A SIMULATION OF THE IBM 370/155 JOB SCHEDULING ALGORITHM

by

ANDREW WASSERMAN

Submitted in Partial Fulfillment

of the Requirements for the

Degree of Bachelor of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 1972

Signature redacted

Signature of Author.....
Department of Electrical Engineering, May 12, 1972

Signature redacted

Certified by.... .....
Thesis Supervisor

Signature redacted

Accepted by.....
Chairman, Departmental Committee on Theses

Acknowledgement

I would like to thank the following people for the help that they gave me in writing this thesis: Professor Malcolm M. Jones, for providing me with the idea and spending so much of his time going over my problems; Mr. Richard Steinberg of the MIT Information Processing Center, for his invaluable criticism and his moral and financial support; Mr. Roy Harrington and Mr. William Hacker, for answering my questions about the SIMPL language; and Miss Alma Hirsch, for volunteering to type the finished thesis, and for putting up with me for the last three months.

Abstract

This thesis is based on a simulation of the job scheduling algorithm used by the IBM 370/155 at the MIT Information Processing Center. The simulation, written in SIMPL (developed by Professors Malcolm M. Jones and Richard C. Thurber), concentrates on the main device scheduler, which is responsible for allocating the seven 9-track tape drives, the two 7-track tape drives, and the ten disk drives to jobs that require these devices. By experimenting with the various parameters that can be changed on the actual system and collecting statistics on the percentage of time that each device spends in the four states (idle, reserved, setup, and executing), it is possible to determine whether or not there are actually too many devices for the system. Although there was not enough money to make as many trials as I would have liked, I have found that the number of devices presently in use is certainly adequate for the system, and that performance would be only slightly downgraded if several of the disks and 9-track tapes were removed.

Table of Contents

List of Figures

A Brief Introduction to the 370 Job Scheduler[1]

The job scheduling algorith under ASP/MVT schedules a job for processing on the IBM System 370/155 in four different steps. First, a job is given an input device. Second, it is given the tapes and/or disks it needs. Third, it is given main storage. Last, it is given an output device.

Input devices

When a job is handed to the operator, it is placed in a tray corresponding to the service requirement index specified by the user: high, standard, or low (jobs with sri emergency are read into the computer immediately). In each tray the jobs are ordered by their submittal time, and the trays are ordered high > standard > low. Allocation of a card reader is according to the following algorithm:

1. Read the jobs in the highest ordered nonempty tray in chronological order into the system.
2. Take the next lowest tray; if jobs have arrived such that a higher tray is nonempty, go to 1.
3. Otherwise read in the lower tray.
4. Go to 2.

---

1. This section is a condensation of GI18-revision2, published by the MIT Information Processing Center, September 30, 1971.

Setup devices

After a job has been read in, it is placed in one of fifteen resident job queues (resqueues), numbered 0 to 14. Jobs with sris emergency, high, standard, and low are filed in queues 12,5,3, and 1 respectively. In addition, some special jobs with low priority go into queue 0. A job remains in resqueue as long as:

1. it needs a device or main storage, and
2. it has not finished executing, and
3. it is not being held.

The main device scheduler (MDS) is responsible for allocating the setup devices. It begins a scan of resqueue when either a new job enters the system or a job finishes executing and frees up one or more setup devices. Starting with the highest numbered queue containing a setup job, it examines each such job sequentially. If all the devices needed by the job are free, they are immediately allocated. The job is moved upward the number of queues specified by INCREMENT[2], where it awaits main storage allocation. If a job cannot be allocated all of its devices, and AGING=YES has been specified, and it was at the top of its queue, it is placed at the bottom of the next higher queue. If there is no higher queue, the job stays where it is.

2. For an explanation of INCREMENT, BARRIER,DEPTH,CHOICE, and AGING, see System 360 and System 370 ASP System V-2, Console Operator's Manual, Revision 8, March, 1971, IBM

There is actually a midpoint between the two extremes of allocating all or none of the required devices. A job can take devices away from another job in the same or a lower queue if the message has not yet been sent to set up the devices. In this manner a job can have some of the devices it needs reserved by MDS. It is also interesting to note that a problem could arise when with priority "high" requests three 7-track tape drives when there are actually only two. All jobs with lower priority could wait indefinitely if they needed any 7-track tapes. Luckily, this situation rarely (if ever) occurs.

MDS ends its scan in four situations:

1. a) When DEPTH=j has been specified, and
   b) j jobs have been set up without finishing execution,

then MDS ends after allocating devices for the jth job.

2. a) When BARRIER=PRTY has been specified, and
   b) a job has been passed over,

then MDS stops after scanning the queue containing the passed-over job.

3. a) When BARRIER=b has been specified, and
   b) MDS started scanning above or in queue b, and
   c) a job above or in list b has been passed over,

then MDS stops after scanning queue b.

4. a) When BARRIER=b has been specified, and
   b) MDS started scanning below queue b,

then MDS stops after scanning queue 0.

Main Storage

The main service routine (MSV) is responsible for allocating main storage. MSV is called in three cases:

1. When a job is read into the system, or
2. when a job has been allocated all it devices, or
3. when a job finished executing.

Like MDS, it scans the fifteen queues. It allocates main storage to a job when:

1. the job has been allocated all of its devices, and
2. there is enough core left to satisfy the region specified, and
3. the CHOICE option is satisfied.

MSV ends the scan when:

1. CHOICE=PRTY has been specified, and a job is passed over, or
2. DEPTH=j has been specified, and j jobs are currently executing, or
3. JSPAN=n has been specified and n jobs have been scanned, or
4. PSPAN=p has been specified, and MSV has completely scanned p queues, or
5. BARRIER=PRTY or BARRIER=b has been specified, and the condition is met.

Output devices

The job segment scheduler (JSS) is responsible for allocating line printers and card punches. It scans the job control table, which contains all jobs in the system ordered by sri and time of arrival within sri. It starts its scan when:

1. a job is read into the system, or
2. a job finishes execution, or
3. a job finishes printing or punching.

JSS allocates a printer or punch when:

1. the job has printed or punched output, and
2. the job has finished execution, and
3. the job is not being held, and
4. there is a printer or punch available.

Once JSS starts a scan, it continues until it reaches the end of JCT.

Assumptions on the Model

In order to simplify the model, yet keep it as close as possible to the real system, the following assumptions have been made:

1. As soon as a job is handed to the operator, it is read into the system. This eliminates the input device scheduler, but is not really a deviation. Under normal conditions, there are enough operators and there is enough time between jobs so that the "waiting trays" are eliminated.
2. The processes of reading in a job and MDS scanning resqueue take no time. Of course, they do take a few seconds, but this will not really affect the model's validity.
3. The time it takes to setup the devices for a job is three minutes. It is almost impossible to measure this time and find a true figure, but three minutes is fairly close.
4. If a job can be executed because the DEPTH=j option has not been passed, then it will be given main storage regardless of its region size. This is the main difference between the model and the actual system. However, this allows me to eliminate MSV and concentrate on MDS, and will probably not appreciably change the statistics on the device usages.
5. Whereas the actual system has fifteen queues numbered from 0, the model has sixteen, numbered from 1. This so that a job with sri "emergency" (queue 13) can be pushed up to queue 16 (normal increment being 3). Numbering from 1 facilitates

the coding.

6. A job will never use more than six devices. There are, of course, infrequent jobs that do require more (such as the Harvard "monster" job), but the chance of such an event occurring is minute enough to be ignored.

Description of the Model

The model is composed of seven procedures, or activities, of which six are external (i.e. written and compiled separately) to the main activity. All variables in the main activity are global (i.e. available to all the activities). However, in order for an activity to reference the data in an external activity, the second must be declared within the first, and the first must set a connector (a pointer to the data base) to the second.[3]

The key data bases of the model are the three arrays corresponding to the devices- tape9(7,3), tape7(2,3) and disk(10,3). The first index is the status of the device, the second index is the queue of the job to which the device is attached, and the third index is the position of the job in the queue. These pieces of information make it quite easy to change the status of the devices and to switch them from one job to another.

All variables that I have used (with the exception of the system variables Time, Agenda, and Current) are lower case. In order to distinguish them in the following pages, they will be underlined whenever referred to. If two words

3. For a complete description of the SIMPL language, see The SIMPL Reference Manual, by Professors Malcolm M. Jones and Richard C. Thurber, October 18, 1971, or The SIMPL Primer, by the same authors, October 4, 1971.

are underlined together like job depth, it means that they are connected by an underscore (job_depth) in the model.

In this section I shall give a description of each activity, along with a flow chart for each. For a listing of the source programs, see Appendix II.

Sched

Sched is the main activity of the simulation. It not only declares and initializes all global variables, but also declares all of the external activities. It receives from the console five of the parameters that can be changed on the actual system (aging, barrier, depth, increment, and job depth), plus the length of the simulation in minutes (simtime).

The body of the activity is quite short. First, it activates initial to initialize the resident job queue (resqueue). Next, it sets the bit start to 0 to indicate that initialization is over. Then it schedules the first sample of resqueue to occur at Time = fifteen minutes.

The statement at the label generate tests to see if it is time to end the simulation. If the system time is greater than or equal to simtime, a final sample is activated; the command "endsim" then ends the simulation.

If simtime has not yet been reached, a new job is generated. First, sched calculates the interarrival time for the job, then delays that length of time. After incrementing the job counter (n), it activates job and passes it nine zeroes. These zeroes are meaningless, inasmuch as they correspond to statistics on the job (region, class, number of devices, etc.) which have not yet been calculated. However, job has been declared to have nine arguments, since initial

does pass nine meaningful numbers; therefore, the zeroes must be included.

After sched has been returned control from job, it activates the main device scheduler (mds). When control again returns to sched, it repeats the cycle by jumping back to generate.

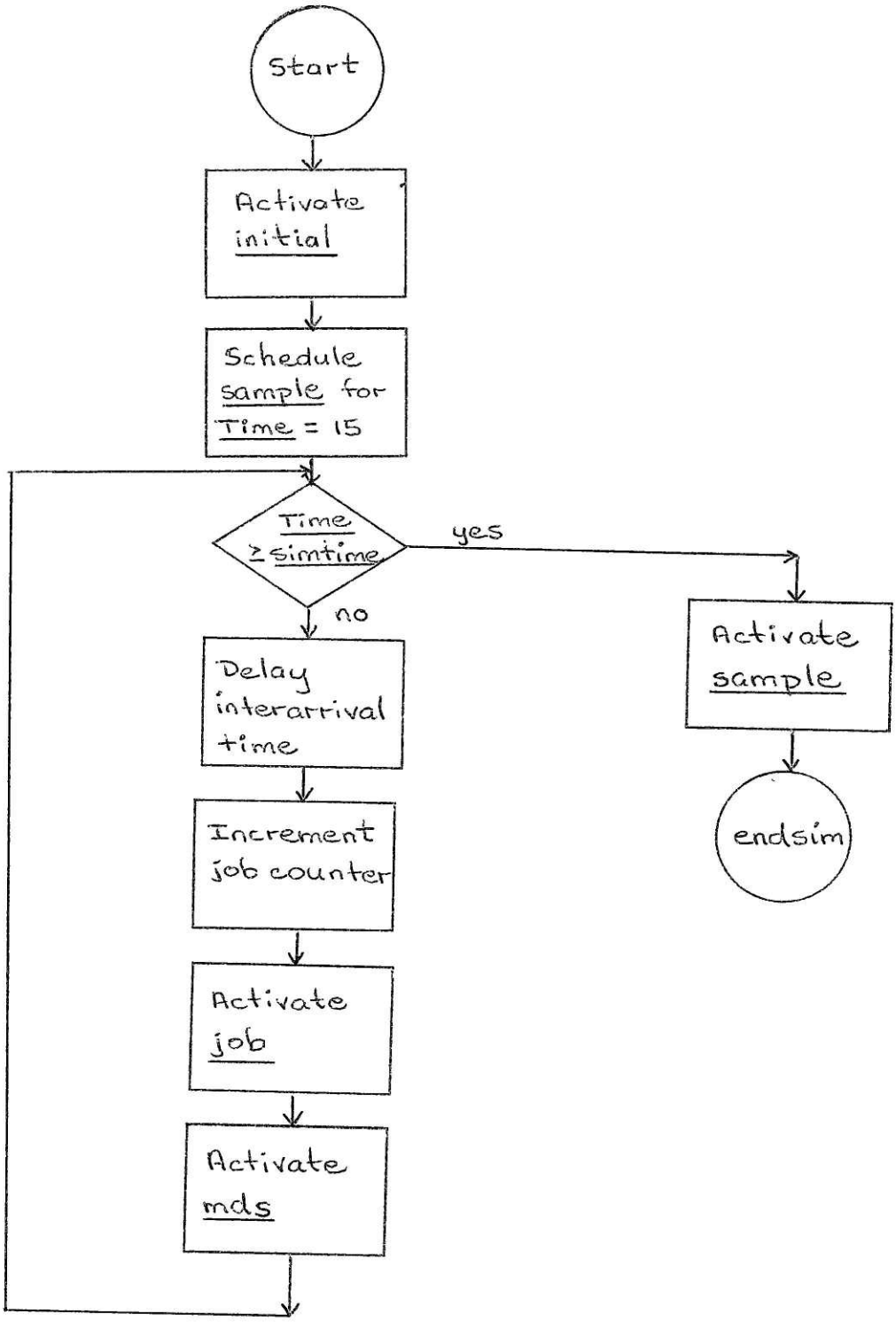


Figure 1 : Flowchart of sched

Initial

Initial is in charge of initializing resqueue. The first thing it does is to request the user to type in the statistics about the job: status, job queue, position in the queue, region, class, run time, total number of devices needed, number of 9-track tapes, 7-track tapes, and disks needed, and number of 9-track tapes, 7-track tapes, and disks still to be allocated. If the status (st) = 2 or 3, indicating that a job is being set up or is ready to start executing, then the last three numbers must be zeroes. If st = 0, indicating that a job has not been allocated any devices, then the last three must equal the previous three. If st = 1, indicating that a job has been allocated some of its devices, then the last three can be less than or equal to the previous three. A status of -1 means that no more jobs are in the system at Time = 0.

After reading in the list and making sure that st $\neq$ -1, initial increments n and prints back a message specifying the job number, status, and types of devices needed. It then gives the right number of devices to the job. I shall go step by step through the allocation of 9-track tapes, since the allocation of the other devices works exactly the same.

Since there are seven 9-track tapes, initial will go through the first do-group a maximum of seven times. However, each time it checks to see whether st = 0 or t9 = t9n.

(whether the job has been given all the 9-track tapes it needs). If both tests are negative, the first tape is checked to see whether it is idle. If so, it is given the same status, queue number, and position in the queue as the job. The number of 9-track tapes for the job (t9) is decremented by 1, as is the number of idle 9-track tapes (t9 idle). If st = 1, then the number of reserved tapes (t9 res) is incremented by 1. Execution then goes to the beginning of the block where the counter (corresponding to the tape number) is incremented and the conditions are retested. If the tape examined is not idle, then nothing is done, and initial passes on to the next tape.

