information stream consists of

1) a fixed size "handle"
2) a number of chained segments (may be zero)

The handle describes the state of the information stream, whether a CM is engaged with it, how many segments there are in the chain, if so, links to the first and last segment in the chain etc.

Producing and consuming the contents of an information stream are strictly sequential operations, an information stream is really a first-in-first-out memory. As a result the output of two programs, via the same (local) output information stream will appear the one after the other when the programs have been processed by the same PM. (A program may leave the PM as soon as it has finished its task, that is chaining its last output to the appropriate stream. The PM becomes immediately available for the next program, although its output information streams may still contain many segments!)

Chaining and unchaining of transput information.

Information is chained onto and unchained from an information stream in fixed size portions, the size being characteristic for the nature of the stream concerned. For the line printer it will be the picture of a form (from fold to fold) for the other streams it will be a segment.

During production such a portion will be treated as an array local to the producer, at "chaining onto the stream" its SV(or SVs) will be transported. Also during consumption such a portion will be treated as a local array of the consumer; before actual consumption can start, it must be "unchained from the information stream", an action which again requires the transportation of SV-values. The advantages of this are

1) no separate addressing mechanism has to be provided in order to address the individual words of such portions
2) chaining and unchaining do not involve any operation on the bulk information itself, but on the SV's only.(Our latest version of Segment Control enables us to move SV-values under all circumstances; in this case however (alas?) it can be shown that the operation of chaining and unchaining will in time always be disjunct with Segment Control activity concerning this segment!)

There is, however, a small difference between chaining and unchaining by a CM or by a PM. Whether a CM is active or not, there will always be set aside a segment (or, say, 4 segments if it is the line printer CM) to store its current portion. This implies a permanent reservation "to keep the transput going" of about 9 segments. For the PM's however, these portions are undistinguishable from any other local array; when used, they count among the total number of non-empty segments occupied.

The concept "document" and equipment sharing.

A document is the amount of information in a single stream (to be) processed by a CM continuously, i.e. without the CM in the mean time switching over to serve another information stream.

For the tape reader, a document is a single physical tape. The document size is given by the input itself and need not concern the programmer.

For the plotter a document is a single "picture". As each PM is only equipped with one plotter output information stream, a user's program has only the facility of specifying one picture at the time. We shall not exploit the possibility that the plotter may leave an unfinished picture, turn to the next picture and finish the earlier one later. It is the duty of the programmer to indicate dynamically begin and end of plotter documents.

For the punch a document is a piece of tape to be punched without "end of paper" interruptions. Also for the paper tape punch the programmer must indicate dynamically begin and end of punch documents. As far as plotters and punches will be treated on the same footing we shall refer to them as "plunchee".

For the line printer a document is a whole number of consecutive forms. The programmer has no control over this number, the line printer prints forms from the different information streams "as it sees fit".

With regard to the deadly embrace, we propose the following, mixed attitude.

From the line printer no difficulties can arise, as the line printer can -in case of emergency- always restrict itself to documents of single forms.

For the tape readers, we assume, that the facilities for input buffering are such, that when a tape reader is wanted, we can always wait, until a tape reader is free again.

For the plunches we propose to apply the banker's algorithm. The moment of the starting of the loan, is not when the program specifies "begin of document" but when it is decided, that the plunch will start with the output of an incomplete document. The initial program description must specify, which plunch streams it is going to use.

The punch CM's keep track of the amount of paper, used from the present reel. When they start a finished document, they know beforehand, whether the amount of paper tape still on the reel is sufficient or not, if not, they will ask the operator to insert a fresh reel. If they have to start with an incomplete document, they will always ask for a fresh reel.

The activity of a CM has two distinct stages, "document selection" and "document processing". In document selection it has to decide what to do next, in document processing it has to decide at what speed.

The easiest part is document processing by an output CM: "as fast as possible". Document selection is a more refined process. The plunches will try not to initiate the processing of an unfinished document, but may be forced to do so, when total output buffering starts to overgrow a certain limit. If an output CM has the choice between various finished documents, the one from the same information stream as the last document processed has a certain preference, but monopolization by a single, very productive PM will be avoided.

In the case of input CM's, both document selection and document processing are complicated. The document selection is difficult on acount of the feature of the open mouth, tape reader request by a PM for the reading of a non-standard tape, a request, which may be refused by the operator, and "urgent request for a reader" by the operator. Document processing is also non-trivial, because the simple rule "as fast as possible" is not applicable. The idea is, that tape processing might occur under two different circumstances: either interest has been shown in this tape reader or not. In the latter case it will regulate its speed so as to keep the information stream reasonably filled (target goal: one segment in the chain). If, however, another user is waiting for it to become free, then it will read as fast as possible.

