A Simulator Program for TSS

Michigan Univ Ann Arbor

Dec 75
**BEST SELLERS**

**FROM NATIONAL TECHNICAL INFORMATION SERVICE**

- Product Liability Insurance: Assessment of Related Problems and Issues, Staff Study
  - PB-252 204/PAT 181 p PCS$9.50/MFS$3.00
- Evaluation of Home Solar Heating System
  - UCRL-51 711/PAT 154 p PCS$6.75/MFS$3.00
- Developing Noise Exposure Contours for General Aviation Airports
  - ADA-023 429/PAT 205 p PCS$7.75/MFS$3.00
- Cooling Tower Environment, 1974. Proceedings of a Symposium Held at the University of Maryland Adult Education Center on Mar. 4-6, 1974
  - CONF-74 0302/PAT 646 p PCS$13.80/MFS$3.00
- Biological Services Program, Fiscal Year 1975
  - PB-251 738/PAT 52 p PCS$4.50/MFS$2.00
- An Atlas of Radiation Histopathology
  - TID-26-676/PAT 234 p PCS$7.60/MFS$2.00
- Federal Funding of Civilian Research and Development, Vol. 1. Summary
  - PB-251 266/PAT 61 p PCS$5.00/MFS$3.00
- Federal Funding of Civilian Research and Development, Vol. 2. Case Studies
  - PB-251 683/PAT 336 p PCS$10.00/MFS$3.00
- Handbook on Aerosols
  - TID-26-606/PAT 141 p PCS$9.00/MFS$3.00

---

**HOW TO ORDER**

When you indicate the method of payment, please note if a purchase order is not accompanied by payment, you will be billed an additional $5.00 ship and handling charge. Please include the card expiration date when using American Express.

Normal delivery time takes three to five weeks. It is vital that you order by number or your order will be manually filled, insuring a delay. You can opt for air mail delivery for a $2.00 charge per item. Just check the Air Mail Service box. If you're really pressed for time, call the NTIS Rush Order Service (703) 557-4650. For a $10.00 charge per item, your order will be air mail delivered within 48 hours. Or, you can pick up your order in the Washington Information Center & Bookstore or at our Springfield Operations Center within 24 hours for a $6.00 per item charge.

You may also place your order by telephone or TELEX. The order desk number is (703) 557-4650 and the TELEX number is 89405.

Whenever a foreign sales price is not specified in the listings, all foreign buyers must add the following charges to each order: $2.50 for each paper copy, $1.50 for each microfiche; and $10.00 for each published report.

Thank you for your interest in NTIS. We appreciate your order.

---

**METHOD OF PAYMENT**

- [ ] Charge my NTIS deposit account no. ____________________________
- [ ] Purchase order no. ____________________________
- [ ] Check enclosed for $__________________________
- [ ] Charge to my American Express Card account number ____________________________
- [ ] Airmail Service requested

Card expiration date ____________________________
Signature ____________________________

**NTIS**

National Technical Information Service
U.S. DEPARTMENT OF COMMERCE
Springfield, Va. 22161
(703) 557-4650 TELEX 89405

**TOTAL**

Quantity

<table>
<thead>
<tr>
<th>Item Number</th>
<th>Paper Copy (PC)</th>
<th>Microfiche (MF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All Prices Subject to Change

11/76

Additional Charge

Enter Grand Total

---

**NAME**

<table>
<thead>
<tr>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITY</td>
</tr>
</tbody>
</table>

**Renew your subscription today!**

NTIS offers a wide range of subscriptions for individuals and organizations. To receive the latest information on the latest publications, please sign up for our subscription service. Visit our website at [NTIS Website] to learn more about our subscription options.

---

[NTIS Website]
A SIMULATOR PROGRAM FOR TSS

The University of Michigan

Approved for public release; distribution unlimited.
This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

This report has been reviewed and approved for publication.

APPROVED:  
ROCCO P. IUORNO  
Project Engineer

APPROVED:  
ROBERT D. KRUTZ, Col., USAF  
Chief, Information Sciences Division

FOR THE COMMANDER:  
JOHN P. HUSS  
Acting Chief, Plans Office

Do not return this copy. Retain or destroy.
This report describes the computer program and its implementation which simulates the Honeywell Time Sharing System (Version 6) operating on the Honeywell 635. The program (mod-1) simulates the behavior of Subsystem programs and the efforts of the scheduling policies of the Time Sharing System. The simulation program is written in Simscript II programming language.
# Table of Contents

1. Introduction ........................................ 1
2. Event and Routine Summary .......................... 2
   2.1 Initialization .................................. 2
   2.2 Diagnostic and Output .......................... 2
   2.3 Allocator Routines .............................. 3
   2.4 Derail Routines ................................ 4
   2.5 Courtesy Calls .................................. 4
   2.6 Line Service .................................... 4
3. Data Structures ..................................... 4
   3.1 The UST Entity .................................. 5
   3.2 The Processor Queue - CPUQ .................... 7
   3.3 The Memory Queue - MEMQ ...................... 7
   3.4 The Memory Map ................................ 7
   3.5 The Disk I/O Queue - DIOQ .................... 9
   3.6 The Courtesy Call Queue - CCQ ............... 9
4. Input Parameters .................................... 10
   4.1 Diagnostic and Control ......................... 10
   4.2 TSS Parameters ................................ 11
   4.3 Driving Data ................................... 12
5. Output .............................................. 15
   5.1 Input Echo .................................... 15
   5.2 Memory Statistics .............................. 15
   5.3 Overall Mean Rates ............................ 15
   5.4 Typical Subsystem Behavior ................... 15
5.5 System Queue Data
5.6 Diagnostic Output

6. Using the Program
6.1 Compilation
6.2 Execution

7. Detailed Program Description
7.1 PREAMBLE
7.2 INIT
7.3 MAIN
7.4 PDIA
7.5 REINIT
7.6 CORSAMP
7.7 MQPRINT
7.8 CQPRINT
7.9 STERM
7.10 OUTPUT
7.11 SNAP.R
7.12 ALLOCI
7.13 MAP
7.14 SDP
7.15 SDP3
7.16 SDP4
7.17 SDP5
7.18 SDP6
7.19 SDP7
7.20 SPMACT
8. Error Messages
Figures

1. Memory Map Configuration 8
2. Sample Input Data Set 14
3. Sample Simulation Output 18

Appendices

1. Variable Description
2. Flow Charts
   - Program Listing
1. Introduction

The purpose of this report is to describe a program written to simulate the Honeywell Time Sharing System (Version 8) operating on the Honeywell 635 computer.

The model includes, in great detail, sections of the TSS Allocator as well as the TSS Executive, the Deral Processor and Line Service. The program simulates the behavior of Subsystem programs and the effects of the scheduling policies of TSS.

The simulation program being described here is written in the Simscript II programming language. Simscript is a high level, event oriented simulation language with several features that are very helpful for this particular use.

The program was written and developed at the Systems Engineering Laboratory at the University of Michigan and was debugged and validated on a Honeywell 3000 system at the Rome Air Development Center, Rome, New York.
2. Event and Routine Summary

As mentioned before, Sinscript is an event oriented language and hence the program is made up of several small events and routines, each corresponding to a subroutine or block of code of TSS. To ease understanding and debugging the names of all routines and events have been preserved from the actual system program listings as far as possible.1

The events and routines in the simulation program can be classified into six categories. The following is a list of the categories and the routines comprising each one with a brief description of their functions.