After all of the devices have been allocated, initial activates job and passes it nine parameters: region, sri, class, run time, total number of devices, how many of each type are still needed, and status. Upon return, a reference name (steinberg) is given, and a pointer (obj ptr) is connected, to the job.

If the job is ready for execution, the execution activity is immediately activated. If the job is ready for setup, then execution is scheduled for three minutes in the future. In either case, obj ptr, job q, and steinberg are passed as arguments to identify the job.

At this point, initial loops back to again, and the cycle is repeated until a status of -1 is input. When this occurs, a message is printed saying that initialization has

been completed, and the activity ends. Initial is called only once in the simulation.

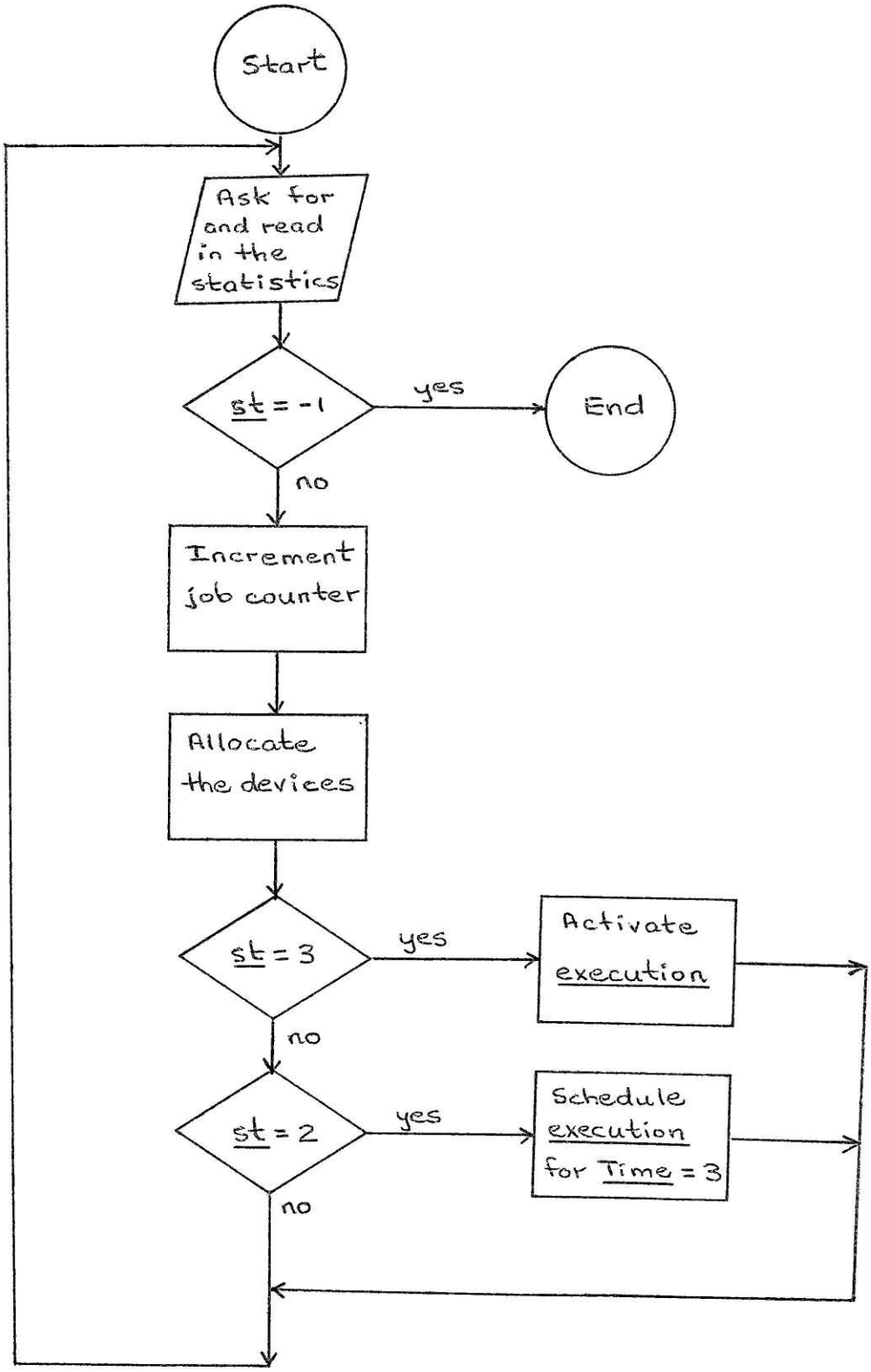


Figure 2: Flowchart of initial

Job

A new job is activated each time a job enters the system. In all cases the number of the job (number) is set equal to the current value of n, and the entry time (entime) is set equal to the current value of Time.

If start = "1"b, meaning that initialization is taking place, job checks to see if status = 2 or 3. In either of these cases, the bit alma is set on to indicate that the job has been allocated all its devices.

If start = "0"b, meaning that the job has been generated by sched, job must calculate the statistics about itself. First, it sets status = 0. Next, it figures out region, sri, classs, runtime, and setup (total number of devices). Then, depending upon this final value, it calculates the number of each type of device needed. For probabilities it uses the float binary lookup arrays declared at the end of sched.

Finally, regardless of the value of status, job files itself at the end of resqueue(sri + 1), then suspends itself. The "suspend" statement merely prevents execution of the statement "end job", which would wipe out the data base of the job and prevent us from ever accessing it. Even when we finish with a job and remove it from resqueue, the job activity and its data base remain (still suspended). Control then passes back to the calling activity.

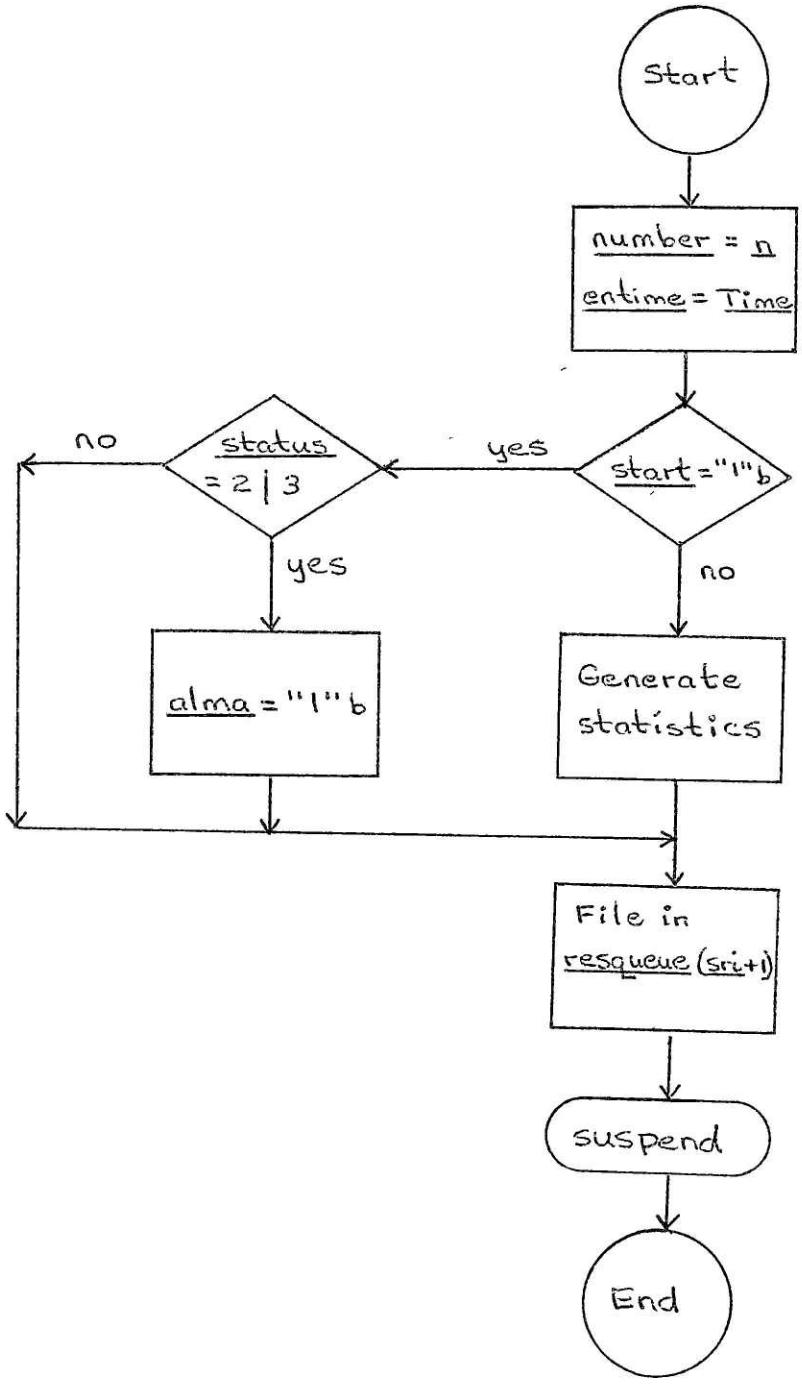


Figure 3: Flowchart of job

Mds

Mds is the heart of the model. It is activated by sched after a new job enters the system, or by release when a job frees its setup devices. However, it does not allocate devices to the jobs in the system at Time = 0; this is done by initial.

The first thing mds does is to perform two tests: it checks to see if the number of jobs already set up (num setup) has reached the maximum (depth), and if all devices have been given to jobs that are set up or executing. If either test is positive, the scan ends immediately.

Next, it starts at resqueue(16) and works downward, trying to find the first nonempty queue. If all queues are empty, the scan ends. If, however, it finds a queue with a job, it saves the queue number as i and transfers to chk set.

At chk set the number of jobs in resqueue(i) is saved as k, and j, the position-in-queue counter, is set to 1. Mds then checks to see if $j>k$, i.e. if it has scanned past the end of the queue. If so, the queue number i is decremented and tested for being 0. If $i = 0$, the bit jump is set on, indicating that all queues have been scanned, and mds transfers to chk age (which will be described soon).

If $j \leq k$, meaning that the last job in the queue has not been scanned, mds connects a pointer (job ptr) to the next job and prints out a message "*****,job number,alma,*****",

where alma is the bit telling whether or not all the devices have been allocated. If alma = "1"b, mds skips over the job by incrementing j and jumping back to next to check if j>k. If alma = "0"b, then mds tries to allocate the devices for the job.

The allocation is quite similar to the process described in initial. Again, I shall describe it for 9-track tapes, inasmuch as mds allocates them first and as the other allocations work the same.

Mds goes through the do-group a maximum of seven times, checking each time to make sure the number of tapes needed (job ptr → t9 needed) is positive. If the job still needs tapes, and the tape is idle, it is given to the job. The three indices in the tape array for that 9-track tape are given the values 1 (reserved), i, and j. Job ptr → t9 needed and t9 idle are decremented, while t9 res is incremented. If the tape is not idle but is reserved for a job in resqueue(i'), where i' ≤ i, then the new job takes the tape away from the old one. Again, the number of tapes needed by the new job is decremented, but the number needed by the old one is incremented. The program then loops to the beginning of the do-group.

After reserving as many devices as possible, mds looks to see if the job still needs more. If the answer is yes, the bit no ex is turned on, meaning that the job has been passed over (partially or totally). Otherwise, mds goes back and

changes the status of each device to 2 (setup), decrements the variables t9 res, t7 res, and d res by the proper amount, and turns job ptr alma on. It then gives the job a name (dick) and schedules it for execution in three minutes, passing execution the arguments job ptr, i, and dick. It also increments num setup by 1 and turns on the bit return to show that a job has been scheduled to execute. Mds then jumps to chk age before restarting the scan.

If the job was not given all its devices, mds checks to see if the job was first in its queue (j=1). If so, and if aging was specified, it turns the age bit on and sets q=i to save the queue number. It then increments j and looks at the next job by going to next.

Mds gets to chk age after it has finished scanning a queue or after a job is scheduled for execution. If age is off, nothing happens; if it is on, though, the job to be aged is moved up a queue.

The process is actually a bit complicated. First, s is set equal to the number of jobs in the next higher queue plus 1. Second, each tape and disk is examined. If it belongs to the job that is being aged, the device's queue number is incremented by 1 and its position is changed to s. If the device belongs to another job in the same queue, its position is decremented by 1, since the second job in the queue will now be the first, the third will be the second,etc. Last, mds removes the job from the lower queue, files it in the higher queue,

and turns off age.

After chk age, mds makes several checks. If return is on, mds turns it off and restarts the scan with chk dep. If jump is on or if no ex is on and the barrier has been passed, mds ends. Otherwise, it resumes the scan with the next queue by jumping to chk set.