The logic for tape reader document selection has been worked out to great detail; for the purpose of illustration we list the console messages, associated with it. (The numbering of the messages should not confuse the reader, they have grown historically)

M1. Machine opening.
It requests the operator to insert a tape (identified in the message) in a tape reader (also identified in the message). Console remains reserved for operator answer.

M2. Operator answer to M1.
Tape requested has been inserted. Console becomes available, conversation closed.

M3. Operator answer to M1.
Tape unavailable now. Console becomes available again, as well as the tape reader specified. Conversation remains open, requiring operator reopening M4.

M4. Operator reopening.
Requested, refused tape available now. This message must identify the PM. The console becomes available again, end of conversation. At some later stage, however, the machine will give again a M1 message.

M5. Operator opening.
Request opportunity to insert standard tape. Console becomes available, conversation remains open, requiring machine reopening M8.

M8. Machine reopening.
An identified free tape reader is at the disposal of requested standard tape insertion. (The operator has no means of disavowing his earlier request. If he comes now to the conclusion, that he does not want to insert a normal standard tape, he should insert a little piece of blank tape, "the empty standard tape".)

M9. Operator opening.
Specified tape reader to be put out of circulation. Console becomes available, conversation requires reopening M12.

M12. Machine reopening.
Disconnection effected. Console becomes available again, conversation requires reopening M13.

M13. Operator reopening.
Reconnection disconnected tape reader. Console available, conversation closed.

M14. Machine opening.
Request to remove the tape from an identified reader. Console remains available for operator answer.

M15. Operator answer to M14
Requested removal has been done. Conversation closed.

M16. Operator answer to M14.
Skip until end of tape. Conversation closed.

(Remark: All messages printed by the operator are tested for applicability. If an operator (re)opening is not applicable, the console becomes available after rejection, if it was an immediate answer, the console remains reserved for the acceptable version of the answer.)

Memory sharing.

In this section the segments that are permanently present (library etc.) are left out of the discussion. We restrict ourselves to the managing of the remaining part of the drum, say "tot" segments.

In this space we have to accomodate three kinds of segments, their total numbers being indicated by p, i and o:
p = the total number of occupied segments that are part of the PM's
i = the total number of segments in input information streams
o = the total number of segments in output information streams.

We have decided to impose a maximum bound on "i + o". The presence of the segments in the information streams implies the presence of SV's and linking information. It would upset the total organization rather drastically if we would have to introduce segments the SV's of which were not permanently stored in core. Furthermore the life times of these SV's considered as a whole are rather chaotic. For that reason we have decided to set aside a part of core store (the transput area), in which the stream segment SV's and the linking information can be accommodated. But this results in the upper bound mentioned. We think about 256 segments, i.e. a quarter of the drum.

Furthermore we wish to set aside, say, a quarter of transput area for output buffering, i.e. 64 segments. It does not seem necessary to make a fixed reservation in transput area for input only: if the shoe begins to pinch it will become available provided that the output is indeed active. We intend to see to it, that output is indeed active, as soon as output buffering exceeds the reservation.

To get an idea of the importance of sharing, it is restricted to three quarters of transput area, available for either o-, i-, or p-segments. Say 20 percent of available store.

The vital inequalities to be respected are

$p + i + o \leq tot$
$i + o \leq tot - resp$ (=size transput area)
$p + i \leq tot - reso$
$i \leq tot - (resp + reso)$.

We have not worked out yet a complete strategy, we have what I should like to call "a pattern of a strategy", i.e. we have a picture of the relevant occurences in the abstract machines, and with these points their possible actions and influences upon each other. We shall now give a list of these points.

They are primarily:

SH1 p-increase in a PM (first reference to an empty segment)
SH2 p-decrease in a PM (killing of non-empty segments at block exit)
SH3 i → p transition in a PM (unchaining from input stream)
SH4 p → o transition in a PM (chaining onto output stream)
SH5 i-increase in an input CM (chaining onto input stream)
SH6 o-decrease in an output CM (unchaining from output stream)
SH7 document selection in output CM

In general, at these points the abstract machine investigates the situation and one of three possible reactions may result
1) it may decide, that a disaster has happened (e.g. store explosion, to be avoided)
2) it may decide to do its action
3) it may decide to postpone its action and to wait. In the last case "the situation" tells clearly, that it is waiting. It then delegates to the remaining abstract machines to discover the situation under which it can go on again and to see to it, that this indeed happens. Therefore, these points, in general, include the inspection whether one or more waiting processes can go on again; we call this "V-inspection" (inspection whether a V-operation is appropriate).