2.1 Initialization

INIT - Reads input parameters and values. Initializes data structures. Schedules events for termination and diagnostic printing.

MAIN - Starts simulation.

2.2 Diagnostic and Output

PDIAG - Controls printing of diagnostic information as per input parameters.

REINIT - Reinitializes statistic counters after given simulation period.

CORSAMP - Samples values of program size, hole size and used core for collecting statistics.

MQPRINT - Prints memory queue.

CQPRINT - Prints processor queue.

STERM - Terminates simulation.

1 In certain cases where two names have the same first four characters, the names have been transformed due to a Simscript requirement that the first four characters of all names be unique. e.g. ALLCC1 and ALLCC2 are renamed as 1ALLCC and 2ALLCC.
OUTPUT - Prints simulation results and statistics.

SWAP.R - Prints debugging and diagnostic information in case the simulation terminates abnormally.

2.3 Allocator Routines

ALLOC - Main entry point to the TSS allocator and the process allocator process (PAP).

MAP - Memory allocator process. Selects programs for memory allocation and swap.

SDP, SDP3, SDP4, SDP5, SDP7 - Swap decision processors. Find programs for regular and force swaps, handle urgent user logic, core fence for urgent user and TSS swap area size control.

SPMACT - Special memory action processor for changing TSS swap area size.

MBA, MBA3 - Memory buffer allocator processes. Allocate free allocation for programs and manages free map.

MBD - Memory buffer deallocator. Returns blocks of memory to available storage pool in memory map.

MVT - Memory map verification process. Also accumulates core statistics.

SMOUT - Swap out routine.

SWIN - Swap in routine.

SWPLD - Schedules courtesy call for swap.

SSFINI - Terminates subsystem execution.

KINSMP - Adjusts subsystem status for keyboard input and output operations.

KIOSRT - Schedules keyboard I/O courtesy calls.

START - Introduces new subsystems into the system and initializes.
subsystem attributes.

SACT - Takes subsystem accounting for statistics.
AITCHG - Changes subsystem state times.
EXENTR - Entry to allocator via an interrupt.
EXEACT - Accounting routine after return from subsystem dispatch.

2.4 Derail Routines

KONDRL - Keyboard input derail.
KOTDRL - Keyboard output derail.
DRLDIO - Disk I/O derail.
DRLRET - Subsystem return (termination) derail.

2.5 Courtesy Calls

KIUCC - Keyboard I/O courtesy call.
DI0CC - Disk I/O courtesy call.
1ALLCC - Swap out courtesy call.
2ALLCC - Swap in courtesy call.

2.6 Line Service

LINSV - Controls TSS action during relinquish to GCOS and TSS idle.
Also schedules new program arrivals into TSS.

3. Data Structures

Three major TSS data structures are modelled in the program exactly as they appear in the real system. These are the processor and memory queues and the memory map. In addition, the model also has a disk I/O queue and a queue of scheduled courtesy calls. The purpose of these is discussed below. Finally, a program in TSS is modelled by means of an entity called a UST. (The name UST was chosen because the entity closely resembles an entry in the User Status Table.)
3.1 The UST Entity

Each active program in the system is represented by a temporary entity called a UST. An entity is created when a new job arrives and is destroyed when it terminates.

The UST entity has a number of attributes some of which are similar to those found in the TSS user status table and others which control the activity of the program.

The following is a list of the attributes of the UST entity. The attributes on the left are those that also appear in the TSS user status table and have the same names as the original variables. (Note: The digits 1 and 2 appended to a variable name signify the upper and lower halves of that word respectively.) The attributes on the right are used to control the simulation of the program that the UST represents. A description of each attribute is given in Appendix 1.

**UST Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FK19</td>
<td>CHKCPU</td>
</tr>
<tr>
<td>FL18</td>
<td>JOBNO</td>
</tr>
<tr>
<td>FL19</td>
<td>KILL</td>
</tr>
<tr>
<td>FL21</td>
<td>NXTDIO</td>
</tr>
<tr>
<td>FL22</td>
<td>NXTKIN</td>
</tr>
<tr>
<td>FL23</td>
<td>NXTKOUT</td>
</tr>
<tr>
<td>FL24</td>
<td>OUTCC</td>
</tr>
<tr>
<td>FL34</td>
<td>DIOIAT</td>
</tr>
<tr>
<td>LSIZE</td>
<td></td>
</tr>
<tr>
<td>LSPTS</td>
<td>K11AT</td>
</tr>
</tbody>
</table>
The UST entity contains the entire description of a program in the system. It is filed in the different system queues in exactly the same manner as the entries in the TSS user status table are linked into queues.

Usually, attributes of entities are referred to in Simscript by the expression:

attribute (entity pointer)

to denote the particular entity whose attribute is being examined. However, the entity pointer can be omitted and the default global entity name 'UST' is assumed. Throughout most of the program such a scheme is used to refer to attributes since the global variable 'UST' always contains the pointer to the UST on which the allocator is presently working.
3.2 The Processor Queue - CPUQ

The TSS processor queue consists of a linked list of user status table entries. In the program, this queue is modelled as a simple ordered Simscript set called CPUQ. UST entities are filed last in the set and the first entity is always picked for allocation.

3.3 The Memory Queue - MEMQ

The TSS memory queue is a linked list of UST entries ordered by increasing program sizes. This is modelled in the program as a set MEMQ whose members (UST entities) are ranked by low LSIZE attributes. When new UST's are filed in MEMQ, Simscript automatically links them so that the increasing order of the LSIZE attribute is preserved.

3.4 The Memory Map

The memory map data structure is almost identical to its TSS counterpart. The map consists of a doubly linked list of blocks each of which represents one program present in core.

The memory map consists of five vectors: SJOB, SHOLE, SUCC, PRED and IDPTR. Each entry in the map is represented by a corresponding element from each of the above vectors. The vectors have the following functions:

- SJOB(I) - contains the program size of the program (in units of 1024 words).
- SHOLE(I) - contains the size of the "hole" (unused portion of core) following the program in core (in units of 1024 words).
- SUCC(I) - pointer to the entry in the map succeeding this entry.
- PRED(I) - pointer to the entry in the map preceding this entry.
- IDPTR(I) - contains the identification number attribute (JOBNO) of
the UST entity that corresponds to this entry in the memory map.

The first entry in the memory map is always a dummy entry which belongs to no program but is kept merely to hold the hole size at the top of TSS core. The variables HEAD and TAIL point to the first and last entries in the map respectively. A possible memory map configuration is shown in Figure 1 below.

<table>
<thead>
<tr>
<th>I</th>
<th>SJOB</th>
<th>SHOLE</th>
<th>SUC</th>
<th>PRED</th>
<th>IDPTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>56</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>15</td>
<td>0</td>
<td>7</td>
<td>67</td>
</tr>
<tr>
<td>7</td>
<td>22</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>24</td>
</tr>
</tbody>
</table>

Figure 1. Memory Map Configuration

In the example of Figure 1 the values of HEAD and TAIL would be 1 and 6 respectively.
3.5 The Disk I/O Queue - DIOQ

Since no data is obtained on the service time of the disk units the program uses an artificial disk I/O queue to simulate the handling of disk I/O operations.

The disk I/O queue is a fifo queue and is modelled with a single exponential server. The formula for obtaining the exponential server rate from the measured values of the disk I/O queue and disk I/O operation arrival rate is described in Reference 1.