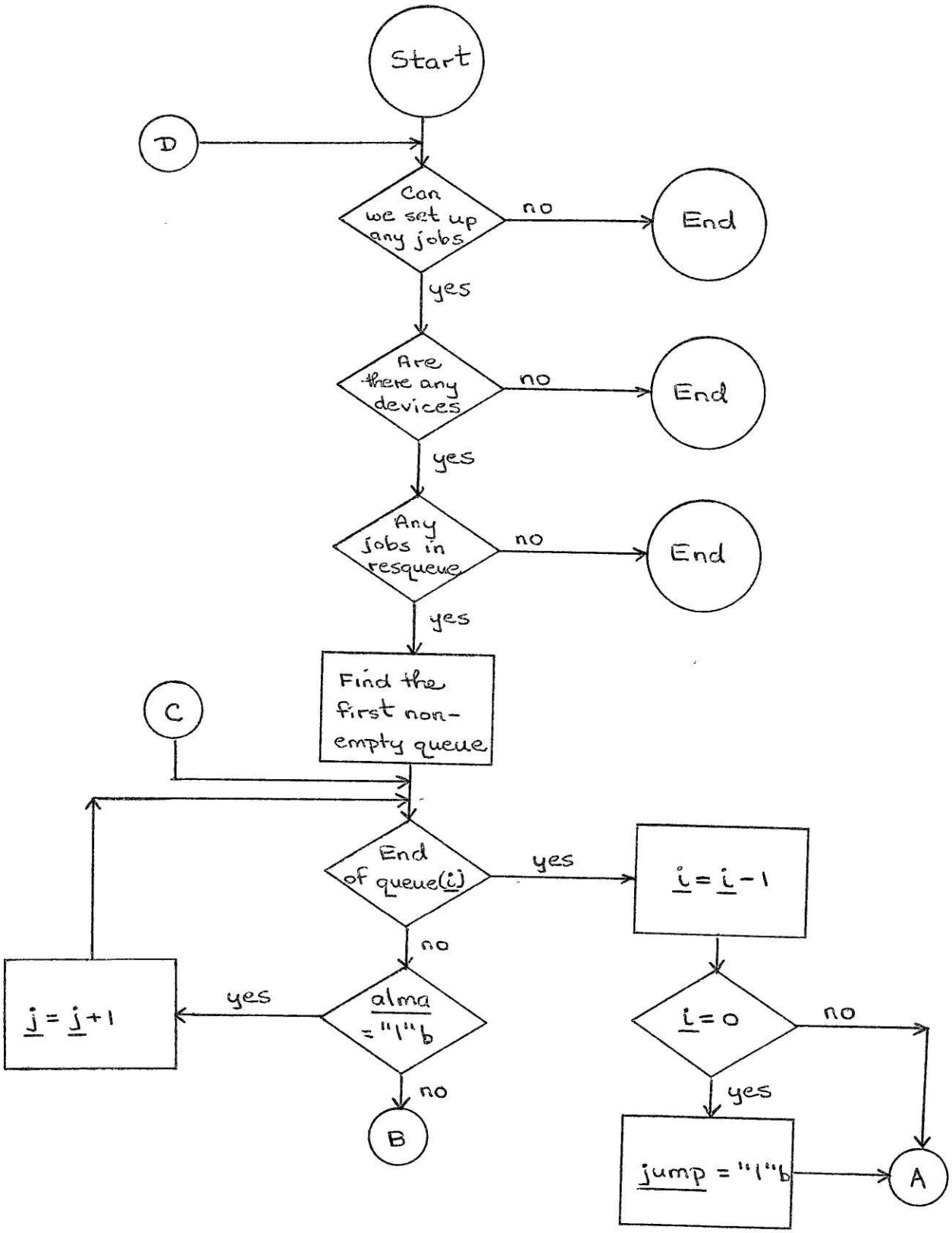


Figure 4: Flowchart of mds

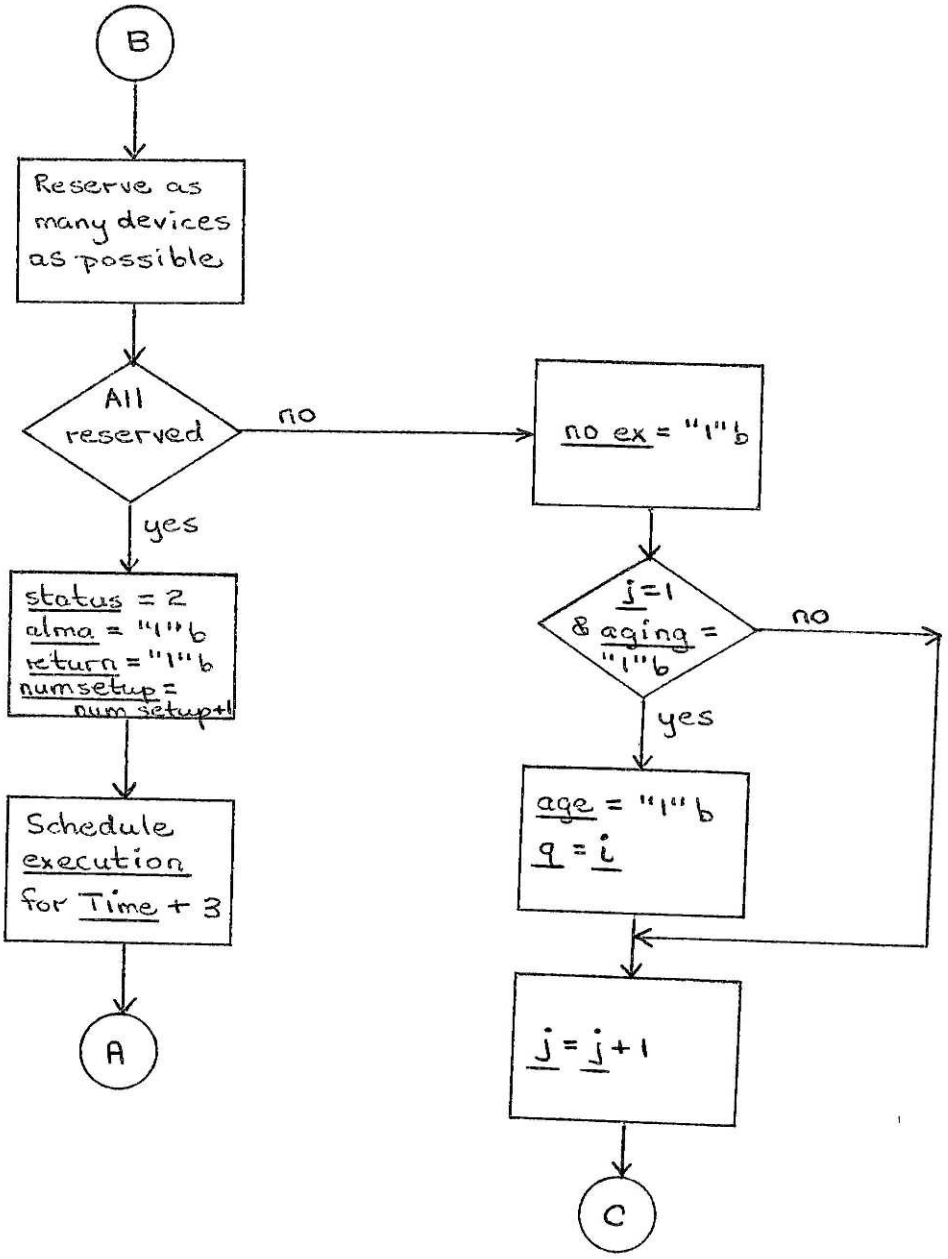


Figure 5: Flowchart of mds (cont.)

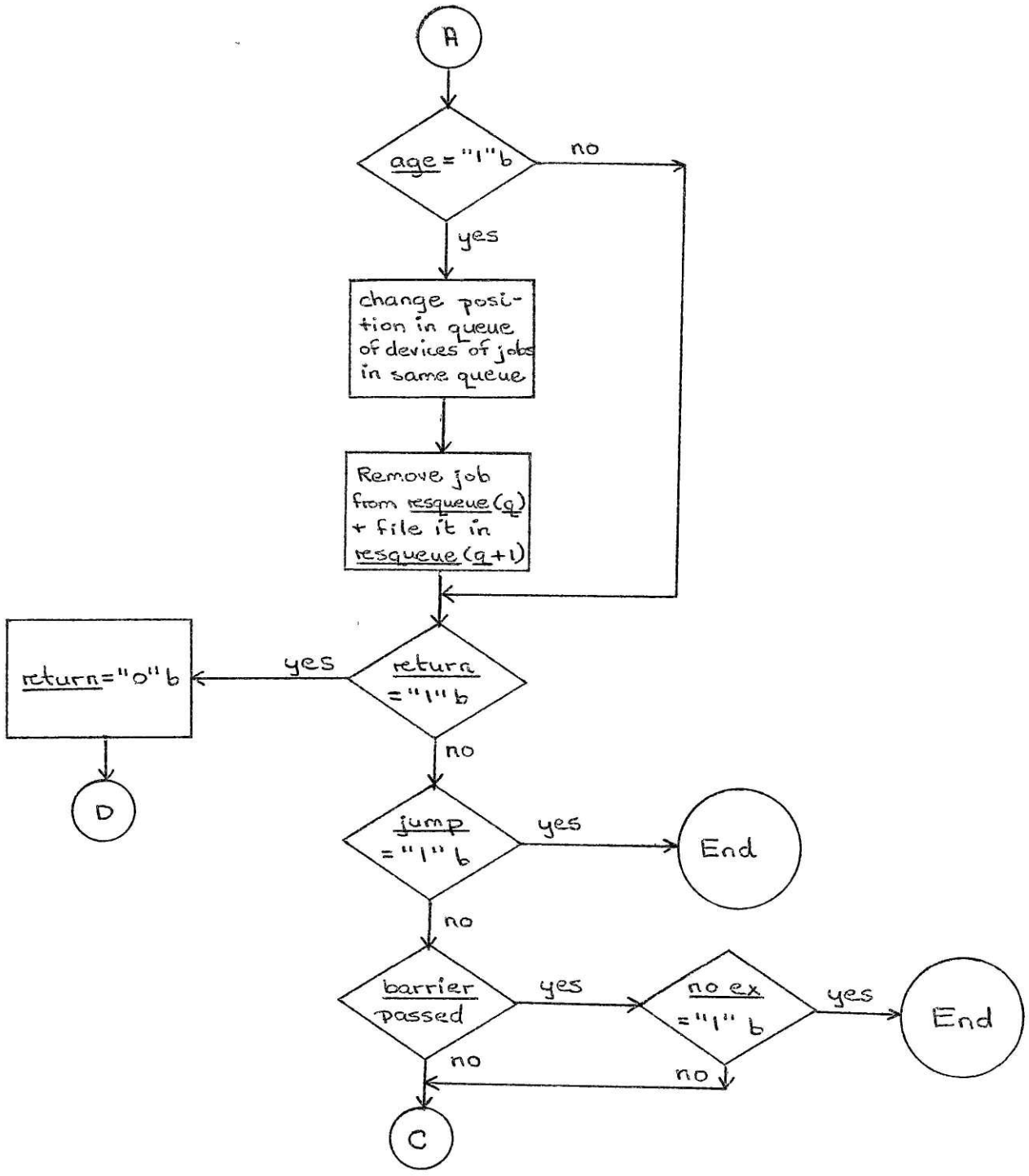


Figure 6: Flowchart of mds (cont.)

Execution

Execution is called when a job is about to start executing. If this occurs during initialization, it simply increments the job-executing counter (num exec), prints a message, and schedules release for Time + runtime. If, however, it is called by mds, it does quite a bit more. First, it checks to see if num exec has reached the limit (job depth). If so, it enters the job in a waiting queue (wait q) according to priority, then suspends itself. Later, release will reactivate execution, at which point the job will begin executing.

The above point is slightly ambiguous. In the actual system it is the job that has to wait. However, in the model the process execution is actually filed in wait q. When release activates the first process in wait q, it is "waking up" the execution corresponding to the job with the highest priority.

Execution reaches the statement "num exec = num exec + 1" after it has been woken up, or when num exec is less than job depth. It increments the number of jobs executing, changes the job's status to 3, finds the new position of the job in resqueue(qno), and changes the status of all the devices the job uses to 3 (executing). Before it ends it schedules release at Time + runtime.

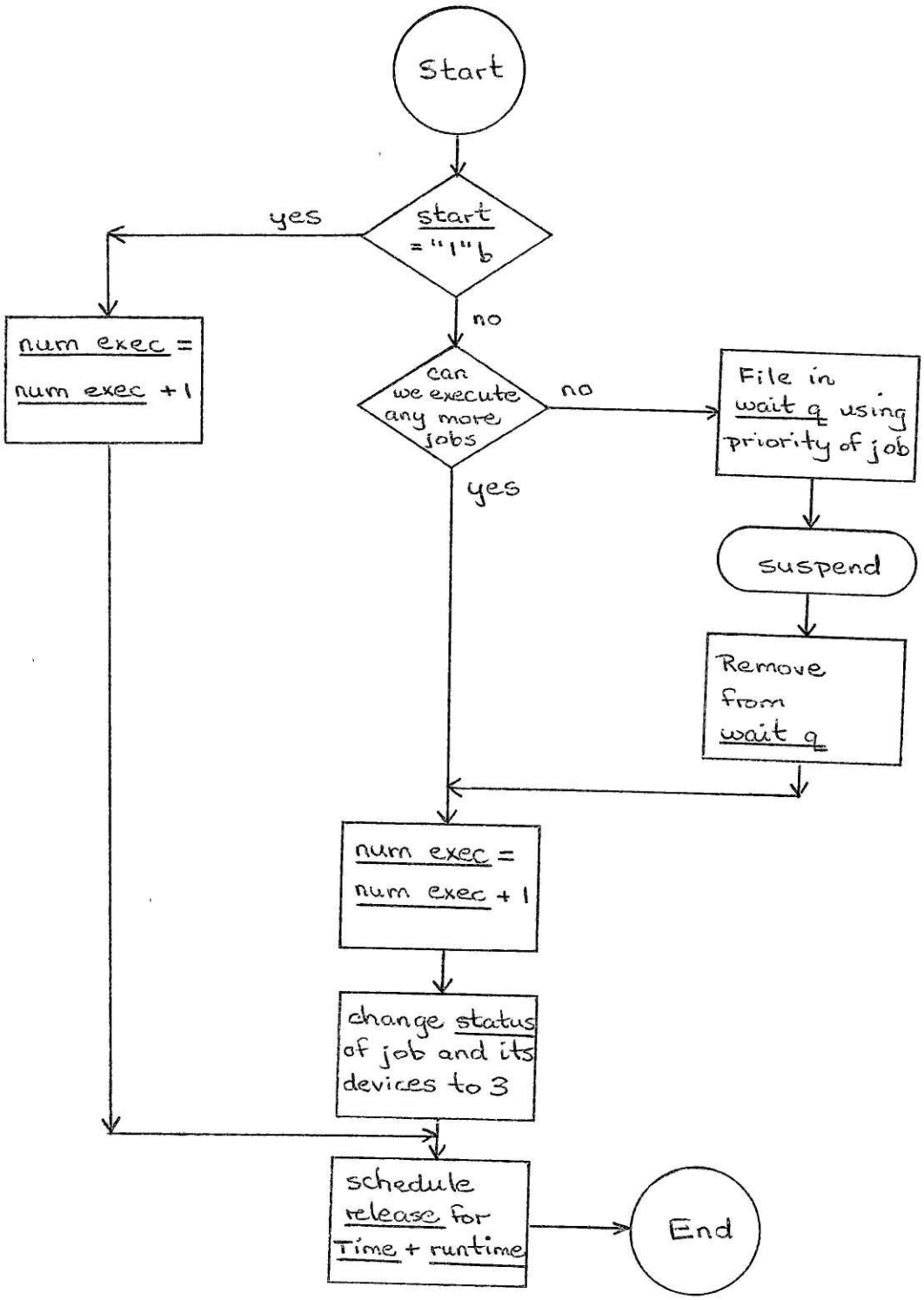


Figure 7: Flowchart of execution

Release

A new release is activated each time a job finishes executing. Execution passes it a connector to the job (rls ptr), its queue number (the q), and a reference names (jones).

First, release finds the position of the job in its queue (the pos), removes it from resqueue(the q), and decrements num exec. Second, it sets to 0 the row of three statistics having to do with each device the job used and increments t9 idle, t7 idle, and d idle by the proper amounts. Then, since all the other jobs in resqueue(the q) that had position lower than the pos are now moved up one, release decrements by 1 the position of each device used by one of these latter jobs. Third, release looks at wait q. If there are jobs waiting to be executed, it activates the execution corresponding to the job at the top of the queue. Last, it activates a new scan by mds, then ends.

