STRETCH Computer Operation

In this paper a "job" is intended to be a single application or problem or task given to the computer as a unified group of instructions. Thus, payroll, matrix inversion, program assembly, and inquiry processing are jobs.

TYPES OF JOBS

1. Jobs requiring all of main memory.
2. Jobs requiring all of main memory only if optimized.
3. Jobs not requiring all of main memory even if optimized.

OPERATION ALTERNATIVES

1. Run only one job at a time.
2. Run more than one job at a time.

In this paper alternative 2 will be called multiprogramming.

Alternative 1 is the only choice for Type 1 jobs. It may be a poor choice for those Type 3 jobs that are of short duration.

Such short jobs may well be in preponderance at a commercial* application STRETCH installation, especially during its initial several months--or years (after which more on-line applications are anticipated.) For example, jobs converted from current 702 and 705 programs may well only require two or three minutes of STRETCH time.

This example implies serious operator problems in physically tending input-output units, if high speed units are used. If low speed units are used, then the main frame is seriously delayed--perhaps doing only two minutes of computing during an hour.

Multiprogramming is an answer to these objections.

AUTOMATIC RESTART using check points may be a valuable technique for correcting machine and operator errors, especially those unanticipated by the programmer. To make restart automatic, input-output devices calling for manual adjustment (such as card readers and printers) cannot be used. This implies preliminary card-to-tape operations as well as tape-to-print.

These are also a source of small jobs which can be multiprogrammed, if done by use of the Exchange (and memory).

Reasons for also doing some calculating (editing, processing) during card-to-tape operations are listed on the following page.

*As opposed to scientific

1. Editing input data for reasonableness can be done in advance of its use in the main job. This is an advantage because —

   a. Errors are discovered when the cards (and perhaps the source data) is available. (Note that when multiprogramming, the computer is not delayed by interruption of a card-to-tape job. It merely advances to another job.)

   b. Interruptions delay only the card reader and the tape drive rather than the several input-output units perpared for the main job.

   c. There is more time to correct errors, because they are observed earlier.

2. This adds good main-frame time to a job that otherwise is completely input-output.

3. Program writing, program testing, and execution efficiency is increased by effecting axis translation and code conversion here rather than during the main job. (A similar statement applies to tape-to-print.) This is not a denouncement of the use of special devices to accomplish these functions. Main frame utilization in some installations may not be sufficient to justify these devices.

   Input-output devices and their control units require less hardware if they need not be used for independent (peripheral) operations.

   They also require less hardware if they do no editing.

   The above paragraphs cite the disir ability of using the Exchange (and also the main frame in certain cases) for operations such as card-to-tape.

CONCLUSION:

Installations, except those having type 1 jobs exclusively, may well want to establish a multiprogramming technique that will permit the completion of type 1 jobs as a special case.

This is probably a necessity in the "inquiry" application.

The following UTILITY PROGRAMS seem to be required to implement a multiprogramming technique:

1. An automatic program to assemble and generate translatable instructions from symbolic and pseudo codes.

2. A translation program to assign actual addresses to these instructions.

3. A monitor program to divert the attention of the main frame from one job to another; i. e., a controlling routine to accomplish multiprogramming without use of multiple instruction counters.

4. A supervision program that will keep a program log in memory on the status of all jobs and input-output devices. This program will choose the next job to be executed and initiate the translation program to locate it in memory. (The choice will depend on adequate memory being available, input-output devices being prepared and proper priority being assigned.) The supervisory program will notify the monitor when the new job is ready for execution.

Observe that the output of the automatic program is a set of instructions having relative addresses. It is in this form that the instructions will be stored on a program tape or disk. The translation program will then convert the relative addresses to actual each time the set of instructions are called into memory for execution.

The monitor program will be a skeletal program on which will be hung the instructions for the jobs being executed in the multiprogram made at any given time. It will have three basic loops---input-output, computing, and break-in. Each of these loops will be controlled by tests made of bits in indicators. These bits will choose the path (among several parallel alternatives) to be followed in each loop--if any.

These indicator bits may be regarded as permissive switches. Setting a sufficient number to 1's will permit the execution of the calculating or processing steps of a program whenever the main frame is available. Setting others will permit input-output instructions to be executed whenever the Exchange and input-output units are ready. The break-in loop will probably do little more than perform some of the above bit-setting functions and permit quick transfer to the input-output loop.

Figure 1 provides a sample flow chart of how the monitor appears when it is controlling four short programs, each with one input and one output unit and one longer program, involving three passes and several input-output units.

In each of these short jobs it is presumed to be necessary to write out the previous result and read in the next record before continuing to process. Thus an M bit is set to 1 (in the break-in loop) when the previous result has been written and an N bit is set when the next record has been read. When the monitor finds both bits are 1's it will start the processing steps. When processing is complete the M and N bits are set to 0's and two K bits are set to 1's. The K bits permit a "read" and a "write" to occur when possible.

At any time a break-in signifying the completion of an input-output operation can occur. This causes a temporary interruption while M and N bits are set to 1's and waiting input-output instructions are initiated. The switch L is provided to assure return to the proper step in the interrupted program.

The longer job is similar except it calls for more indicator bits.

The programmer makes no effort to synchronize his job with any other. In fact he cannot know what other jobs will be running with his. They will probably be different each time. His task is to write instructions for one or more paths in each of the three basic loops in the monitor. He will use symbolic or relative addressing and, in particular, will address input-output units and indicators relatively.

The console operator will have tittle to do other than to watch the program lag, assign priorities, and direct the activities of the input-output operators. Priorities can be assigned manually be appropriate entrys into the program lag and the execution of a few steps in the supervisory program to change the sequence of the computing steps. In figure 1 this means changing the sequence of the M, N tests.