3.6 The Courtesy Call Queue - CCQ

The courtesy call queue is strictly for the internal use of the model and does not exist in the real system. The purpose of this queue is to accurately simulate the arrival of scheduled courtesy calls when they interrupt the execution of a subsystem program.

In the actual system a courtesy call would generate an interrupt to TSS. In the model, this interrupt is generated with the help of the CCQ.

The CCQ consists purely of a list of all future courtesy calls ordered by increasing arrival times. When a new courtesy call is scheduled it is entered in the queue at the appropriate position depending on its arrival time. When a courtesy call arrives, it is removed from the CCQ. Note that at this time it should be at the head of CCQ and the simulation time will be equal to its arrival time.

When the TSS allocator dispatches the CPU to a subsystem, it checks to see if a courtesy call would be the next event to interrupt the subsystem. This is done by checking the arrival time of the first entry in the CCQ. If that is the case, an interrupt and consequent
entry into the allocator is scheduled immediately following the
courtesy call.

4. Input Parameters

The input to the simulation program is divided into three categories
as follows. For each category the input parameters are given in the
order in which they must appear in the input stream. The designations
(R) and (I) specify the mode of the variable as real or integer
respectively. All time values are in milliseconds and all memory values
are in units of 1024 words unless otherwise stated.

4.1 Diagnostic and Control

These input values control the execution of the simulation program
and the printing of diagnostic comments. The form of messages printed
is described fully in Section 5.6.

STOPTIME - (R) Value of simulated time at which the simulation is
to be terminated.

DEVS - (I) Event diagnostics switch. If this switch is set to 1,
a message is printed. If no diagnostics are required this switch must
be 0.

DRS - (I) This switch is similar to DEVS but is for allocator
routines.

DQS - (I) If this switch is non-zero, the MEMQ or CPUQ are
displayed whenever that queue is changed.

DMS - (I) If this switch is non-zero, the memory map is displayed
every time it is changed.

DSJS - (I) If this switch is non-zero, a message is printed every
time the TSS swap area size is changed, and a sample of core utilization is displayed at every call of routine MMV.

**DUS** - (I) If this switch is non-zero a message is printed for every urgent user detected and every force swap performed.

**DKS** - (I) If this switch is non-zero a message is printed for each key I/O derail and courtesy call.

**DSS** - (I) If this switch is non-zero, a message is printed at the start and termination of every subsystem.

**DBEGIN** - (R) Simulation time at which diagnostic printing is to begin.

**DDUR** - (R) Duration of diagnostic printing.

**STREAM(1) ... STREAM(10)** - (I) These ten integer numbers are used as seeds for the ten Simscript pseudo-random number generators.

### 4.2 TSS Parameters

These input values consist of various TSS parameters that are used by the allocator for decision making. These parameters are described more fully in Reference 2, TSS Executive SMF. The values of each parameter in the system can be obtained from the listing of the communications region, TSSA.

**AMFTM** - (R) Maximum core fence maintenance time.

**LNSF** - (I) Number of swap files.

**TASWT** - (R) Minimum memory allocation wait time to cause further memory allocation and swap actions.

**TAMIS** - (R) Maximum high priority service program size.

**TALPP** - (R) Large program penalty factor.

**TASWF** - (R) Factor for program urgency calculation.

**TASI** - (R) Damper for urgent user size increases.
TLTLK - (R) Maximum time between line service calls.

TAMIT - (I) Minimum memo size increase increment.

TAMMS - (I) Maximum TSS swap area size. (In the real system this is total TSS memory size.)

AMTQ - (R) Minimum core residency time before consideration for force swap.

TAGVI - (R) Minimum time between request for core size increases.

TATNC - (R) Maximum time for core size change to stay pending.

TATMD - (R) Delay before scheduled size reduction is completed.

TAMAW - (R) Delay before informing user "NOT ENOUGH CORE TO RUN JOB".

TLNLM - (R) Minimum time between periodic line service functions.

TASMS - (I) Minimum TSS swap area size. (In the real system this is total TSS memory size.)

TAMRI - (R) Minimum time between size reduction considerations.

TAPMR - (R) Value of TAPMU above which memory reduction is not requested.

TASRI - (I) Memory size reduction amount.

TASCF - (R) Minimum interval between urgent user size increases.

ASD31 - (R) Minimum in interval between entries to SDP3 for scan of urgent users.

TCDEL - (R) Time slice for each subsystem dispatch.

4.3 Driving Data

The data under this category consists of the data collected from the actual system that is to be used to drive the model. All this data is obtained from the real systems using TSS accounting records and the programs mentioned in References 3 and 4.
INITCORE - (I) Initial TSS swap area size.

INTMEAN - (R) Mean interrupt interarrival times (cpu time/interrupt).

MINDELAY, MAXDELAY - (R) Lower and upper limits for a uniform distribution of time delay after each interrupt.

DIOMEAN - (R) Mean disk I/O service time (real time).

OUTMEAN - (R) Mean duration of a keyboard output operation (real time).

USTIAT - Distribution of subsystem interarrival times (real time/subsystem).

NOKIN - (I) Distribution of keyboard inputs per subsystem.

NOKOUT - (I) Distribution of keyboard output per subsystem.

NODIO - (I) Distribution of disk I/O's per subsystem.

SWAPDUR - (R) Distribution of swap channel service time (real time).

CPUDUR - (R) Distribution of cpu time used by subsystems.

KIODUR - (R) Distribution of keyboard input durations (real time).

SIZEDIST - (I) Distribution of subsystem program sizes.

The input data must be terminated with the number 9999. The program uses this as a check to make sure the input data had the correct number of data elements.

Figure 2 shows a sample input data set.
| Figure 2. Sample Input Data Set | 14 |
5. Output

The normal simulation output consists of five categories as described below. The final section deals with output messages printed when the diagnostic switches are set on.

5.1 Input Echo

The first part of the output consists of an echo of the entire input data except for the distributions in Section 4.3.

5.2 Memory Statistics

The mean and standard deviation of the following TSS swap area statistics are printed.

1. Program sizes in core.
2. Hole sizes.
3. Total swap area size.
4. Total used core size.
5. Percentage core utilization.

The values in this section are accumulated automatically by Simscript every time their values are sampled in routines MXIV and CORSAMP.

5.3 Overall Mean Rates

This section contains the mean rate of occurrence per hour of the following TSS events and processes.

1. Keyboard inputs.
2. Keyboard outputs.
3. Disk I/O's.
4. Swap outs (total).
5. Swap outs (due to keyboard I/O).
6. Swap outs (due to key output only).
7. Force swap.
8. TSS swap area size increases.
9. TSS swap area size decreases.
10. Total subsystem CPU time.
11. Subsystem starts.
12. Subsystem terminations.
13. Dispatches of the CPU to subsystems.
14. Urgent users detected.
15. Entries to Processor Allocator.
17. Allocator idles.
18. Interrupts.
19. Total core swapped.

These items are accumulated explicitly in the program and converted to hourly rates in routine OUTPUT.

5.4 Typical Subsystem Behavior

This section contains information on the behavior of a typical subsystem that executed during the simulation period. These statistics are collected at each subsystem termination as in the real system or the TSS accounting records. (See Ref. 3).

1. Subsystem interarrival time.
2. Keyboard input interarrival time.
4. Disk I/O interarrival time.
5. Program size.
6. CPU time (sampled).
7. CPU time (used).
8. Response time (R19).
9. Response time (individual).
10. Time spent in each time state:
   (a) Non-useful core residency.
   (b) Swap.
   (c) Useful core residency.
   (d) Out of core.
   (e) Awaiting memory.
   (f) Awaiting memory after being, force swapped.