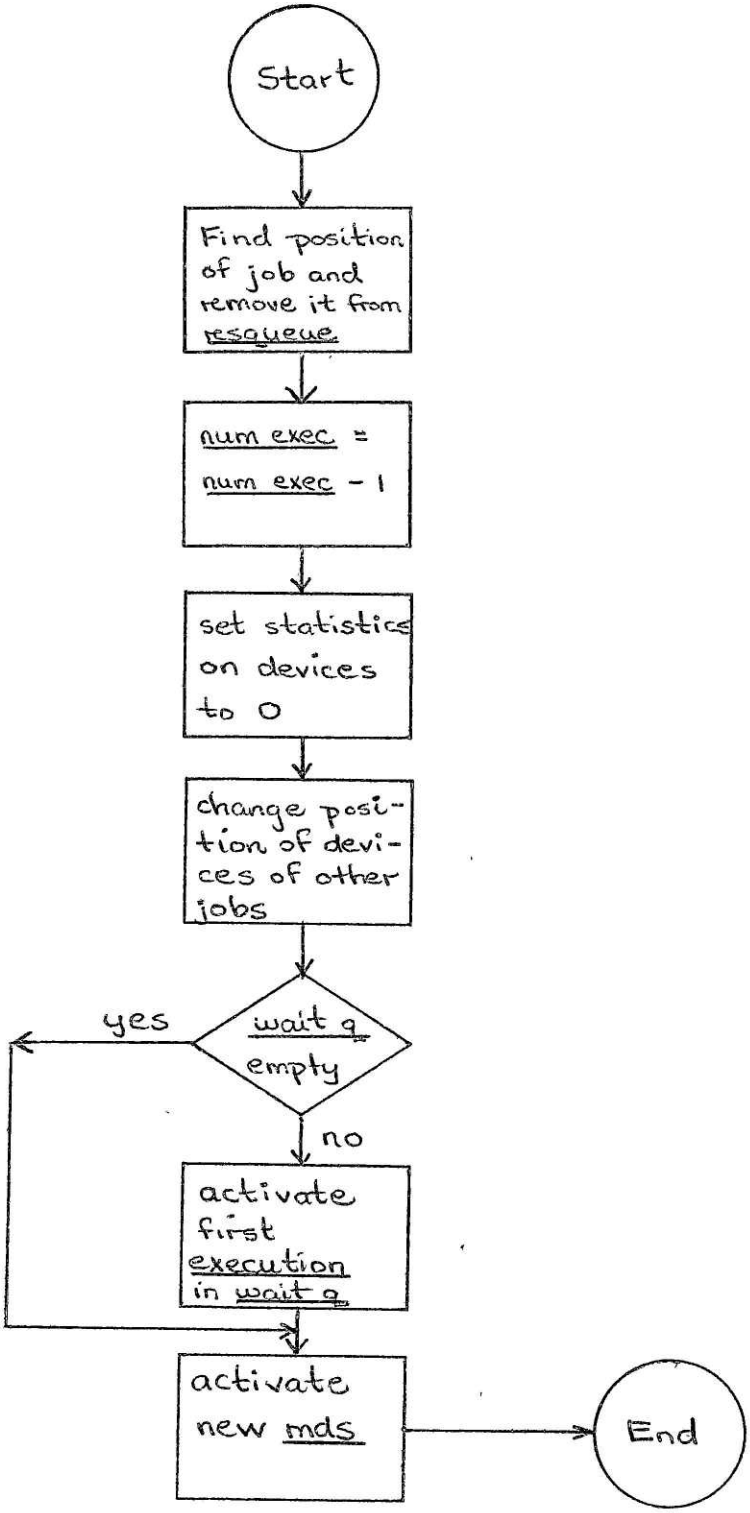


Figure 8: Flowchart of release

Sample

Sample is first activated by sched at Time = 15. It then reschedules itself every fifteen minutes. In addition, it is activated again by sched at the end of the simulation. Its purpose is to give a listing of all jobs in the resident job queue so that the user can check their progress.

Its logic is straightforward. It looks at all resqueue(p), $16 \geq p \geq 1$. When it finds a nonempty queue, it prints out statistics on each job, resetting a pointer (sptr) each time so that it can reference the data base of each job. If the paper in the terminal has been set correctly, the list will be printed on a new page.

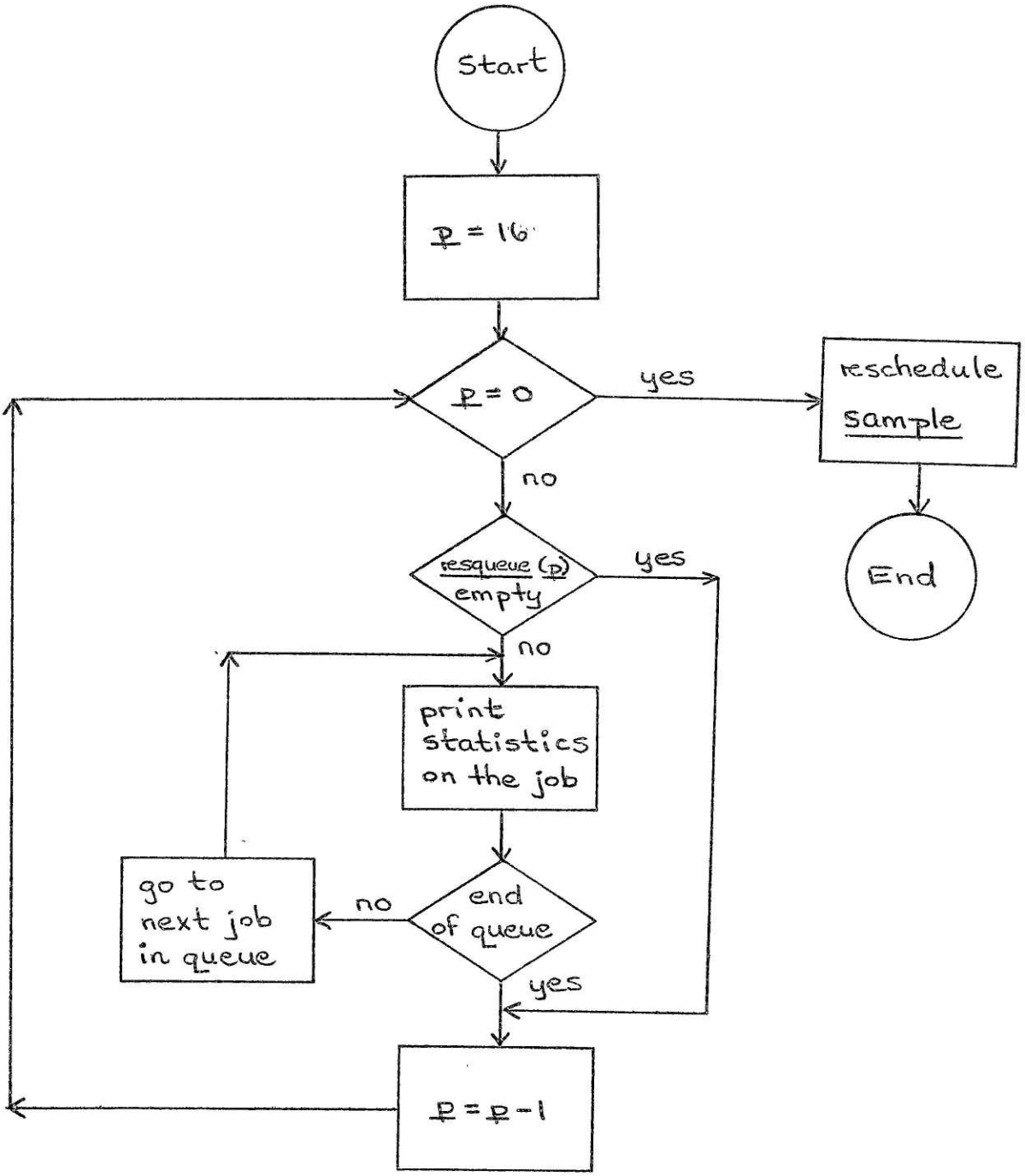


Figure 9: Flowchart of sample

Testing and Results

The model was run with the following parameters held constant: aging = "0"b, barrier = 0, depth = 15, increment = 3, and job depth = 5. The last three, the most important, are the normal values used at the Information Processing Center. The mean interarrival time used was that of shift 1, i.e. daytime usage. To facilitate the explanation, I will use as an example a run with runtime = 60 (one hour); the actual output is included as Appendix IV.

To purposely overload the system, I initialized it with eight jobs that used all the devices except for one disk (by chance, the first job generated took that, too). These jobs had execution times of up to seven minutes. I was quite sure that this overloading would cause a long lag in execution times.

For the first five minutes, five jobs were constantly executing, with between zero and three in the waiting queue (see figure 10). However, after six minutes, no more jobs were ever in the waiting queue, and the maximum number of jobs in the system was six. At about nine minutes, there were four jobs executing, one below the maximum; afterwards, there were never more than three executing at once. At one point, between sixteen and twenty-four minutes, there were no jobs in the system at all.

Similar results can be obtained by examining the device

usage graph (figure 11). For the first four minutes, nine or ten disks were used. Thereafter, it dropped steadily (although not monotonically) to a low of zero at sixteen minutes. During the rest of the simulation, the number of disks used never exceeded six. The same can be said for the 9-track tapes. From at high of seven for the first few minutes, the number of tapes used also fell to zero (since there were no jobs in the system), then only rose as high as six for a brief period. The 7-track tapes spent about half the time working (usually just one) and the other half idle.

The results are clear. Except for the initial flurry, three disks and one 9-track tape were never used, while two more 9-track tapes were only used for about five minutes. Removing three disks and a 9-track tape might prolong the "calming down" period at the beginning, but would hardly affect the system's efficiency for the rest of the time. On the other hand it would probably be best not to remove a 7-track tape for two reasons. First, there are periods of several minutes when both tapes are in use, and second, doing so would prevent anyone from running a job that used two 7-track tapes. The probability of someone using both tapes is small, but not small enough to be ignored.

Whereas the above results follow in a straightforward manner from the simulation, one should not hasten to throw out IPC's disks and tapes. In the first place, my limited budget prevented me from doing the comprehensive testing

that I would have liked to do. In the second place, the program for gathering statistics on variables in the simulation is in the process of being implemented. When it is working, it will calculate the usage statistics more accurately (and much more easily) than I have done. Then we may be able to draw firm conclusions as to the supply of devices at IPC.

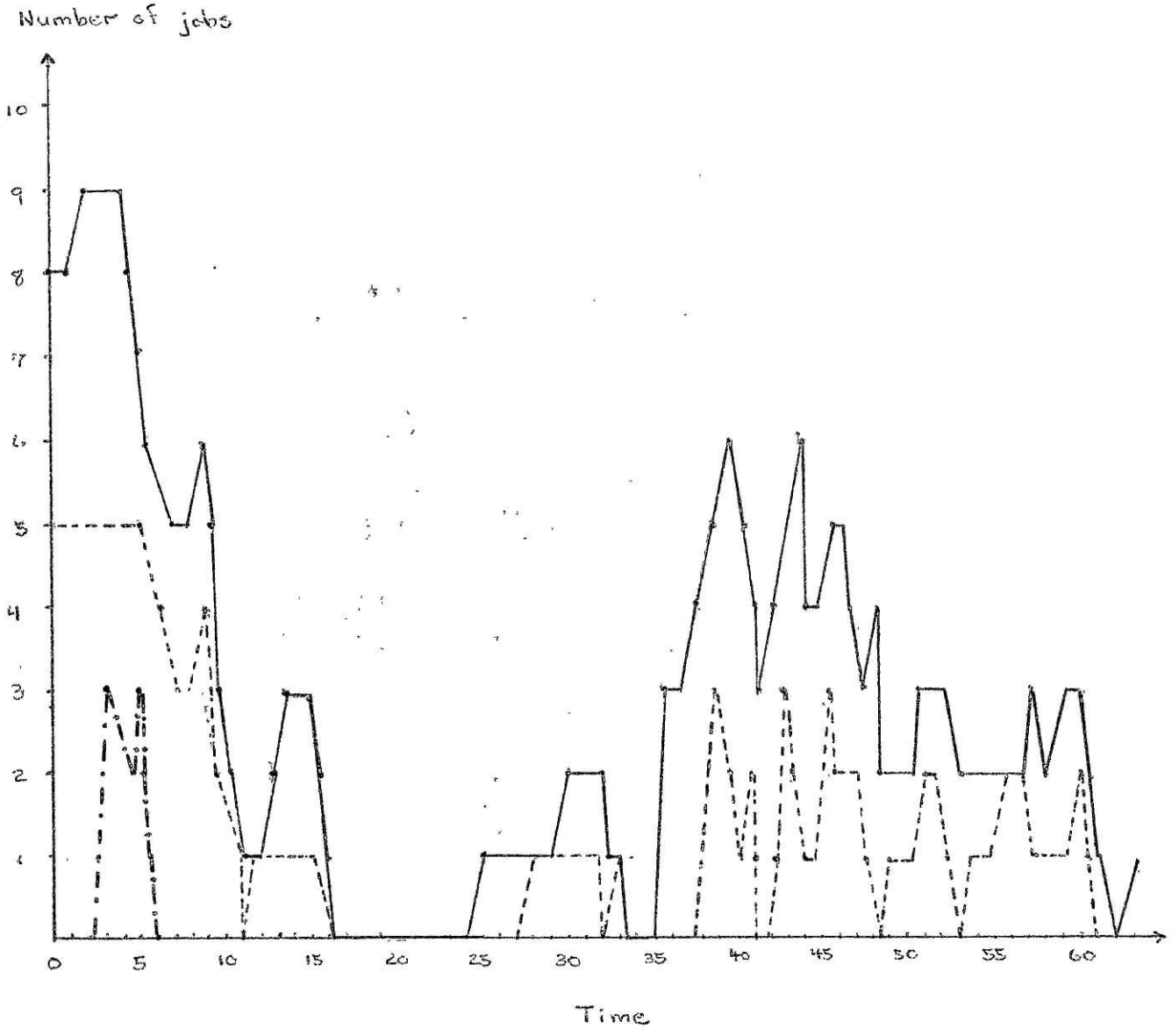
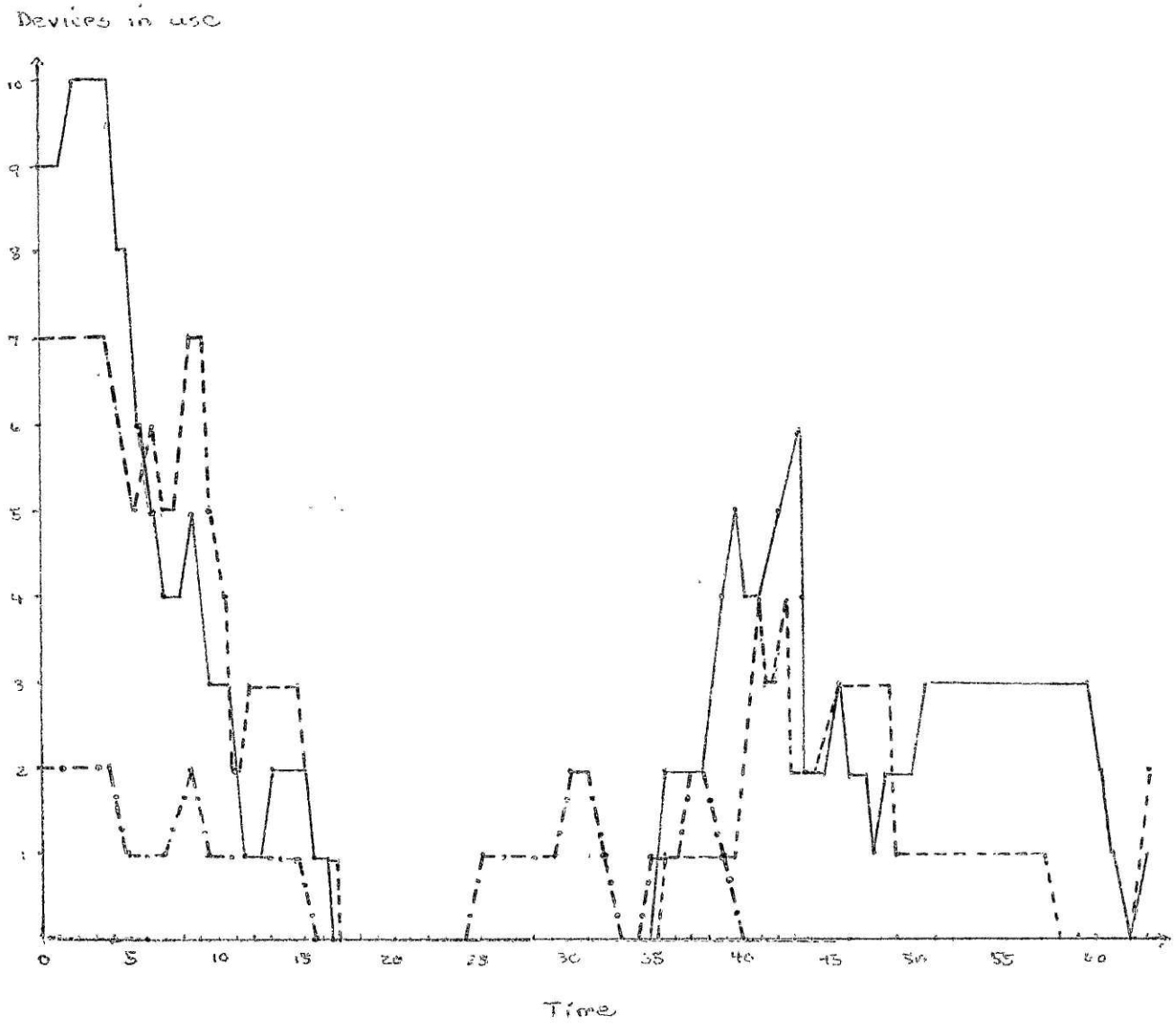