The input-output operators will prepare units for the next job. (This job has probably been chosen by the console operator.) The input-output operator can choose any accessible card reader for card input, any available tape drive for tape input etc,. He then reports the unit number, file number and job number to the program lag by using an inquiry station or the console. (The supervisory program can then observe the completion of the preparation of all input-output units for the job and can cause the translator to locate it in memory as soon as space is available.)

Program testing will have to be accomplished through an interpretive routine designed to prevent an unproven program from contaminating proven ones currently being executed.

JCG/jv

J. C. Gibson

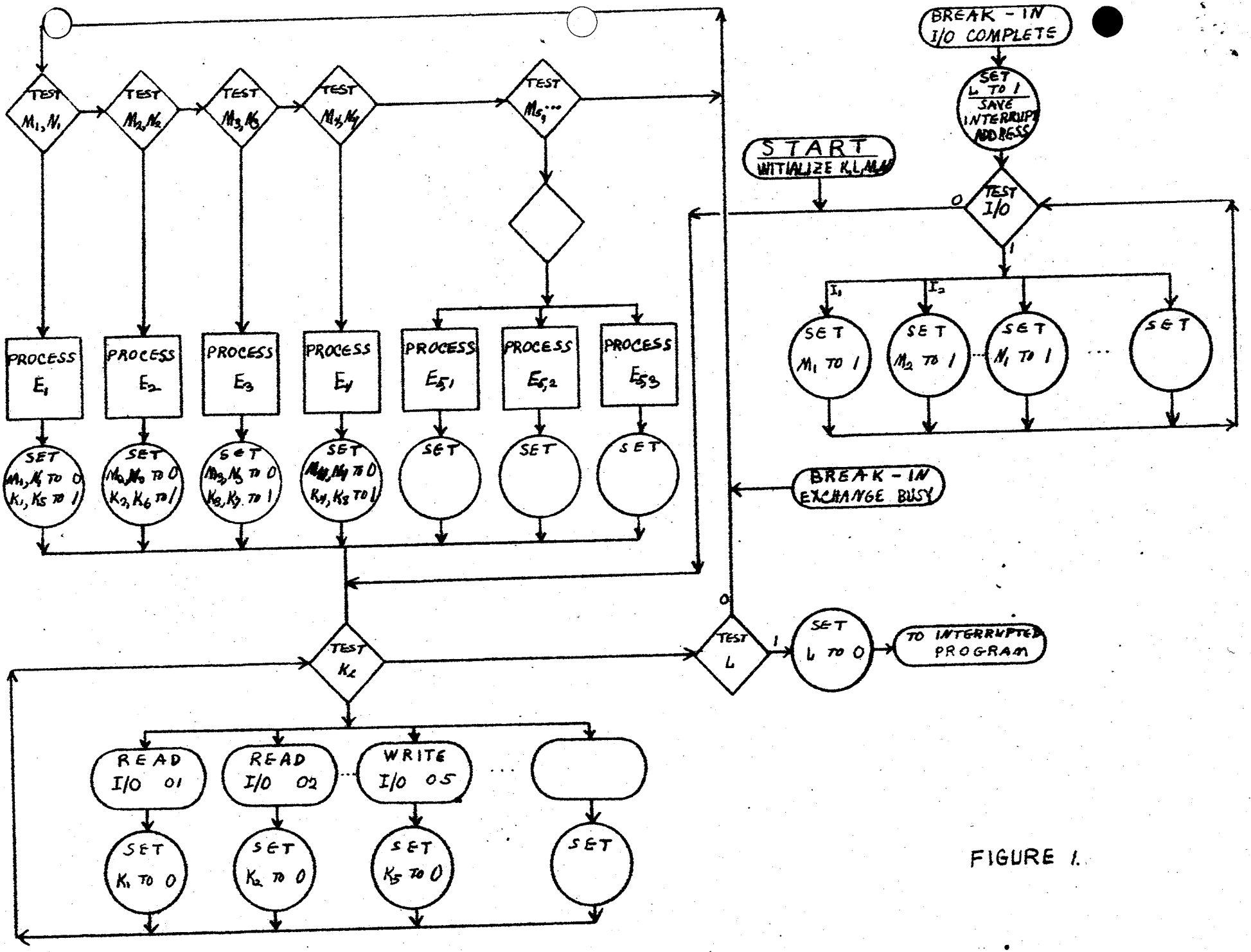
BREAK - IN I/O COMPLETE
SET L TO 1 SAVE INTERRUPT ADDRESS
START
INITIALIZE K,L,M,N
TEST I/O
TEST M1, N1
TEST M2, N2
TEST M3, K3
TEST M4, K4
TEST M5, ...
PROCESS E1
PROCESS E2
PROCESS E3
PROCESS E4
PROCESS E5,1
PROCESS E5,2
PROCESS E5,3
SET M1, K1 TO 0 K1, K5 TO 1
SET M2, M6 TO 0 K2, K6 TO 1
SET M3, K3 TO 0 K3, K7 TO 1
SET M4, M8 TO 0 K4, K8 TO 1
SET
SET
SET
I1
I2
SET M1 TO 1
SET M2 TO 1
SET M1 TO 1
SET
BREAK - IN EXCHANGE BUSY
TEST L
SET L TO 0
TO INTERRUPTED PROGRAM
TEST K2
READ I/O 01
READ I/O 02
WRITE I/O 05
SET K1 TO 0
SET K2 TO 0
SET K5 TO 0
SET
L
M1 N1 M2 K2 M3 K3 M4 M5
K1 K2 K3 K4 K5

FIGURE 1.