11. Keyboard inputs.
15. Total swaps.

Items 1 through 6 are obtained from the samples of the input distributions and are useful for verifying that the input distributions were correct.

Items 6 and 7 differ because the former is the sampled time whereas the latter is computed only on those subsystems that terminate during the simulation period.

Item 8 is the response time as computed from the TSS accounting records in Reference 3. This response time is the total response time divided by the total number of keyboard inputs and outputs. The "individual" response time consists of samples of response time taken at every courtesy call for a keyboard operation.

5.5 System Queue Data

This section gives information on the average number of subsystems in each system queue as well as the average number of subsystems
BEGINNING OF SIMULATION AT 0.

STOPTIME DEVS DRS DQS DMS DSCS DUS
10800000.000000 0 0 0 0 0 0

DKS DSS DBEGIN DDUR INITIME
0 0 10800001.000000 0.3600000.000000

RANDOM NUMBER SEEDS
57839
84934
62192
72206
58023
34548
34886
88730
75442
78955

AMFT LNSF TASWT TAMIS TALPP TASWF TASID
20000.000000 4 3000.000000 36 4 .008000 1000.000000

TLTLM TAMII TAMMS AMTQ TAGMI TAIMC TAIMD
500.000000 7 58 7000.000000 14000.000000 60000.000000 30000.000000

TAMAW TLNLM TASMS TAMRI TAPMR TASRI TASCF
15000.000000 3000.000000 20 300000.000000 75 5 30000.000000

ASD31 TCDEL INITCORE INTMEAN MNDLAV MXDLAV DIOMEAN
1000.000000 25.000000 40 30.000000 7.000000 16.000000 R8.29999

OUTMEAN
2200.000000

SIMULATION TERMINATION AT 10800000.000

CORE STATISTICS MEAN STANDARD DEVIATION
PROGRAM SIZES 7.677 4.59893
HOLE SIZES 4.980 9.11604
SWAP AREA SIZE 48.785 8.08320
TOTAL USED CORE 26.746 12.83447
PERCENT CORE USED 54.265 24.29143

Figure 3. Sample Simulation Output
OVERALL MEAN RATES (PER HOUR)

KEYBOARD OUTPUTS 2338.00
KEYBOARD INPUTS 1665.50
DISK I/O'S 12188.50
SWAP OUTS 1891.50
SWAP OUTS (KEY I/O) 1827.00
SWAP OUTS (KEY OUTPUT) 551.50
FORCE SWAPS 46.50
SIZE INCREASES 4.50
SIZE DECREASES 6.00
SUBSYSTEM CPU TIME 52245.906 MS.
SUBSYSTEM STARTS 760.50
SUBSYSTEM KILLS 754.50
SUBSYSTEM DISPATCHES 69012.50
URGENT USERS 47.00
ENTRIES TO PAP 80558.00
ENTRIES TO MAP 3302.00
ALLOCATOR IDLES 11545.50
INTERRUPTS 69003.00
TOTAL CORE SWAPPED 13017.50 K

SUBSYSTEM STATISTICS MEAN STANDARD DEVIATION

SUBSYSTEM IAT 4737.04 6200.02
KEY INPUT IAT 178.42 1453.09
KEY OUTPUT IAT 242.71 1790.60
DISK I/O IAT 62.39 881.83
PROGRAM SIZE R.14 5.03
CPU TIME (SAMPLED) 1146.56 5179.26
CPU TIME (USED) 1040.62 5006.96
RESPONSE TIME (RMS) 3238.88 22184.43
RESPONSE TIME (INDIVIDUAL) 1740.16 13153.81
TIME IN STATE
NON-USEFUL CORE 8159.79 31824.45
SWAP 1607.56 4395.30
USEFUL CORE 5053.05 28368.11
OUT OF CORE 34956.78 150524.82
AWAIT MEMORY 347.57 1776.40
AWAIT MEMORY AFTER FS 30.91 336.93
NO. OF KEY IN'S 1.83 7.52
NO. OF KEY OUTS 3.10 11.25
NO. OF DISK I/O'S 15.43 195.43
NO. OF FORCE SWAPS .06 .43
NO. OF SWAPS 3.80 13.15

QUEUE LENGTHS MEAN STANDARD DEVIATION

PROCESSOR QUEUE 1.11 1.17
MEMORY QUEUE 11.23 2.91
DISK I/O QUEUE .40 .68
USERS SWAPPING .35 .80
USERS IN CORE 3.49 1.78

URGENT USERS .02 .16
USERS WAITING FOR CORE .48 .54
USERS ELIGIBLE FOR CPU .93 1.06

Figure 3 (Contd.). Sample Simulation Output

19
performing special activities,

1. Processor queue.
2. Memory queue.
3. Disk I/O queue.
4. Number of users swapping.
5. Number of users in core.
6. Number of urgent users.
7. Number of users waiting for core.
8. Number of users eligible for the CPU.

The queue lengths are maintained and accumulated by Simscript automatically. The other items are updated explicitly and accumulated automatically.

Figure 3, on the following page, shows a sample simulation output.

5.6 Diagnostic Output

This section describes the types of messages printed when each of the diagnostic switches described in section 4.1 are turned on.

1. **DEVS** - A message of the type

   eeee AT ttttt.t

   is printed whenever an event is entered. eeee is the event name and ttttt.t is the simulation time. For events that are associated with a subsystem (e.g. derails and courtesy calls), one of the phrases

   BY nnnn
   FOR nnnn
   OF nnnn
   TO nnnn

   are postfixed to the message as appropriate. nnnn is the subsystem index number.
This switch also causes the printing of the message

\[
\text{START AT \ ttttt.t OF \ mnnn \ SIZE = cccc} \\
\text{CPU = sssss.s}
\]

in routine START. cccc and sssss.s are the program size and cpu time allocated to the new subsystem.

2. DRS - A message of the type

\[
\text{rrrr CALLED}
\]

is printed at the beginning of every routine execution. rrrr is the name of the routine. This message is not printed for routine START.

3. DQS - The MEMQ and CPUQ are printed whenever their members are changed or rearranged. The UST's in the queue are printed in the order in which they appear in the queue. For each UST the following line is printed.

\[
nnn \ b1 \ b2 \ b3 \ b4 \ b5 \ b6 \ b7 \ s
\]

nnnn is the index number of the UST. b1 through b7 represent the flag word bits FL18, FL19, FL21, FL22, FL23, FL24 and FL34 respectively. s represents one of six states that the UST can be in. s takes on values from 0 through 5 and represents the following subsystem states:

0 Non-useful core residency.
1 Swap.
2 Useful core residency.
3 Out of core.
4 Awaiting memory.
5 Awaiting memory after force swap.

4. DMS - The memory map is printed whenever it is altered. For each entry in the map the following line is printed

\[
nnn \ pppp \ hhhh
\]
where nnnn is the UST index number, \( pp \) is its program size and hhhh is the size of the hole succeeding the program in core. The first entry in the core map, which is a dummy entry has an undefined value for nnnn and a program size of zero.