Number of jobs
10
9
8
7
6
5
4
3
2
1
0
5
10
15
20
25
30
35
40
45
50
55
60
Time
Legend :
jobs in system
jobs executing
jobs in wait_q

Figure 10



Legend:

——— disks

– – – – 9-track tapes

—·—·— 7-track tape

Figure 11

Appendix I: How to use the Model

The SIMPL language is currently only implemented on the Multics system. Therefore, step 1 below corresponds to a command that would change from system to system.

step 1: Type in: ssd >udd>SIMPLE>system with no blanks between the first and last ">". This sets the search directory so that the supervisor will know where to look for the keyword "sim" in step 2. This command only has to be typed in once, no matter how many times the model is run in a sitting.

step 2: Type in: sim sched aging barrier depth increment runtime job depth where

aging is a bit with value "0"b = aging off, "1"b = aging on;

barrier is an integer from 0 to 16, with 16 being BARRIER=PRTY;

depth is an integer telling how many jobs can be set up at once;

increment is an integer telling how many queues to move a job up once it has been allocated all its devices;

runtime is a float-point number telling the length of the simulation in minutes;

depth is an integer telling how many jobs can execute

at once.

There are blanks but no commas between the values. A normal command might be

sim sched "0"b 4 15 3 2.4e2 5

step 3: The terminal will respond with the message
PLEASE TYPE IN THE STATISTICS ABOUT THE JOB
The user responds with the statistics:
status, queue number, position in the queue, region, class, runtime, total number of devices, number of 9-track tapes, number of 7-track tapes, number of disks, number of 9-track tapes yet to be allocated, number of 7-track tapes yet to be allocated, and number of disks yet to be allocated.
For special values of status and class see Appendix II.
The numbers must be separated by blanks. A typical response might be:
3 6 1 128 2 3.0e0 2 0 0 2 0 0 0
meaning that the job is ready to execute, is at the top of queue 6, class B, 128 K, 3 minutes of execution, uses 2 devices, both are disks, and has no devices yet to be allocated (note: if status is 2 or 3, the last three numbers are 0).
Another might be:
1 4 2 200 111.0e0 3 2 1 0 0 1 0
meaning that it has been allocated some devices, is

in queue 4, second in the queue, 200 K, class A, 1 minute of executing, uses 3 devices, of which 2 are 9-track tapes and 1 is a 7-track, and it still needs the 7-track tape.

The console will continue to ask for statistics after reading in each set. When the user wishes to end the initialization, he types in

-1 0 0 0 0 0 0 0 0 0 0 0

Actually, the twelve zeroes can be any numbers as they are ignored, but the -1 must come first. After this, the simulation will execute without further help from the user.

Appendix II: Source Listings of Activities

```
sched:  simpl (aging,barrier,depth,increment,simtime,job_depth);
         dcl aging bit(1),
            barrier fixed bin,
            depth fixed bin,
            increment fixed bin,
            simtime float bin,
            n fixed bin init(0),
            tape9(7,3) fixed bin stat init((21)0),
            tape7(2,3) fixed bin stat init(( 6)0),
            disk(10,3) fixed bin stat init((30)0),
            t9_idle fixed bin init(7),
            t7_idle fixed bin init(2),
            d_idle  fixed bin init(10),
            t9_res fixed bin init(0),
            t7_res fixed bin init(0),
            d_res fixed bin init(0),
            resqueue(16) set queue,
            num_exec fixed bin init(0),
            job_depth fixed bin,
            wait_q set queue,
            start bit(1) init("1"b),
            classx(3) float bin init(3.45e1,8.47e1,1.0e2),
            classy(3) float bin init(1.0e0,2.0e0,3.0e0),
            srix(5) float bin init(8.7e0,3.36e1,9.26e1,9.99e1,1.0e2),
            sriy(5) float bin init(0.0e0,1.0e0,3.0e0,5.0e0,12.0e0),
            regx(13) float bin init(7.9e0,1.37e1,2.03e1,2.58e1,5.36e1,6.06e1,6.38e1,8.13e1,8.39e1,9.23e1,9.48e1,
                 9.75e1,1.0e2),
            regy(13) float bin init(0.0e0,7.60e1,1.0e2,1.25e2,1.3e2,1.5e2,1.75e2,2.0e2,2.25e2,2.5e2,2.75e2,3.0e2,3.5e2),
            setx(6) float bin init(7.38e1,9.23e1,9.65e1,9.89e1,9.97e1,1.0e2),
            sety(6) float bin init(1.0e0,2.0e0,3.0e0,4.0e0,5.0e0,6.0e0),
            set1x(3) float bin init(2.43e1,5.58e1,1.0e2),
            set1y(3) float bin init(0.0e0,1.0e0,2.0e0),
            set2x(3) float bin init(1.7e-2,6.1e1,1.0e2),
            set3x(2) float bin init(3.48e1,1.0e2),
            set3y(2) float bin init(1.0e0,2.0e0),
            job activity(fixed bin,fixed bin,fixed bin,float bin,fixed bin,fixed bin,fixed bin,fixed bin,
                  fixed bin) external,
            (mds,initial,execution,release,sample) activity external;

         /* initialize the system */

         activate new initial;
         start = "0"b;
         schedule new sample for 1.5e1;

generate:
         if Time >= simtime then do;
              activate new sample;
              endsim;
            end;
         delay expon(1.77e0);  /* average interarrival time of 106 seconds */
         n = n + 1;
         put edit ("Job number ",n," arrives in system at ") (skip(2),column(1),a,f(5),a);
         put data (Time);
         activate new job (0,0,0,0.0e0,0,0,0,0,0);
         activate new mds;
         goto generate;
         end sched;
```