As SH1 and SH2 are the most frequent occurrences, our guiding principle has been to burden them as little as possible. In particular:

SH1 will not be burdened by the V-inspection, whether on account of the now tight store situation, document selection SH7 becomes most desirable. We intend to do so by guaranteeing that as soon as o > reso, there will certainly be active output.

SH2 will not be burdened by V-inspection, whether on account of the now less tight store situation, completion of SH1 or SH5 can be done. This is accomplished by signalling "disaster" when in SH1 or SH5 "p +i" is in danger of exceeding its upper bound.

We have first to see, how we can guarantee active output as soon as o > reso. The only way in which we are forced to violate this rule is if o > reso without being able to perform document selection , i.e. when the appropriate CM's are not available (tied up with other unfinished documents) or not assignable on account of the bankers algorithm. That is, the banker must avoid unfinished unselected documents with a total size exceeding reso. That is: as soon as this limit is reached, unfinished document selection has to be done, if the banker allows it, otherwise the PM has to be held up, until the banker allows it or the total amount of unselected unfinished documents is decreased otherwise (by selection, say)

From this it follows, that we have to mention a next relevant occurence:

SH8 end of pluncher document, in a PM.

After these preparations we may hope to be in a position to give an outline of the necessary actions at these relevant points.

At SH1 it is inspected whether the p-increase is permissible . If so, the PM continues its activity, if not the reason is
a) the upper bound of "p +i" →disaster
b) the upper bound of "p + i + o" → wait.
(Case b implies o > reso, this implies output activity and in due time, the o-decrease will be in a position to remove the barrier.)

At SH2 we only decrease p by the amount of non-empty segments killed.

At SH3 -which may be preceded by a delay, because the next input portion was not present yet- it is inspected
a) whether now a SH5 should be completed (if there is still a CM active in producing portions for this stream, that one is a succesful candidate)
b) whether now a PM should complete its SH4 (which might be held up by the upper bound for "i + o"). This mechanism makes it possible that space used for input becomes available for output.

At SH4 distinction is made between "forms and selected documents" on the one hand and "unselected documents" on the other hand. In the first case, the "i + o" bound may represent an obstacle; if so, the PM will wait, otherwise it will chain. When, as a result of chaining, "o > reso" then it will inspect, which output CM's shall complete their SH7. If it is an unselected document, the PM inspects, whether the total number of unselected segments would exceed reso. If not, it will attach, if so, it will call for the banker. Either this results in document selection or not. The PM will either be allowed to chain on or it has to wait.

At SH5 the input CM will inspect whether it shall chain on its next portion to the stream (i - increase).
It will signal disaster when
a) its stream is empty and
b) chaining on would result in exceeding the upper bound for "i" or "p + i".
It will wait, when
a) its stream is empty
b) chaining on would result in exceeding the upper bound for "i + o" or "p + i + o".

When its stream is not empty, it will wait, when the stream is reasonably filled, it will chain on, when it is not reasonably filled. (At this stage we do not propose to give a further description of the criterion "reasonably filled"; it shall be dependent on how far the upper bounds are still away and whether this CM has been selected for the next paper tape, so that there is a point in trying to make it free.)

At SH6 -i.e. o-decrease by the unchaining from the output stream, an extensive inspection is performed, whether an SH1, an SH4 or an SH5 can now be completed.

Finally SH8 (end of pluncher document) will inspect, whether this document was already selected; if so, the banker comes in action and SH4-waits may now be finished. Above that, programs will be able to signal "end of output": this has influence on the document selection of the CM concerned, for then there is no point in waiting for more. (It is very nice if the line printer prints as a rule at least 5 consecutive forms from the same program, but this is a little bit difficult if the program produces only three forms.)

Remark: The scheme contains, right from the start, the danger, that a PM, privileged by the banker, fails to get a free tape reader. The description given above does not include the detection of this desaster.

Next remark. The count p of "occupied segments" does not include the stack segments, as they are never dumped on the drum. The reason that we have chosen the number of "dumpable" segments as the relevant unit in which to express the storage demands of a PM is that now we can guarantee that, whenever Segment Control needs a free drum page, there will certainly be one available. And if Segment Control would need to wait for free drum pages, we would be faced with the logical translation of "Who was first, the hen or the egg?".