5. **DSCS** - A message is printed whenever TSS core size is changed. The two possible messages are:

- **SIZE INCREASED TO ssssK AT ttttt.t**
- **SIZE DECREASED TO ssssK AT ttttt.t**

where ssss is the new core size and ttttt.t is the time of the size change. In addition, whenever routine CORSAMP is called (at the start of every subsystem) the values of total, used and percent core used are printed as follows:

\[
\text{AT ttttt.t TOTCOR} = \text{sss} \quad \text{USED} = \text{ttt} \quad \text{ppp.p\% \ (MEAN = mmm.m\%)}
\]

6. **DUS** - A message is printed whenever an urgent user is detected and a force swap is performed.

The urgent user message is:

\[\text{nnnn FOUND URGENT FOR sssss.s MS. AT ttttt.t}\]

where nnnn is the UST index number, sssss.s is the time elapsed since this UST was first discovered urgent, and ttttt.t is the current simulation time.

The force swap message is:

\[\text{nnnn FORCE SWAPPED AT ttttt.t}\]

7. **DKS** - A message is printed at every keyboard input and output derail and courtesy call. The derail messages are:

- **nnnn START INPUT AT tttt.t UNTIL ssss.s**
- **nnnn START OUTPUT AT tttt.t UNTIL ssss.s**
The courtesy call messages are:

```
nnnn  FINISHED KIO AT  tttt.t
```

where nnnn and tttt.t are as before, and ss.s is the time of the scheduled courtesy call for the I/O operation.

8. **DSS** - Messages are printed at the start and termination of every subsystem. The start message is identical to that for DEVS.

The start message is:

```
STOP AT  tttt.t  OF  nnnn  KIN = iii  KOUT = jij
TOT RESP = ss.s  CPU ALLOC = uuuu.u  USED = vvvv.v
```

where nnnn and tttt.t are as before, iii and jjj are the total keyboard inputs and outputs performed by the subsystem respectively, ss.s is the total response time accumulated. uuuu.u and vvvv.v are the CPU time allocated to the subsystem and total time used by it and must always be equal (See Sec. 8).

6. **Using the Program**

The program was tested and run on a Honeywell 6000 computer under the GCOS operating system. The use of this program is covered in two sections, compilation and execution.

6.1 **Compilation**

The program can be compiled by the CACI Simscript II.5 compiler for HIS 600/6000 computers (USAF release 9). Compilation requires approximately 61k words of storage and 7 minutes of processor time. The following deck would compile the program in the source file SIM and store the object deck in the file SIMOB.
To compile the program from TSS the following underlined commands
must be issued. It is assumed that the above lines are in the file COMP.

```
SYSTEM ?CARDIN OLD COMP
READY
*RUN
SNUMB # 1234
```

To examine the output of the compilation from TSS the following
commands must be entered.

```
SYSTEM ?JOUT 1?3u*
FUNCTION?SCAN 74
FORM?DUMP
EDIT?YES
?PRINT /***/ERROR/;*
?DONE
FUNCTION?DIRECT ONL
```
If no error comments are printed in response to the print command it indicates that the compilation was error-free.

6.2 Execution

To execute the program the following commands are required.

Execution requires 25 K words of storage and approximately fifteen minutes of processor time for every hour simulated. It is assumed that the data to be used is in the file SIMDATA.

1000 #S,U,J ,8,16,32
1010 $ IDENT BFCAUMQI,MULLA J ,558102110001,UNIV. OF MICHIGAN
1020 $ LOWLOAD
1030 $ OPTION FORTRAN
1040 $ LIBRARY SL
1050 $ SELECT BFCAUMO1/SIMO8
1060 $ EXECUTE
1070 $ LIMITS 50,25K,-3K
1080 $ PRMFL SL,R,S,SMSCP2.5/LIBRARY
1090 $ PRMFL 17,R,S,SMSCP2.5/ERRORS
1100 $ FILE B*,B1R
1110 $ DATA I*
1120 $ SELECTA BFCAUMO1/SIMDATA
1130 $ ENDJOB

To run the program from a terminal the following underlined commands must be entered. It is assumed that the above GCOS commands are in the file RUN.
To examine the output from the program on TSS the following commands
must be issued.

SYSTEM ?JOUT 5678T
FUNCTION?PRINT 06

7. Detailed Program Description

The following is a detailed, line by line description of each routine
and event in the program. The order is as in the program listing in
Appendix 3.

7.1 PREAMBLE

8 - 11 Text substitutions for the compiler.
18 - 33 Random variable distribution entities. See Section 4.3.
38 - 49 Definition of UST entity and its attributes.
50 Definition of memory queue ordering on LSIZE attribute.
54 - 65 Definition of event routines.
66 Definition of CCQ set and its limited attributes.
70 - 71 Definition of memory map arrays.
72 - 91 Definition of global variables. See Appendix 1 for details.
95 - 134 Definitions of variables for automatic collection of
  statistics and global counters.
7.2 INIT

145 - 162 Read and echo input data from sections 4.1 and 4.2. (Diagnostic control and TSS parameters).

163 Read input values for distributions in Section 4.3. (Driving data).

164 - 167 Read check and stop if it is not 9999.

172 Reserve array locations for memory map vectors.

176 - 183 Initialize memory map. Link all blocks in a doubly linked list. Create dummy block as first one in map.

187 - 189 Schedule the arrival of the first subsystem and I/O interrupt.

192 - 195 Schedule entry to allocator termination of simulation, and event to test diagnostic switches.

7.3 MAIN

201 Call INIT to initialize program

202 Start simulation clock.

7.4 PDIAG

212 - 215 If the call is at the beginning of the diagnostic period set all internal switches to the values read in INIT.

218 - 220 At the end of the diagnostic period reset all switches to zero.

7.5 REINIT

226 - 233 Call Simscript generated routines to initialize counters for TALLY and ACCMULATE variables.

235 - 238 Reset all global counters to zero.
7.6 CORSAMP
246 - 253 Scan through the memory map and add up the program sizes in USEDCORE and both program and holesizes in TOTCOR.
254 Calculate percent core used in PERUSED.
255 - 256 Print core size and percentage used for diagnostics.

7.7 MQPRINT
262 - 268 For each UST in the memory queue print the index number, flag word bits and state number.

7.8 CPPRINT
274 - 280 Do the same as MQPRINT for each UST in the processor queue.

7.9 STERN
286 Call OUTPUT to print simulation results and then terminate the simulation.

7.10 OUTPUT
291 - 292 Print time at which simulation was terminated.
296 - 304 Print memory utilization statistics.
308 - 331 Print the number of certain TSS events that occurred per hour during the simulation period.
335 - 364 Print statistics on an average subsystem the executed during the period.
368 - 380 Print statistics on queue lengths.

7.11 SNAP.R
389 - 393 Print next arrival time for each event in the program.
394 Print arrival time of derail for the UST that was being served last by the allocator.
Print arrival times of all courtesy calls in the CCQ.

Call OUTPUT to print simulation results up to this point.

7.12 ALLOC

Call LINSV or MAP if their respective flags are set.

Increment count of entries to allocator and set allocator flag.

Scan CPUQ for jobs that are not doing I/O and not swapping.

See if job found is scheduled for force swap. If so call SDP7.

If not call RCTSSX to dispatch cpu to job.

Reset the UST's new subsystem bit.

If no jobs eligible for cpu, reset allocator flag and go to LINSV.

7.13 MAP

Increment count of entries to MAP.

Reset map flag.

If special memory action flag is set, call SPMACT.

Set time of last entry into MAP.

If no urgent user is waiting, go to \( 7.2A. \)

If urgent user can fit in current core fence, go to MAP.3.

If he has been urgent for more than 657M ms, reject him and clear the core fence.

Scan MF24Q for jobs that are not doing keyboard I/O, not in core and not swapping.