r 1109 2.913 30+68

```
initial: activity;
         dcl (reg,job_q,cl,numd,t9n,t7n,dn,st,pos_in_q,t9,t7,d,c) fixed bin,
             runt float bin,
             obj_ptr connector(job),
             steinberg reference,
             job activity(fixed bin,fixed bin,fixed bin,float bin,fixed bin,fixed bin,fixed bin,
                  fixed bin,fixed bin) external,
             execution activity(pointer,fixed bin,pointer) external;

again:
         put edit ("PLEASE TYPE IN THE STATISTICS ABOUT THE JOB") (skip(2),column(1),a);
         get list (st,job_q,pos_in_q,reg,cl,runt,numd,t9,t7,d,t9n,t7n,dn);
         if st =  -1 then goto eoi;
         n = n + 1;
         put edit ("Job number ",n," is initially in the system with status ",st," and uses ",t9,
              " 9-track tapes, ",t7," 7-track tapes, and ",d," disks") (skip(2),column(1),
             a,f(5),a,f(5),a,f(5),a,f(5),a,f(5),a);
         do c = 1 to 7 while (st ¬= 0 & t9 ¬= t9n);
              if tape9(c,1) = 0 then do;
              tape9(c,1) = st;
              tape9(c,2) = job_q;
              tape9(c,3) = pos_in_q;
              t9 = t9 - 1;
              t9_idle = t9_idle - 1;
              if st = 1 then t9_res = t9_res + 1;
              end;
              end;
         do c = 1 to 2 while (st ¬= 0 & t7 ¬= t7n);
              if tape7(c,1) = 0 then do;
              tape7(c,1) = st;
              tape7(c,2) = job_q;
              tape7(c,3) = pos_in_q;
              t7 = t7 - 1;
              t7_idle = t7_idle - 1;
              if st = 1 then t7_res = t7_res + 1;
              end;
              end;
         do c = 1 to 10 while (st ¬= 0 & d ¬= dn);
              if disk(c,1) = 0 then do;
              disk(c,1) = st;
              disk(c,2) = job_q;
              disk(c,3) = pos_in_q;
              d = d - 1;
              d_idle = d_idle - 1;
              if st = 1 then d_res = d_res + 1;
              end;
              end;
         activate new job(reg,job_q - 1,cl,runt,numd,t9,t7,d,st);
         steinberg = set_ref(pos_in_q,resqueue(job_q));
         connect obj_ptr to steinberg;

         /* if the job is ready to be executed, schedule it for execution.  if it is setup,        */
         /* schedule it for execution in 3 minutes, the average setup time                          */

         if st = 3 then activate new execution(obj_ptr,job_q,steinberg);
         else if st = 2 then schedule new execution(obj_ptr,job_q,steinberg) for Time + 3.0e0;
         goto again;
eoi:
         put edit ("Initialization has been completed") (skip(2),column(1),a);
         end initial;
```

```
job:    activity (region,sri,classs,runtime,setup,t9_needed,t7_needed,d_needed,status);
        dcl region fixed bin,
            sri fixed bin,        /* 0,1,3,5,12 = bottom,low,standard,high,emergency */
            classs fixed bin,     /* 1,2,3 = A,B,C                                    */
            runtime float bin,
            setup fixed bin,      /* total number of setup devices needed             */
            t9_needed fixed bin,  /* number of 7- or 9-track                          */
            t7_needed fixed bin,  /* tape drives or 3330 disks                        */
            d_needed fixed bin,   /* needed for the job                               */
            number fixed bin,     /* job number, assigned sequentially from 1         */
            entime float bin,
            z fixed bin,
            alma bit(1) init("0"b),
            status fixed bin;     /* 0,1,2,3 = no devices allocated, some devices     */
                                  /* allocated, all devices allocated and awaiting    */
                                  /* execution, executing                             */

        number = n;
        entime = Time;

        /* if this is not one of the initial jobs in the system, generate the       */
        /* statistics on it                                                         */

        if start = "1"b & (status = 2 | status = 3) then alma = "1"b;
        if start = "0"b then do;
            status = 0;
            region = c_lookup(regx,regy,uniform(0,100));
            sri = d_lookup(srix,sriy,uniform(0,100));
            classs= d_lookup(classx,classy,uniform(0,100));
            runtime = expon(1.987e0);
            setup = d_lookup(setx,sety,uniform(0,100));
            z = d_lookup(set1x,set1y,uniform(0,100));
            if z = 0 then t7_needed = t7_needed + 1;
            else if z = 1 then t9_needed = t9_needed + 1;
            else d_needed = d_needed + 1;
            if setup = 1 then goto eoj;
            z = d_lookup(set2x,set1y,uniform(0,100));
            if z = 0 then t7_needed = t7_needed + 1;
            else if z = 1 then t9_needed = t9_needed + 1;
            else d_needed = d_needed + 1;
            if setup = 2 then goto eoj;
            z = d_lookup(set3x,set3y,uniform(0,100));
            if z = 1 then t9_needed = t9_needed + 1;
            else d_needed = d_needed + 1;
            if setup = 3 then goto eoj;
            d_needed = d_needed + 1;
            if setup = 4 then goto eoj;
            d_needed = d_needed + 1;
            if setup = 5 then goto eoj;
            d_needed = d_needed + 1;
eoj:
            put edit ("it needs ",t7_needed," 7-track tapes, ",t9_needed," 9-track tapes, and ",
                      d_needed," disks") (skip(2),column(1),a,f(5),a,f(5),a,f(5),a);
            put edit ("region = ",region) (column(1),a,f(5));
            put edit ("sri = ",sri) (column(1),a,f(5));
            put edit ("run time = ",runtime) (column(1),a,f(5,2));
            end;

        /* file the job in the resident job queue according to sri                  */

        enter Current in resqueue(sri + 1);
```

```
mds:        activity;
            dcl (i,j,k,m,q,s) fixed bin,
                age bit(1) init("0"b),
                jump bit(1) init("0"b),
                return bit(1) init ("0"b),
                job_ptr connector(job),
                dick reference,
                (dummy1,dummy2,dummy3) reference,
                no_ex bit(1),
                execution activity(pointer,fixed bin,pointer) external;

            put edit ("MDS begins scan of resident job queue at ") (skip(2),column(1),a);
            put data (Time);

            /* check to see if any more jobs can be set up                                    */

chk_dep:
          no_ex = "0"b;
            if num_setup = depth then do;
                put edit ("No more jobs can be set up") (skip(2),column(1),a);
                goto the_end;
                end;

            /* check to see if there are any devices left */

chk_dev:
            if (t9_idle + t7_idle + d_idle + t9_res + t7_res + d_res) = 0 then do;
                put edit ("There are no more idle or reserved devices") (skip(2),column(1),a);
                goto the_end;
                end;

            /* find the first nonempty queue in the resident job queue */

            do i = 16 to 1 by -1;
                if set_count(resqueue(i)) ^= 0 then goto chk_set;
                end;
            put edit ("There are no more jobs in the system") (skip(2),column(1),a);
            goto the_end;

chk_set:
            j = 1;
            k = set_count(resqueue(i));
next:
            if j > k then do;
                i = i - 1;
                if i = 0 then jump = "1"b;
                goto chk_age;
                end;
            dummy1 = set_ref(j,resqueue(i));
            connect job_ptr to dummy1;
            put edit ("*****",job_ptr -> number,job_ptr -> alma,"*****") (skip(2),a,f(5),f(1),a);
            if job_ptr -> alma = "1"b then do;
                j = j + 1;
                goto next;
                end;

            /* try to reserve the devices for the job */

            do m = 1 to 7 while (job_ptr -> t9_needed > 0);
                if tape9(m,1) = 0 then do;
                    job_ptr -> status = 1;
```

```
                    tape9(m,2) = i;
                    tape9(m,3) = j;
                    job_ptr -> t9_needed = job_ptr -> t9_needed - 1;
                    t9_idle = t9_idle - 1;
                    t9_res = t9_res + 1;
                    end;
                else if tape9(m,1) = 1 & tape9(m,2) <= i & tape9(m,3) ¬= j then do;
                    dummy2 = set_ref(tape9(m,3),resqueue(tape9(m,2)));
                    connect job_ptr to dummy2;
                    job_ptr -> t9_needed = job_ptr -> t9_needed + 1;
                    tape9(m,2) = i;
                    tape9(m,3) = j;
                    connect job_ptr to dummy1;
                    job_ptr -> t9_needed = job_ptr -> t9_needed - 1;
                    job_ptr -> status = 1;
                    end;
            end;
        do m = 1 to 2 while (job_ptr -> t7_needed > 0);
                    if tape7(m,1) = 0 then do;
                        job_ptr -> status = 1;
                        tape7(m,1) = 1;
                        tape7(m,2) = i;
                        tape7(m,3) = j;
                        job_ptr -> t7_needed = job_ptr -> t7_needed - 1;
                        t7_idle = t7_idle - 1;
                        t7_res = t7_res + 1;
                        end;
                    else if tape7(m,1) = 1 & tape7(m,2) <= i & tape7(m,3) ¬= j then do;
                        dummy2 = set_ref(tape7(m,3),resqueue(tape7(m,2)));
                        connect job_ptr to dummy2;
                        job_ptr -> t7_needed = job_ptr -> t7_needed + 1;
                        tape7(m,2) = i;
                        tape7(m,3) = j;
                        connect job_ptr to dummy1;
                        job_ptr -> t7_needed = job_ptr -> t7_needed - 1;
                            job_ptr -> status = 1;
                            end;
                end;
    do m = 1 to 10 while (job_ptr -> d_needed > 0);
                if disk(m,1) = 0 then do;
                            job_ptr -> status = 1;
                            disk(m,1) = 1;
                            disk(m,2) = i;
                            disk(m,3) = j;
                            job_ptr -> d_needed = job_ptr -> d_needed - 1;
                            d_idle = d_idle - 1;
                            d_res = d_res + 1;
                            end;
                    else if disk(m,1) = 1 & disk(m,2) <= i & disk(m,3) ¬= j then do;
                            dummy2 = set_ref(disk(m,3),resqueue(disk(m,2)));
                            connect job_ptr to dummy2;
                            job_ptr -> d_needed = job_ptr -> d_needed + 1;
                            disk(m,2) = i;
                                    disk(m,3) = j;
                                    connect job_ptr to dummy1;
                                    job_ptr -> d_needed = job_ptr -> d_needed - 1;
                                    job_ptr -> status = 1;
                                    end;
                            end;

        /* are there enough devices free for the job?  if so, allocate them; if not, and if the    */
        /* job was first in the queue and aging was specified, bump it up                          */

        if job_ptr -> t9_needed + job_ptr -> t7_needed + job_ptr -> d_needed > 0 then no_ex = "1"b;
```

```
                    do m = 1 to 7;
                         if tape9(m,2) = i & tape9(m,3) = j then do;
                              tape9(m,1) = 2;
                              t9_res = t9_res - 1;
                              end;
                         end;
                    do m = 1 to 2;
                         if tape7(m,2) = i & tape7(m,3) = j then do;
                              tape7(m,1) = 2;
                              t7_res = t7_res - 1;
                              end;
                         end;
                    do m = 1 to 10;
                         if disk(m,2) = i & disk(m,3) = j then do;
                              disk(m,1) = 2;
                              d_res = d_res - 1;
                              end;
                         end;
                    dick = set_ref(j,resqueue(i));
                    schedule new execution(job_ptr,i,dick) for Time + 3.0e0;
                    job_ptr -> alma = "1"b;
put edit ("*****",job_ptr -> number,job_ptr -> alma,"*****") (skip(2),a,f(5),f(1),a);
                    num_setup = num_setup + 1;
                    return = "1"b;
                    goto chk_age;
                    end;
else if j = 1 & aging = "1"b then do;
                    age = "1"b;
                    dummy3 = first(resqueue(i));
                    q = i;
                    end;
j = j + 1;
goto next;

     /* here is where we actually age the job */

chk_age:
   if age = "1"b then do;
                    s = set_count(resqueue(q+1)) + 1;
                    do m = 1 to 7;
                         if tape9(m,2) = q & tape9(m,3) = 1 then do;
                              tape9(m,2) = q + 1;
                              tape9(m,3) = s;
                              end;
                         else if tape9(m,2) = q then tape9(m,3) = tape9(m,3) - 1;
                         end;
                    do m = 1 to 2;
                         if tape7(m,2) = q & tape7(m,3) = 1 then do;
                              tape7(m,2) = q + 1;
                              tape7(m,3) = s;
                              end;
                         else if tape7(m,2) = q then tape7(m,3) = tape7(m,3) - 1;
                         end;
                    do m = 1 to 10;
                         if disk(m,2) = q & disk(m,3) = 1 then do;
                              disk(m,2) = q + 1;
                              disk(m,3) = s;
                              end;
                         else if disk(m,2) = q then disk(m,3) = disk(m,3) - 1;
                         end;
                    age = "0"b;
                    enter first(resqueue(q)) in resqueue(q+1);
                    remove first(resqueue(q)) from resqueue(q);
                    end;
```

```
                    /* if a job has been scheduled for execution, restart the scan by this */

                    if return = "1"b then do;
                                    return = "0"b;
                                    goto chk_dep;
                                    end;

                    /* check to see if the barrier has been passed */

                    if jump = "1"b then goto the_end;
                                    if i <= barrier & no_ex = "1"b then goto the_end;
                    goto chk_set;

the_end:
               put edit ("Scan of resident job queue has ended") (skip(2),column(1),a);
                end mds;


r 1139  5.785  48+112
```

```
execution:activity (x_ptr,qno,mal);
        dcl x_ptr connector(job),
            (qno,qpos) fixed bin,     /* queue number and position in queue of job */
            b fixed bin,
            mal reference,
            release activity(pointer,fixed bin,pointer) external;

        /* if the system is being initialized, increment the number of jobs executing */

        if start = "1"b then num_exec = num_exec + 1;

        /* otherwise, see if there is room for another job to be executed.  if not, suspend it and */
        /* file it in a waiting queue.  if there is room for it, change its status and the status  */
        /* of its devices to 3 (executing) and schedule its release for current time + runtime     */

        else do;
             if num_exec = job_depth then do;
                  enter Current in wait_q using (qno + increment);
                   put edit ("Job number ",x_ptr -> number," has been put into the waiting queue at ")
                         (skip(2),column(1),a,f(5),a);
                   put data (Time);
                  suspend;
                  remove Current from wait_q;
                  end;
             num_exec = num_exec + 1;
             x_ptr -> status = 3;
             qpos= location(mal,resqueue(qno));
             do b = 1 to 7;
                  if tape9(b,2) = qno & tape9(b,3) = qpos then tape9(b,1) = 3;
                  end;
             do b = 1 to 2;
                  if tape7(b,2) = qno & tape7(b,3) = qpos then tape7(b,1) = 3;
                  end;
             do b = 1 to 10;
                  if disk(b,2) = qno & disk(b,3) = qpos then disk(b,1) = 3;
                  end;
             end;
         put edit ("Job number ",x_ptr -> number," begins executing at ")
                   (skip(2),column(1),a,f(5),a);
         put data (Time);
        schedule new release(x_ptr,qno,mal) for Time + x_ptr -> runtime;
        end execution;
```

r 1824 2.127 18+72

```
release:  activity(rls_ptr,the_q,jones);
          dcl rls_ptr connector(job),
              (the_q,the_pos) fixed bin,
              x fixed bin,
              execution activity external,
              jones reference,
              mds activity external;

          /* find the position in its queue of the job to be released */

          the_pos = location(jones,resqueue(the_q));

          /* remove the job from the resident job queue and decrement the number of jobs executing */

          remove jones from resqueue(the_q);
          num_exec = num_exec - 1;

          /* free up the devices it used.  any job it the same queue below the one just removed will */
          /* be moved up one position, so we have to change the position number on the devices       */
          /* that these latter jobs use                                                             */

          do x = 1 to 7;
               if tape9(x,2) = the_q then do;
                    if tape9(x,3) = the_pos then do;
                         tape9(x,1),tape9(x,2),tape9(x,3) = 0;
                         t9_idle = t9_idle + 1;
                         end;
                    if tape9(x,3) > the_pos then tape9(x,3) = tape9(x,3) - 1;
                    end;
               end;
          do x = 1 to 2;
               if tape7(x,2) = the_q then do;
                    if tape7(x,3) = the_pos then do;
                         tape7(x,1),tape7(x,2),tape7(x,3) = 0;
                         t7_idle = t7_idle + 1;
                         end;
                    if tape7(x,3) > the_pos then tape7(x,3) = tape7(x,3) - 1;
                    end;
               end;
          do x = 1 to 10;
               if disk(x,2) = the_q then do;
                    if disk(x,3) = the_pos then do;
                         disk(x,1),disk(x,2),disk(x,3) = 0;
                         d_idle = d_idle + 1;
                         end;
                    if disk(x,3) > the_pos then disk(x,3) = disk(x,3) - 1;
                    end;
               end;
          put edit ("Job number ",rls_ptr -> number," finishes execution at ") (skip(2),column(1),a,f(5),a);
          put data (Time);

          /* if there are any jobs waiting to be executed, free up the one with the highest         */
          /* priority, which really means activate one of the suspended execution processes         */

          if set_count(wait_q) > 0 then activate first(wait_q);
          activate new mds;
          end release;
```


r 1829 2.241 12+58

```
sample:  activity;
         dcl sptr connector(job),
             (p,g,the_count) fixed bin,
             andy reference,
             sample activity,
             klass char(1);

         put page;
         put edit ("Sample of resident job queue at ") (column(1),a);
         put data (Time);
          put edit (" Job  Queue   Entry  Class  Total dev.  9-tracks  7-tracks  disks  Status")
                   (skip(3),column(1),a);
         put edit ("time","for job","still needed for the job") (column(17),a,column(31),a,column(44),a);
         put skip(2);

         /* scan each queue, starting with the highest numbered (16).  if it is nonempty, */
         /* print the information about the jobs in it.                                    */

         do p = 16 to 1 by -1;
              the_count = set_count(resqueue(p));
              g = 1;
loop:
              if the_count > 0 then do;
                 andy = set_ref(g,resqueue(p));
                 connect sptr to andy;
                 if sptr -> classs = 1 then klass = "A";
                 else if sptr -> classs = 2 then klass = "B";
                 else klass = "C";
                 put edit (sptr -> number,p,sptr -> entime,klass,sptr -> setup,
                          sptr -> t9_needed,sptr -> t7_needed,sptr -> d_needed,sptr -> status)
                               (skip(1),column(1),f(5),column(7),f(5),column(16),f(5,2),column(26),
                               a(1),column(31),f(5),column(45),f(5),column(55),f(5),column(65),f(5),
                          column(70),f(5));
                 g = g + 1;
                 the_count = the_count - 1;
                 goto loop;
                 end;
              end;
         put page;
         schedule new sample for Time + 1.5e1;
        end sample;
```

```
r 1322  2.038  20+61
```

Appendix III: Statistics on Jobs

The following statistics are courtesy of Mr. Richard Steinberg of the MIT Information Processing Center. They are for jobs run in December, 1971, and were used as the probabilities declared as floating point arrays in sched. Job used these arrays to calculate the parameters on jobs generated by the model.

Interarrival time

Interarrival time is an exponential function with an expected value of 106 seconds during shift 1.

Run time

Run time is also exponential with an expected value of 119 seconds.

SRI

SRI	Number of jobs	Percent of jobs
defer	3170	8.7
low	9065	24.9
standard	20044	55.1
high	2670	7.3
emergency	27	0.1

In addition, 1396 jobs, or 3.9%, were run on either advanced scheduling or IPC priorities.

Class

34.5% of the jobs are class A, 50.2% are class B, and 15.3% are classC.

Region

Region size (K)	Percent of jobs
0 - 76	7.9
76 - 100	5.8
100 - 125	6.6
125 - 130	5.5
130 - 150	27.8
150 - 175	7.0
175 - 200	3.2
200 - 225	17.5
225 - 250	2.6
250 - 275	8.4
275 - 300	2.5
300 - 325	2.7
325 and up	2.5

Setups

	Number of Devices					
	1	2	3	4	5	6
jobs/percent	8373/73.8	2102/18.5	466/4.2	279/2.4	90/.8	29/.3

Types of Devices

	Total number of Devices					
	1	2	3	4	5	6
7-track tapes	856	5	0	0	0	0
9-track tapes	1106	173	8	0	0	0
disks	1158	114	15	5	0	0

Appendix IV: A Sample Run

The parameters for this run, along with the results, are listed on page 37. The last two pages of the run, "Fixed and Floating Point Statistics", are invalid, since the statistic collecting program has not been fully implemented yet. However, it has been left in to give the user an idea of what he will get when the program is implemented.

Bibliography

IBM Corporation, System 360 and System 370 ASP System V-2, Console Operator's Manual, Revision 8, White Plains, New York, March, 1971.

Jones, Malcolm M. and Thurber, Richard C., The SIMPL Primer, M.I.T., Cambridge, Mass., October, 1971.

Jones, Malcolm M. and Thurber, Richard C., The SIMPL Reference Manual, M.I.T., Cambridge, Mass., October, 1971.