If such a job was not force swapped to \( \to MAP.4 \) and try to swap him in immediately.
Otherwise remember the first job we find eligible for swap in.

If no swap candidate was found and no special memory action is waiting return to ALLOCI.

If special memory action waiting, call swap decision processor.

If swap candidate was found call MBA to allocate storage.

If allocation was unsuccessful call swap decision processor.

If successful, call SWIN to swap him in.

Return to MAP to allocate storage for the next program.

If urgent user got core, destroy the core fence.

If he is in the MEMQ and still needs core call MBA to allocate it. Otherwise go to MAP to look for another swap candidate.

If more swaps are in progress than there are swap files, go back to ALLOCI.

Reset the pointer to job to be swapped.

Scan the MEMQ for jobs that are doing keyboard I/O and are in core and not swapping.

If the size of such a job exceeds the required size set AMNI to that UST.

If no swap candidate was found call SDP.

Call SWOUT to swap out job found.

Repeat SDP for more jobs to swap out.
7.15 SDP3

526 - 529 Enter only if last entry was at least ASD31 ms. ago or five unsuccessful attempts were made previously.

530 - 535 Reset counter, entry time, number of urgent users found, memory needed, largest urgent user wait time.

539 - 543 Find user that needs memory and has been waiting at least TASWT ms.

547 - 550 Calculate job wait factor depending on size of program.

551 - 552 See if wait factor calculated is less than time waited. If so he is eligible to be urgent.

555 - 560 If he was force swapped, reset the force swap bit and change his state from "waiting memory after force swap" to "waiting memory", but do not consider him urgent.

563 - 564 If he was found urgent, increment urgent user counts.

566 - 567 If USWITCH is set print urgent user message.

568 - 569 If he is the longest waiting urgent user, set ITALUT to his UST.

572 - 573 Record number of urgent users detected in this pass through SDP3. Call SDP4.

7.16 SDP4

581 - 582 If no urgent user was found call SDP6.

583 - 589 Call SDPS if (i) longest wait time is less than TASID ms., (ii) it has been less than TASCF ms. since the last size change, or (iii) a size change request is already scheduled.

592 - 298 Calculate a new (larger) size for TSS swap area.

599 Increment size increase requests.
600  Set special memory action flag.
601  Call SDP5 to set up core fence.

7.17  SDP5

610  If core fence is already up call SDP6.
611 - 613  If urgent user size is greater than TSS swap area size call MBA3 to increase size.
614 - 615  Set core fence size (2AURWT) needed to LSIZE of urgent user.
616  Set time fence was established.
618  Call SDP6 to force swap jobs.

7.18  SDP6

627  If special memory action waiting go directly to scan for force swap jobs.
628 - 632  If an urgent user was found and swap files are not full begin scan for force swaps. Otherwise return to ALLOCI.
633 - 635  Scan CPUQ for a job that has been in core more than AMTQ ms.
636  If the job is not new and is not already scheduled for a force swap, set the force swap bit.
637 - 638  If the job is doing I/O return to ALLOCI.
639 - 640  Otherwise call SDP7 to force swap it immediately.
643  Return to ALLOCI.

7.19  SDP7

652 - 659  Call SWOUT to swap job out. Increment force swap counters for UST and system. Move UST from CPUQ to MEMQ. Print diagnostic messages for QSWITCH and USWITCH.
660 Return to ALLOCI.

7.20 SPMACT

670 - 672 If no size change is scheduled, clear flag and return.
673 If size decrease is scheduled go to SPM.3.
676 If size has been increased less than TAGMI ms. ago, ignore this request.
678 - 679 Print diagnostics for increase.
683 If new size requested is current size or less than 3K, ignore it.
684 - 688 If there is no hole at the tail end of core large enough for the reduction, or the request has been pending longer than TATMC ms. or less than TATMD ms. go to SPM.5 to see if request should be ignored.
690 - 691 Print diagnostics for decrease.
695 Update memory map entry.
696 - 697 Clear size request and request flag.
699 Set time of last change or change attempt.
703 - 706 Ignore request if time since last request is greater than TATMC.

7.21 MBA

714 Clear success return flag.
715 Increment count of entries.
719 - 726 Check memory map for a hole big enough to satisfy request.
729 - 731 If search was unsuccessful and program will not fit in TSS size, call MBA3.
734 - 737 If allocated job was the urgent user, clear the core fence.

738 - 740 If he is not the urgent user, make sure he does not use up the core fence set up for the urgent user.

744 - 759 Build and link new entry into the memory map.

760 Set flag to indicate that allocation was successful.

761 Call MMV to verify memory map is still good.

7.22 MBA3

770 - 772 If urgent user has waited more than TAMAW ms. for core, return.

773 - 775 If a size increase is already scheduled, return.

776 Set new request to size of program.

777 Set the special memory action flag.

778 - 780 If urgent user needs more than TAMII plus the current TSS swap area size, return.

781 - 783 Otherwise set the new size request to TAMII more than the current size but not more than TAMMS.

7.23 MBD

793 - 797 Scan memory map for the job with the same pointer as the one to be deallocated.

800 - 807 Remove the entry from the core map and relink preceding and succeeding entries.

808 If the deallocated program was larger than 2K, set the MAP flag.

809 Call MMV to verify the memory map.
7.24 MEMV

823 - 832 Scan every entry in the memory map and add up each program and hole size in TOTCOR.

825 - 827 Print diagnostics when MSWITCH is set.

833 Sample value of CORSIZE for statistics.

837 - 839 Compare TOTCOR with the current TSS swap area size. If they are not equal print an error message and stop.

844 - 848 Scan the MEMQ for jobs waiting for core (i.e., not doing I/O, not swapping, not in core and not new subsystem).

849 - 853 Scan the CPUQ for jobs eligible for the cpu (i.e., not doing I/O, not swapping, not scheduled for a force swap and in core).

7.25 SWOUT

861 Clear "in core" bit.

862 Set "swapping out" bit.

863 Increment swap count.

864 - 865 If keyboard I/O is in progress, set "roadblocked" bit, and clear "scheduled for force swap" bit.

866 Call SWPLD to swap out.

7.26 SWIN

875 Set "swapping in" bit.

876 Call SWPLD to perform swap in.

877 Increment count of jobs in core.

7.27 SWPLD

883 DIR specifies direction of the swap to be performed.

(0 = in, 1 = out.)
886 Increment subsystem's swap count.
887 Change subsystem state to "swapping".
888 Increment count of jobs swapping.
890 - 891 Schedule swap in courtesy call. File the courtesy call in CCQ.
892 Schedule swap out courtesy call.
893 Add program size to total core swapped out.
894 File the courtesy call in CCQ.

7.28 SSFINI
902 Increment count of subsystem terminations.
903 - 904 Decrement count of jobs in core. Change subsystem state to "out of core".
905 Call MBD to release core.
908 - 911 Remove job from whichever queue it was in and print queue diagnostics if necessary.
914 - 920 If an output courtesy call is still pending for this subsystem, cancel it.
921 Call SACT to take subsystem accounting and destroy UST entity.

7.29 BUFDMP
930 For output operation, bypass this routine and call KIOSRT.
931 - 933 For input, move the UST from the CPUQ to the MEMQ.
934 Change subsystem state to "non-useful core residency".
935 Set "I/O roadblocked" bit.
937 Print queues for diagnostic purposes.
938 Call KIOSRT to start I/O operation.