M.I.T. Information Processing Center, GI18-revision 2, Cambridge, Mass., September, 1971.

```
ssd >udd>SIMPLE>system
r 1719  1.405  9+95

sim sched "0"b 0 15 3 6.0e1 5


PLEASE TYPE IN THE STATISTICS ABOUT THE JOB 3 13 1 350 1 5.5e0 4 2 0 2 0 0 0


Job number     1 is initially in the system with status     3 and uses
2 9-track tapes,     0 7-track tapes, and     2 disks

Job number     1 begins executing at    run_system$.Time= 0.00000000e+00;

PLEASE TYPE IN THE STATISTICS ABOUT THE JOB 3 6 1 128 2 4.5e0 2 0 0 2 0 0 0


Job number     2 is initially in the system with status     3 and uses
0 9-track tapes,     0 7-track tapes, and     2 disks

Job number     2 begins executing at    run_system$.Time= 0.00000000e+00;

PLEASE TYPE IN THE STATISTICS ABOUT THE JOB 3 6 2 200 2 5.0e0 1 0 1 0 0 0 0


Job number     3 is initially in the system with status     3 and uses
0 9-track tapes,     1 7-track tapes, and     0 disks

Job number     3 begins executing at    run_system$.Time= 0.00000000e+00;

PLEASE TYPE IN THE STATISTICS ABOUT THE JOB 3 4 1 300 1 7.0e0 2 1 0 1 0 0 0


Job number     4 is initially in the system with status     3 and uses
1 9-track tapes,     0 7-track tapes, and     1 disks

Job number     4 begins executing at    run_system$.Time= 0.00000000e+00;

PLEASE TYPE IN THE STATISTICS ABOUT THE JOB 3 2 1 386 3 6.0e0 1 0 1 0 0 0 0


Job number     5 is initially in the system with status     3 and uses
0 9-track tapes,     1 7-track tapes, and     0 disks

Job number     5 begins executing at    run_system$.Time= 0.00000000e+00;

PLEASE TYPE IN THE STATISTICS ABOUT THE JOB 2 2 2 100 2 5.2e0 2 0 0 2 0 0 0


Job number     6 is initially in the system with status     2 and uses
0 9-track tapes,     0 7-track tapes, and     2 disks

PLEASE TYPE IN THE STATISTICS ABOUT THE JOB 2 2 3 76 3 4.0e0 2 2 0 0 0 0 0


Job number     7 is initially in the system with status     2 and uses
2 9-track tapes,     0 7-track tapes, and     0 disks

PLEASE TYPE IN THE STATISTICS ABOUT THE JOB 2 2 4 250 1 5.0e0 4 2 0 2 0 0 0


Job number     8 is initially in the system with status     2 and uses
2 9-track tapes,     0 7-track tapes, and     2 disks
```

PLEASE TYPE IN THE STATISTICS ABOUT THE JOB -1 0 0 0 0 0 0 0 0 0 0 0 0

Initialization has been completed

Job number 9 arrives in system at run_system$.Time= 1.76978980e+00;Lookup: input value 8.60257030 less than lowest ¢ce entry.

it needs 0 7-track tapes, 0 9-track tapes, and 1 disks
region = 184
sri = 3
run time = 1.39

MDS begins scan of resident job queue at run_system$.Time=
1.76978980e+00;

***** 11*****

***** 21*****

***** 31*****

***** 41*****

***** 90*****

***** 91*****

There are no more idle or reserved devices

Scan of resident job queue has ended

Job number 6 has been put into the waiting queue at
run_system$.Time= 3.00000000e+00;

Job number 7 has been put into the waiting queue at
run_system$.Time= 3.00000000e+00;

Job number 8 has been put into the waiting queue at
run_system$.Time= 3.00000000e+00;

Job number 2 finishes execution at run_system$.Time= 4.50000000e+00;

Job number 6 begins executing at run_system$.Time= 4.50000000e+00;

MDS begins scan of resident job queue at run_system$.Time=
4.50000000e+00;

***** 11*****

***** 31*****

***** 41*****

***** 91*****

***** 51*****

***** 61*****

***** 71*****

***** 81*****

```
Job number     9 has been put into the waiting queue at
run_system$.Time= 4.76978976e+00;

Job number     3 finishes execution at  run_system$.Time= 5.00000000e+00;
Job number     9 begins executing at    run_system$.Time= 5.00000000e+00;

MDS begins scan of resident job queue at          run_system$.Time=
 5.00000000e+00;

*****    11*****

*****    41*****

*****    91*****

*****    51*****

*****    61*****

*****    71*****

*****    81*****

Scan of resident job queue has ended

Job number     1 finishes execution at  run_system$.Time= 5.50000000e+00;

Job number     7 begins executing at    run_system$.Time= 5.50000000e+00;

MDS begins scan of resident job queue at          run_system$.Time=
 5.50000000e+00;

*****    41*****

*****    91*****

*****    51*****

*****    61*****

*****    71*****

*****    81*****

Scan of resident job queue has ended

Job number    10 arrives in system at   run_system$.Time= 5.99783278e+00;

it needs     0 7-track tapes,     1 9-track tapes, and     1 disks
region =   226
sri =     1
run time =  1.37

MDS begins scan of resident job queue at          run_system$.Time=
 5.99783278e+00;

*****    41*****

*****    91*****

*****    51*****

*****    61*****
```

```
*****    81*****
*****   100*****
*****   101*****
*****    41*****
*****    91*****
*****    51*****
*****    61*****
*****    71*****
*****    81*****
*****   101*****
Scan of resident job queue has ended
Job number     5 finishes execution at  run_system$.Time= 6.00000000e+00;
Job number     8 begins executing at    run_system$.Time= 6.00000000e+00;
MDS begins scan of resident job queue at          run_system$.Time=
 6.00000000e+00;
*****    41*****
*****    91*****
*****    61*****
*****    71*****
*****    81*****
*****   101*****

Scan of resident job queue has ended

Job number    11 arrives in system at   run_system$.Time= 6.23002315e+00;Lookup: input value 36.6423993 less than lowest t
ce entry.
Lookup: input value 12.8025961 less than lowest table entry.


it needs     1 7-track tapes,     0 9-track tapes, and     0 disks
region =   138
sri =     3
run time =  0.16

MDS begins scan of resident job queue at          run_system$.Time=
 6.23002315e+00;

*****    41*****
*****    91*****
*****   110*****
*****   111*****
*****    41*****
```

```
*****    91*****

*****   111*****

*****    61*****

*****    71*****

*****    81*****

*****   101*****

Scan of resident job queue has ended

Job number     9 finishes execution at  run_system$.Time= 6.39376748e+00;

MDS begins scan of resident job queue at          run_system$.Time=
 6.39376748e+00;

*****    41*****

*****   111*****

*****    61*****

*****    71*****

*****    81*****

*****   101*****

Scan of resident job queue has ended

Job number     4 finishes execution at  run_system$.Time= 7.00000000e+00;

MDS begins scan of resident job queue at          run_system$.Time=
 7.00000000e+00;

*****   111*****

*****    61*****

*****    71*****

*****    81*****

*****   101*****

Scan of resident job queue has ended

Job number    12 arrives in system at   run_system$.Time= 8.52667105e+00;Lookup: input value 17.1587989 less than lowestb
ce entry.
Lookup: input value 22.1548817 less than lowest table entry.
Lookup: input value 14.8746043 less than lowest table entry.


it needs     1 7-track tapes,     2 9-track tapes, and     1 disks
region =   198
sri =     3
run time =  3.98
MDS begins scan of resident job queue at          run_system$.Time=
 8.52667105e+00;

*****   111*****
```

```
*****   121*****
*****   111*****
*****   121*****
*****    61*****
*****    71*****
*****    81*****
*****   101*****
Scan of resident job queue has ended
Job number    10 begins executing at    run_system$.Time= 8.99783278e+00;
Job number    11 begins executing at    run_system$.Time= 9.23002315e+00;
Job number    11 finishes execution at  run_system$.Time= 9.39023125e+00;
MDS begins scan of resident job queue at          run_system$.Time=
 9.39023125e+00;
*****   121*****
*****    61*****
*****    71*****
*****    81*****
*****   101*****
Scan of resident job queue has ended
Job number     7 finishes execution at  run_system$.Time= 9.50000000e+00;
MDS begins scan of resident job queue at          run_system$.Time=
 9.50000000e+00;
*****   121*****
*****    61*****
*****    81*****
*****   101*****
Scan of resident job queue has ended
Job number     6 finishes execution at  run_system$.Time= 9.69999993e+00;
MDS begins scan of resident job queue at          run_system$.Time=
 9.69999993e+00;
*****   121*****
*****    81*****
*****   101*****
Scan of resident job queue has ended
```

```
MDS begins scan of resident job queue at          run_system$.Time=
 1.03629055e+01;

*****   121*****

*****    81*****

Scan of resident job queue has ended

Job number     8 finishes execution at  run_system$.Time= 1.10000000e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 1.10000000e+01;

*****   121*****

Scan of resident job queue has ended

Job number    12 begins executing at    run_system$.Time= 1.15266711e+01;

Job number    13 arrives in system at   run_system$.Time= 1.24111865e+01;Lookup: input value 51.2573738 less than lowestb
¢ce entry.


it needs     0 7-track tapes,     1 9-track tapes, and     0 disks
region =    88
sri =     1
run time =  0.53

MDS begins scan of resident job queue at          run_system$.Time=
 1.24111865e+01;

*****   121*****

*****   130*****

*****   131*****

*****   121*****

*****   131*****

Scan of resident job queue has ended

Job number    14 arrives in system at   run_system$.Time= 1.30800546e+01;Lookup: input value 67.0662470 less than lowestb
¢ce entry.


it needs     0 7-track tapes,     0 9-track tapes, and     1 disks
region =   147
sri =     3
run time =  0.44

MDS begins scan of resident job queue at          run_system$.Time=
 1.30800546e+01;

*****   121*****

*****   140*****

*****   141*****

*****   121*****
```

***** 131*****

Scan of resident job queue has ended

Sample of resident job queue at run_system$.Time= 1.50000000e+01;

Job	Queue	Entry time	Class	Total dev. for job	9-tracks still needed for the job	7-tracks still needed for the job	disks still needed for the job	Status
12	4	8.53	A	4	0	0	0	3
14	4	13.08	B	1	0	0	0	2
13	2	12.41	B	1	0	0	0	2

```
Job number     13 begins executing at     run_system$.Time= 1.54111865e+01;

Job number     12 finishes execution at   run_system$.Time= 1.55051504e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 1.55051504e+01;

*****    141*****

*****    131*****

Scan of resident job queue has ended

Job number     13 finishes execution at   run_system$.Time= 1.59435935e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 1.59435935e+01;

*****    141*****

Scan of resident job queue has ended

Job number     14 begins executing at     run_system$.Time= 1.60800545e+01;
```

```
Job number    14 finishes execution at   run_system$.Time= 1.65187709e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 1.65187709e+01;

There are no more jobs in the system

Scan of resident job queue has ended


Job number    15 arrives in system at   run_system$.Time= 2.48153071e+01;Lookup: input value 4.61658973 less than lowest
¢ce entry.

Lookup: input value 6.86185980 less than lowest table entry.


it needs     1 7-track tapes,     0 9-track tapes, and     0 disks
region =   216
sri =      3
run time =  4.76

MDS begins scan of resident job queue at          run_system$.Time=
 2.48153071e+01;

*****   150*****

*****   151*****

*****   151*****

Scan of resident job queue has ended


Job number    15 begins executing at    run_system$.Time= 2.78153071e+01;




Sample of resident job queue at         run_system$.Time= 3.00000000e+01;


 Job    Queue    Entry   Class   Total dev.  9-tracks  7-tracks  disks  Status
                  time            for job    still needed for the job


   15     4     24.82     B        1            0         0        0
```

```
Job number    16 arrives in system at   run_system$.Time= 3.04200656e+01;Lookup: input value 13.9649853 less than lowestb
¢ce entry.
Lookup: input value 65.4852848 less than lowest table entry.
Lookup: input value 21.9491677 less than lowest table entry.


it needs     1 7-track tapes,     0 9-track tapes, and     0 disks
region =   101
sri =      3
run time =  0.24

MDS begins scan of resident job queue at          run_system$.Time=
 3.04200656e+01;

*****   151*****

*****   160*****

*****   161*****

*****   151*****

*****   161*****

Scan of resident job queue has ended

Job number    15 finishes execution at  run_system$.Time= 3.25720000e+01;
```

```
MDS begins scan of resident job queue at          run_system$.Time=
 3.25720000e+01;

*****   161*****

Scan of resident job queue has ended

Job number    16 begins executing at   run_system$.Time= 3.34200654e+01;

Job number    16 finishes execution at  run_system$.Time= 3.36615787e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 3.36615787e+01;

There are no more jobs in the system

Scan of resident job queue has ended

Job number    17 arrives in system at   run_system$.Time= 3.51232257e+01;Lookup: input value 8.85159326 less than lowestb
¢ce entry.


it needs     1 7-track tapes,     0 9-track tapes, and     1 disks
region =   129
sri =     3
run time =  1.44

MDS begins scan of resident job queue at          run_system$.Time=
 3.51232257e+01;

*****   170*****

*****   171*****

*****   171*****

Scan of resident job queue has ended

Job number    18 arrives in system at   run_system$.Time= 3.52415566e+01;Lookup: input value 29.8323674 less than lowestb
¢ce entry.
Lookup: input value 44.3717661 less than lowest table entry.


it needs     0 7-track tapes,     0 9-track tapes, and     1 disks
region =   156
sri =     3
run time =  1.65

MDS begins scan of resident job queue at          run_system$.Time=
 3.52415566e+01;

*****   171*****

*****   180*****

*****   181*****

*****   171*****

*****   181*****

Scan of resident job queue has ended

Job number    19 arrives in system at   run_system$.Time= 3.56532979e+01;Lookup: input value 17.4722843 less than lowestb
¢ce entry.
```

```
it needs     0 7-track tapes,     1 9-track tapes, and     0 disks
region =    77
sri =     1
run time =  2.47

MDS begins scan of resident job queue at          run_system$.Time=
 3.56532979e+01;

*****   171*****

*****   181*****

*****   190*****

*****   191*****

*****   171*****

*****   181*****

*****   191*****

Scan of resident job queue has ended

Job number    20 arrives in system at   run_system$.Time= 3.75488963e+01;Lookup: input value 3.96934122 less than lowest¬
¢ce entry.
Lookup: input value 5.22289783 less than lowest table entry.
Lookup: input value 6.20009303 less than lowest table entry.


it needs     1 7-track tapes,     0 9-track tapes, and     0 disks
region =   237
sri =     0
run time =  0.14

MDS begins scan of resident job queue at          run_system$.Time=
 3.75488963e+01;

*****   171*****

*****   181*****

*****   191*****

*****   200*****

*****   201*****

*****   171*****

*****   181*****

*****   191*****

*****   201*****

Scan of resident job queue has ended

Job number    17 begins executing at    run_system$.Time= 3.81232257e+01;

Job number    18 begins executing at    run_system$.Time= 3.82415566e+01;

Job number    19 begins executing at    run_system$.Time= 3.86532979e+01;
```

```
it needs     0 7-track tapes,     0 9-track tapes, and     2 disks
region =   235
sri =     3
run time =  2.21

MDS begins scan of resident job queue at          run_system$.Time=
 3.88508530e+01;

*****   171*****

*****   181*****

*****   210*****

*****   211*****

*****   171*****

*****   181*****

*****   211*****

*****   191*****

*****   201*****

Scan of resident job queue has ended

Job number    22 arrives in system at   run_system$.Time= 3.95220966e+01;Lookup: input value 20.5038729 less than lowestb
¢ce entry.

it needs     0 7-track tapes,     2 9-track tapes, and     2 disks
region =   119
sri =     5
run time =  0.39

MDS begins scan of resident job queue at          run_system$.Time=
 3.95220966e+01;

*****   220*****

*****   221*****

*****   221*****

*****   171*****

*****   181*****

*****   211*****

*****   191*****

*****   201*****

Scan of resident job queue has ended

Job number    17 finishes execution at  run_system$.Time= 3.95616188e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 3.95616188e+01;

*****   221*****
```

```
*****   181*****

*****   211*****

*****   191*****

*****   201*****

Scan of resident job queue has ended

Job number    23 arrives in system at   run_system$.Time= 3.95941110e+01;Lookup: input value 28.6126192 less than lowest e
¢ce entry.


it needs     0 7-track tapes,     1 9-track tapes, and     0 disks
region =   125
sri =     3
run time =  2.93

MDS begins scan of resident job queue at          run_system$.Time=
 3.95941110e+01;

*****   221*****

*****   181*****

*****   211*****

*****   230*****

*****   231*****
*****   221*****

*****   181*****

*****   211*****

*****   231*****

*****   191*****

*****   201*****

Scan of resident job queue has ended

Job number    18 finishes execution at  run_system$.Time= 3.98913217e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 3.98913217e+01;

*****   221*****

*****   211*****

*****   231*****

*****   191*****

*****   201*****

Scan of resident job queue has ended

Job number    20 begins executing at    run_system$.Time= 4.05488963e+01;
```

```
MDS begins scan of resident job queue at          run_system$.Time=
 4.06858358e+01;

*****   221*****

*****   211*****

*****   231*****

*****   191*****

Scan of resident job queue has ended

Job number    19 finishes execution at  run_system$.Time= 4.11280394e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 4.11280394e+01;

*****   221*****

*****   211*****

*****   231*****

Scan of resident job queue has ended
Job number    21 begins executing at    run_system$.Time= 4.18508530e+01;

Job number    24 arrives in system at   run_system$.Time= 4.20392323e+01;Lookup: input value 18.5419455 less than lowestl
¢ce entry.
Lookup: input value 30.3989928 less than lowest table entry.

it needs     0 7-track tapes,     0 9-track tapes, and     1 disks
region =   241
sri =     3
run time =  0.70

MDS begins scan of resident job queue at          run_system$.Time=
 4.20392323e+01;

*****   221*****

*****   211*****

*****   231*****

*****   240*****

*****   241*****

*****   221*****

*****   211*****

*****   231*****

*****   241*****

Scan of resident job queue has ended

Job number    22 begins executing at    run_system$.Time= 4.25220966e+01;

Job number    23 begins executing at    run_syste m$.Time= 4.25941110e+01;

Job number    25 arrives in system at   run_system$.Time= 4.26047077e+01;Lookup: input value 17.3569872 less than lowest
```

```
Lookup: input value 43.7173758 less than lowest table entry.


it needs     0 7-track tapes,     1 9-track tapes, and     0 disks
region =   182
sri =     3
run time =  2.76

MDS begins scan of resident job queue at          run_system$.Time=
 4.26047077e+01;

*****   221*****

*****   211*****

*****   231*****

*****   241*****

*****   250*****

*****   251*****

*****   221*****

*****   211*****

*****   231*****

*****   241*****

*****   251*****

Scan of resident job queue has ended

Job number    26 arrives in system at   run_system$.Time= 4.26094518e+01;Lookup: input value 5.07961422 less than lowes
¢ce entry.
Lookup: input value 35.7184758 less than lowest table entry.


it needs     0 7-track tapes,     0 9-track tapes, and     1 disks
region =   186
sri =     0
run time =  2.06

MDS begins scan of resident job queue at          run_system$.Time=
 4.26094518e+01;

*****   221*****

*****   211*****

*****   231*****

*****   241*****

*****   251*****

*****   260*****

*****   261*****

*****   221*****
```

```
*****   231*****

*****   241*****

*****   251*****

*****   261*****

Scan of resident job queue has ended

Job number    22 finishes execution at  run_system$.Time= 4.29128170e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 4.29128170e+01;

*****   211*****

*****   231*****

*****   241*****

*****   251*****
*****   261*****

Scan of resident job queue has ended

Job number    21 finishes execution at  run_system$.Time= 4.40586534e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 4.40586534e+01;

*****   231*****

*****   241*****

*****   251*****

*****   261*****

Scan of resident job queue has ended
```