7.30 KIOSRT

950 - 954 If keyboard I/O is already in progress (FK19 = 1), it must be an output since the subsystem was executing. In this case, cancel the output courtesy call.

955 Set "I/O in progress" bit.

958 If operation is an output:

959 Increment subsystem's count of outputs.

960 - 965 If no output is in progress collect response time value since the last key I/O courtesy call.

966 Set LTIN to the arrival time of the courtesy call.

967 - 969 Schedule the output courtesy call and file it in the COQ.

970 - 971 Print diagnostic message if KSWITCH is set.

975 If operation is an input:

976 - 981 Increment subsystem's count of inputs.

976 - 981 If no previous output is in progress, collect a response time sample.

982 - 986 Schedule input courtesy call and print diagnostic message if KSWITCH is set.

7.31 START

992 Increment count of subsystem arrivals.

994 Call CORSAMP to sample values for memory map statistics.

997 Create a new UST entity.

998 - 999 Sample, assign and tally values for program size and cpu time allocated.
100 - 1006 Sample values for the number of disk I/O's, keyboard inputs and outputs for this subsystem, and calculate the mean interarrival time between each. If zero operations are sampled the interarrival time is set to RINF.C (Simscript constant for the largest possible real number).

1007 Set time until next disk I/O, and key input and outputs as half the interarrival time. The I/O operations are equally spaced in cpu time throughout the subsystem execution period.

1008 Set "new subsystem" bit.

1009 Assign sequential index number.

1010 Initialize LTIN, LTMWT.

1011 File the UST in the memory queue.

1012 - 1013 Print message for diagnostics.

1014 Change subsystem state to "awaiting memory".

1015 Print MEMQ if QSWITCH is set.

7.32 RETSSX

1026 - 1027 Set NXTCC to the arrival time of the next courtesy call.

1028 Set NEXT to the smallest of the following times:

- Time until the next disk I/O.
- Time until the next key input.
- Time until the next key output.
- Cpu time remaining for the subsystem.
- Time until the next interrupt.
- Subsystem quantum (TCDEL).
- Time until next courtesy call.
Initialize DISPT to the time of dispatch.
Reduce each of the above times by the minimum value.
Schedule the derail which was to occur next. In case TCDEL, NXTINT or NXTCC was the least time, schedule an EXENTR to re-enter the allocator via an interrupt.

Sample each of the following values from the UST entity and assign them to their respective global tallied variables:
- Time spent in each of the six subsystem states.
- Keyboard inputs.
- Keyboard outputs.
- Disk I/O's.
- Force swaps.
- Swaps.

Calculate response time by dividing total response time accumulated by the total number of keyboard I/O's. If no key I/O's were done, response time is total subsystem duration.
Sample subsystem cpu time allocated.
Print message if SSSWITCH is set.

Add current time minus time of previous call to ATCHG for this subsystem to the accumulated time for the previous state (specified by LTCW).
Set LTCW to new state (N).
Update time of last state change.
7.35 EXENTR

1097 - 1098 Move the interrupted UST to the last position in the CPUQ.

1099 Print CPUQ for diagnostic aid.

1100 Call EXEACT to take accounting.

1101 Re-enter the allocator via ALLOCI.

1104 Sample the arrival time of the next interrupt.

7.36 EXEACT

1115 Increment count of interrupts.

1116 - 1118 Add cpu time used by subsystem to totals for subsystem and overall system.

1119 If MAP was last called more than ASD3I ms. ago, set MAP flag.

1120 If LINSV was last called more than TLTLM ms. ago, set LINSV flag.

7.37 KONDRL

1128 Set input operation flag.

1129 Call EXEACT to take accounting.

1130 Increment global key input count.

1131 Call dUFDMP to process input.

1134 Reset time until next input to mean input arrival time.

1135 Return to ALLOCI.

7.38 KOTDRL

1143 Set output operation flag.

1144 Call EXEACT to take accounting.

1145 Increment global key output count.
1146 Call BUFDMP to process output.
1149 Reset time until next output to mean output arrival time.
1150 Return to ALLOCI.

7.39 DRLDIO
1158 Set "disk I/O in operation" bit.
1159 Call EXEACT to take accounting.
1160 - 1161 Increment UST's and system's count of disk I/O operations.
1162 Reset time until next disk I/O to mean disk I/O arrival time.
1166 - 1168 If the DIOQ is empty, schedule the courtesy call for this UST and file it in the CCQ.
1169 File the UST at the end of the DIOQ.
1170 Return to ALLOCI.

7.40 DRLRET
1178 Call EXEACT to take accounting.
1181 Check to see that the subsystem utilized all the cpu time allocated.
1182 - 1183 If not, print an error message.
1184 Call SSFINI to terminate the subsystem.
1185 Return to ALLOCI.

7.41 KIOCC
1191 Save UST pointer in TUST and set UST to the subsystem for which KIOCC was scheduled.
1192 - 1193 Print message if KSWITCH set.
1196 Set time of courtesy call in LTIN.
1197 Reset "data in transmission" bit.
1200 If operation was an input:
1202 Reset "roadblocked" bit.
1203 - 1207 If program is not swapping out and not in core, move the UST from the MEMQ to the CPUQ. Change state to "useful core residency".
1208 Print MEMQ and CPUQ for diagnostics.
1211 If program is not in core, set MAP flag, and change state to "awaiting memory".
1213 Restore UST pointer and return.

7.42 DIOCC
1220 Save UST pointer as in KIOCC.
1223 Remove the UST from the top of the DIOQ. He must be the one whose courtesy call is being serviced.
1224 Clear "disk I/O in progress" bit.
1227 - 1228 Move the UST to the top of the CPUQ.
1229 Print CPUQ for diagnostics.
1230 Restore UST pointer.
1234 - 1237 If there are more UST's in the DIOQ, schedule the first one.

7.43 IALLCC
1244 Save the UST pointer as in KIOCC.
1247 Decrement count of users in core.
1248 Clear "swapping out" bit.
1249 Set MAP work flag on.
1251 - 1253 If UST was not doing key I/O and was force swapped change state to "awaiting memory after force swap".
1255 If UST was not force swapped and not doing key I/O change state to "awaiting memory".
1258 If UST was doing key I/O change state to "out of core".
1259 Call MBD to deallocate core.
1260 Decrement count of jobs swapping.
1261 - 1262 Restore UST pointer. Remove the courtesy call from the CCQ and destroy it.

**7.44 2ALLCC**

1268 Save UST pointer as in KIOCC.
1271 Set "program in core" bit.
1272 - 1273 Clear "swapping in" and "scheduled for force swap" bits.
1274 - 1275 Move the UST from the MEMQ to the top of the CPUQ.
1276 Change state to "useful core residency".
1277 Print MEMQ and CPUQ for diagnostics.
1278 Decrement count of users swapping.
1279 - 1280 Restore UST pointer, remove this 2ALLCC call from the CCQ and destroy it.

**7.45 LINSV**

1288 Clear LINSV flag.
1289 Clear "no users" switch.
1290 Reset time of last call to LINSV.
1292 Test if at least TLNLM ms. have passed before last entering the next section. If not, bypass the next section and go to MSRK.
1293 Update TLOLD, time of entry to this section.
1295 Test to see if both CPUQ and MEMQ are empty. If so, set MSR240 indicating no users.
If the current core size is not at the minimum go to MSRKS to try to reduce it.

If core is at its minimum, go to MSR300 to relinquish.

Clear "no users" flag.