Sample of resident job queue at run_system$.Time= 4.50000000e+01;

Job	Queue	Entry time	Class	Total dev. for job	9-tracks still needed for the job	7-tracks still needed for the job	disks still needed for the job	Status
23	4	39.59	B	1	0	0	0	3
24	4	42.04	A	1	0	0	0	2
25	4	42.60	A	1	0	0	0	2
26	1	42.61	B	1	0	0	0	2

```
Job number    27 arrives in system at   run_system$.Time= 4.53637309e+01;Lookup: input value 9.33293653 less than lowest
¢ce entry.


it needs     0 7-track tapes,     1 9-track tapes, and     1 disks
region =   157
sri =     5
run time =  0.04

MDS begins scan of resident job queue at          run_system$.Time=
 4.53637309e+01;

*****   270*****

*****   271*****

*****   271*****

*****   231*****

*****   241*****

*****   251*****

*****   261*****

Scan of resident job queue has ended

Job number    23 finishes execution at  run_system$.Time= 4.55267320e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 4.55267320e+01;

*****   271*****

*****   241*****

*****   251*****

*****   261*****

Scan of resident job queue has ended

Job number    25 begins executing at    run_system$.Time= 4.56047077e+01;

Job number    26 begins executing at    run_system$.Time= 4.56094518e+01;

Job number    28 arrives in system at   run_system$.Time= 4.56457992e+01;

it needs     0 7-track tapes,     1 9-track tapes, and     1 disks
region =   191
sri =     3
run time =  3.82

MDS begins scan of resident job queue at          run_system$.Time=
 4.56457992e+01;

*****   271*****
*****   241*****

*****   251*****

*****   280*****

*****   281*****
```

```
*****   271*****

*****   241*****

*****   251*****

*****   281*****

*****   261*****

Scan of resident job queue has ended

Job number     24 finishes execution at  run_system$.Time= 4.57377324e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 4.57377324e+01;

*****   271*****

*****   251*****

*****   281*****

*****   261*****

Scan of resident job queue has ended

Job number     26 finishes execution at  run_system$.Time= 4.76725202e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 4.76725202e+01;

*****   271*****

*****   251*****

*****   281*****

Scan of resident job queue has ended

Job number     29 arrives in system at    run_system$.Time= 4.79412642e+01;Lookup: input value 63.9178343 less than lowestb
¢ce entry.


it needs      0 7-track tapes,      0 9-track tapes, and      1 disks
region =     92
sri =      5
run time =  1.89

MDS begins scan of resident job queue at          run_system$.Time=
 4.79412642e+01;

*****   271*****

*****   290*****

*****   291*****

*****   271*****

*****   291*****

*****   251*****

*****   281*****
```

```
Scan of resident job queue has ended

Job number     27 begins executing at     run_system$.Time= 4.83637309e+01;

Job number     25 finishes execution at   run_system$.Time= 4.83647919e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 4.83647919e+01;

*****    271*****

*****    291*****

*****    281*****

Scan of resident job queue has ended

Job number     27 finishes execution at   run_system$.Time= 4.84023070e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 4.84023070e+01;

*****    291*****

*****    281*****

Scan of resident job queue has ended

Job number     28 begins executing at     run_system$.Time= 4.86457992e+01;
```

```
Job number     30 arrives in system at    run_system$.Time= 5.06404967e+01;Lookup: input value 30.5395389 less than lowestb
¢ce entry.
Lookup: input value 24.0839105 less than lowest table entry.


it needs      0 7-track tapes,      0 9-track tapes, and      1 disks
region =    275
sri =      3
run time =   6.93

MDS begins scan of resident job queue at          run_system$.Time=
 5.06404967e+01;

*****    291*****

*****    281*****

*****    300*****

*****    301*****

*****    291*****

*****    281*****

*****    301*****

Scan of resident job queue has ended

Job number     29 begins executing at     run_system$.Time= 5.09412642e+01;

Job number     28 finishes execution at   run_system$.Time= 5.24680519e+01;
```

```
 5.24680519e+01;

*****   291*****

*****   301*****

Scan of resident job queue has ended

Job number     31 arrives in system at    run_system$.Time= 5.28295832e+01;

it needs      0 7-track tapes,      1 9-track tapes, and      1 disks
region =    108
sri =      3
run time =  1.75

MDS begins scan of resident job queue at          run_system$.Time=
 5.28295832e+01;

*****   291*****

*****   301*****

*****   310*****

*****   311*****

*****   291*****

*****   301*****

*****   311*****

Scan of resident job queue has ended

Job number     29 finishes execution at  run_system$.Time= 5.28341842e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 5.28341842e+01;

*****   301*****

*****   311*****

Scan of resident job queue has ended

Job number     30 begins executing at     run_system$.Time= 5.36404967e+01;

Job number     31 begins executing at     run_system$.Time= 5.58295832e+01;

Job number     32 arrives in system at    run_system$.Time= 5.70297236e+01;Lookup: input value .354917649 less than lowestb
ce entry.
Lookup: input value 13.2798510 less than lowest table entry.
Lookup: input value 47.6000462 less than lowest table entry.


it needs      0 7-track tapes,      0 9-track tapes, and      1 disks
region =      0
sri =      3
run time =  1.06

MDS begins scan of resident job queue at          run_system$.Time=
 5.70297236e+01;

*****   301*****
```

***** 320*****

***** 321*****

***** 301*****

***** 311*****

***** 321*****

Scan of resident job queue has ended

Job number	31 finishes time	executi	atus for job	still needed for the job			
30	4	50.64	A	1	0	0	0
32	4	57.03	A	1	0	0	0
33	4	58.10	B	1	0	0	0

3

2

2

```
Job number    32 begins executing at    run_system$.Time= 6.00297236e+01;

Job number    30 finishes execution at  run_system$.Time= 6.05735784e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 6.05735784e+01;

*****   321*****

*****   331*****

Scan of resident job queue has ended

Job number    32 finishes execution at  run_system$.Time= 6.10866523e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 6.10866523e+01;

*****   331*****

Scan of resident job queue has ended

Job number    33 begins executing at    run_system$.Time= 6.11047287e+01;

Job number    33 finishes execution at  run_system$.Time= 6.21447229e+01;

MDS begins scan of resident job queue at          run_system$.Time=
 6.21447229e+01;

There are no more jobs in the system

Scan of resident job queue has ended

Job number    34 arrives in system at   run_system$.Time= 6.33201351e+01;Lookup: input value .210024776 less than lowest¤
¢ce entry.


it needs     0 7-track tapes,     2 9-track tapes, and     1 disks
region =   143
sri =     3
run time =  0.39

MDS begins scan of resident job queue at          run_system$.Time=
 6.33201351e+01;

*****   340*****

*****   341*****

*****   341*****

Scan of resident job queue has ended
```

Sample of resident job queue at run_system$.Time= 6.33201351e+01;

Job	Queue	Entry time	Class	Total dev. for job	9-tracks still needed for the job	7-tracks	disks	Status
34	4	63.32	A	3	0	0	0	

<id>Set name	NUMBER IN QUEUE: Total	Current	Maximum	Average	No-wait	Time empty	WAITING TIME: Maximum	Average	Ave non-zero
<1>wait_q	4.	0.	3.	.11	0.	95.26%	3.00	1.81	1.81
<1>resqueue(16)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(15)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(14)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(13)	1.	0.	1.	.09	0.	91.31%	5.50	5.50	5.50
<1>resqueue(12)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(11)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(10)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(9)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(8)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(7)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(6)	5.	0.	2.	.33	0.	74.95%	5.00	4.16	4.16
<1>resqueue(5)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(4)	19.	1.	4.	1.51	0.	17.27%	9.93	5.30	5.30
<1>resqueue(3)	0.	0.	0.	0.00	0.	100.00%	0.00	Undefined	
<1>resqueue(2)	7.	0.	5.	.78	0.	68.40%	11.00	7.08	7.08
<1>resqueue(1)	2.	0.	1.	.13	0.	87.05%	5.06	4.10	4.10

FIXED AND FLOAT STATISTICS

<id>Variable name	Tot. Assigns	Maximum	Minimum	Current	Time ave	Zero value time
<1>disk(10,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(10,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(10,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(9,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(9,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(9,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(8,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(8,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(8,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(7,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(7,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(7,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(6,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(6,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(6,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(5,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(5,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(5,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(4,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(4,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(4,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(3,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(3,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(3,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(2,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(2,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(2,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(1,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(1,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>disk(1,1)	0.	0.00	0.00	0.00	0.00	100.00%

<1>tape7(2,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>tape7(2,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>tape7(1,3)	0.	0.00	0.00	0.00	0.00	100.00%
<1>tape7(1,2)	0.	0.00	0.00	0.00	0.00	100.00%
<1>tape7(1,1)	0.	0.00	0.00	0.00	0.00	100.00%
<1>tape9(7,3)	5.	4.00	0.00	0.00	.57	82.63%
<1>tape9(7,2)	2.	2.00	0.00	0.00	.35	82.63%
<1>tape9(7,1)	3.	3.00	0.00	0.00	.43	82.63%
<1>tape9(6,3)	5.	4.00	0.00	0.00	.57	82.63%
<1>tape9(6,2)	2.	2.00	0.00	0.00	.35	82.63%
<1>tape9(6,1)	3.	3.00	0.00	0.00	.43	82.63%
<1>tape9(5,3)	3.	3.00	0.00	0.00	.39	85.00%
<1>tape9(5,2)	2.	2.00	0.00	0.00	.30	85.00%
<1>tape9(5,1)	3.	3.00	0.00	0.00	.36	85.00%
<1>tape9(4,3)	7.	3.00	0.00	0.00	.56	75.63%
<1>tape9(4,2)	4.	4.00	0.00	0.00	.67	75.63%
<1>tape9(4,1)	7.	3.00	0.00	0.00	.60	75.63%
<1>tape9(3,3)	11.	3.00	0.00	0.00	.44	61.80%
<1>tape9(3,2)	8.	6.00	0.00	0.00	1.64	61.80%
<1>tape9(3,1)	14.	3.00	0.00	0.00	1.00	61.80%
<1>tape9(2,3)	10.	2.00	0.00	1.00	.31	70.14%
<1>tape9(2,2)	9.	13.00	0.00	4.00	2.18	70.14%
<1>tape9(2,1)	16.	3.00	0.00	2.00	.75	70.14%
<1>tape9(1,3)	19.	5.00	0.00	1.00	.84	53.60%
<1>tape9(1,2)	13.	13.00	0.00	4.00	2.22	53.60%
<1>tape9(1,1)	24.	3.00	0.00	2.00	1.16	53.60%

r 1815 157.809 2040+9419