If less than TAMRI ms. passed since the last time through the next section, skip to MSR300 to scan for new subsystems.

Calculate percentage core being used and average it with the previous value.

Find the largest program in the system.

If no users, skip to MSR250.

Skip memory reduction if either of the following is true:

(i) Time since last here is less than TAMRI ms.
(ii) An urgent user was found since the last time through here.
(iii) The last size change was less than TATMD ms. ago.
(iv) A program has been awaiting memory since the last time through here.
(v) The percent core utilization (TAPMU) is greater than TAPMR.

If current size minus TASRI is less than the minimum size, set change request to minimum. Otherwise set request to reduction by TASRI.

If a change request is already scheduled, ignore this one.
1329 If new size is less than largest program found, ignore it.
1330 If time since last change is less than TASCI, ignore this request.
1331 Set special memory action flag.
1332 Set TAHOJ to the new requested core size.
1333 Increment count of reduction requests.
1339 If the arrival time of the next UST is past, call START to create it.
1340 - 1341 Sample time of next UST arrival.
1342 Loop back to see if next arrival is also past.
1343 If either Allocator or MAP flags are set, return to ALLOCI.
1348 Increment count of idles.
1349 If MAP was last called more than ASD3I ms. ago, set MPWF.
1350 - 1351 Set time of next courtesy call.
1352 - 1353 Schedule an entry into the allocator at the arrival of the next UST, courtesy call or TAGMAI ms., whichever is smallest.
8. Error Messages

This section describes only those error messages printed by the program. For Simscript generated error messages see Reference 5.

1. After reading the input data the program reads a check value and compares it with a preset value (see Section 7.2). If this value is not correct the program terminates with the following error message.

### ERROR - INPUT FORMAT INCORRECT

2. When the memory buffer allocator (MBA) is called to find memory for a certain job, it checks to see if the program is already in core. If such a call is made the error is considered fatal and the following message is printed.

### ERROR - MBA CALLED FOR JOB ALREADY IN MEMORY

3. The memory map arrays (see section 3) are all dimensioned 50. If more than fifty jobs are ever allocated memory, the MBA process prints the following message and terminates the program. In this case the dimension figure in the INIT routine must be increased.

### ERROR - NO MORE AVAILABLE BLOCKS FOR MEMORY MAP ARRAY

4. The memory map verification routine (MMV) prints a fatal message if the map does not verify. This occurs if the sum of all programs and holes in the map does not equal the total core size. The message printed is:

### ERROR - MEMORY MAP DOES NOT VERIFY

5. The only non fatal error occurs when a program terminates without using up all the cpu time allocated to it. In such a case the return detail (DRLRET) prints the following message:

### ERROR - nnnn ALLOCATED

CPU = tttt.t      USED = ssss.s

46
where nnnn is the UST index number and tttt.t and ssss.s are the allocated and used cpu times respectively.
**APPENDIX 1**

**Variable Description**

The following is an alphabetically ordered list of variable names used in the program and their functions. When a name is followed by (M/S) it means that its mean and standard deviation are automatically collected by Simscript in the variables Mname and Sname respectively.

For certain TSS system constants, their values at the time this program was debugged are given in parentheses. Most variable and parameter names are the same as those in TSS and can be found described in Reference 2.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALOCI</td>
<td>Number of entries to the allocator.</td>
</tr>
<tr>
<td>ALUTM</td>
<td>Time at which urgent user was last detected.</td>
</tr>
<tr>
<td>AMAP</td>
<td>Number of entries to MAP.</td>
</tr>
<tr>
<td>AMAP2</td>
<td>Number of times the core fence limit was exceeded.</td>
</tr>
<tr>
<td>AMBA</td>
<td>Number of entries to MBA.</td>
</tr>
<tr>
<td>AMBA4</td>
<td>Number of times a hole fit was detected by MBA.</td>
</tr>
<tr>
<td>AMBA5</td>
<td>Number of successful allocations after hole fit detected.</td>
</tr>
<tr>
<td>AMFTM</td>
<td>Maximum force fence maintenance time (20000 ms.).</td>
</tr>
<tr>
<td>KKN</td>
<td>Pointer to UST for SDP.</td>
</tr>
<tr>
<td>AMN2</td>
<td>Size of memory requested from SDP.</td>
</tr>
<tr>
<td>AMTQ</td>
<td>Minimum core residency time (7000 ms.).</td>
</tr>
<tr>
<td>APAP1</td>
<td>Number of allocator idles.</td>
</tr>
<tr>
<td>APAP2</td>
<td>Number of times subsystem selected for processor allocation.</td>
</tr>
<tr>
<td>ASDP</td>
<td>Number of entries to SDP because program awaiting memory.</td>
</tr>
<tr>
<td>ASDP7</td>
<td>Number of force swaps.</td>
</tr>
</tbody>
</table>
ASD3C -- Counter to override timer for entry to SDP3.

ASD3I -- Minimum time between entries to SDP3 (1000 ms.).

ASD3T -- Time of last entry to SDP3.

AVAIL -- Pointer to the list of available blocks in the memory map arrays.

CCQ -- Queue of pending courtesy calls. Ordered FIFO.

CHANGE -- Temporary variable to hold core size change in LINSV.

CHECK -- Check value (9999) to verify that data was read correctly in INIT.

CHKCPU -- Allocated cpu time attribute of UST. Used to check used against alloc'd cpu time.

CORSIZE -- Global sample of TACOR in MMV (M/S).

CPUDUR -- Input distribution of cpu time per subsystem.

CPUQ -- Processor queue.

DBEGIN -- Time at which diagnostic printing is to start.

DDUR -- Duration of diagnostic period.

DEVS -- Diagnostic switch for event trace.

DIOIAT -- UST attribute, disk I/O interarrival time.

DIOMEAN -- Input mean of disk I/O service time for exponential distribution.

DIOQ -- Disk I/O queue. Ordered FIFO.

DIR -- Argument for SWPLD routine to specify direction of swap.

1 = out, 0 = in.

DISKIO -- Global count of disk I/O's.

DISPT -- Time at which last subsystem was dispatched.

DKS -- Diagnostic switch for key I/O trace.

DMS -- Diagnostic switch for memory activity trace.
DQS - Diagnostic switch for queue activity trace.
DRS - Diagnostic switch for routine trace.
DSCS - Diagnostic switch for core size and precentage core utilization trace.
DSS - Diagnostic switch for subsystem start and termination trace.
DUS - Diagnostic switch for urgent user and force swap trace.
DUST - Temporary location for UST in DIOCC.
ELIGCPU - Number of UST's eligible for the processor (M/S).
ES - Temporary index.
EVDIAG - Short for "FOR ES=1 TO EVSWITCH".
EVSWITCH - See DEVS.
EXT - Variable for accumulating time used to execute TSS code.
The values for EXT were calculated from the actual TSS listings with the assumption that the average instruction execution takes approximately 2.2 microseconds.
FK19 - UST attribute bit, "data in transmission."
FL18 - UST attribute bit, "disk I/O in progress."
FL19 - UST attribute bit, "keyboard I/O in progress."
FL21 - UST attribute bit, "swap out in progress."
FL22 - UST attribute bit, "program in core."
FL23 - UST attribute bit, "swap in in progress."
FL24 - UST attribute bit, "new subsystem."
FL34 - UST attribute bit, "force swap scheduled."
HEAD - Pointer to the head entry in the memory map.
HOLSIZE - Samples of hole sizes in memory map (M/S).
I - Temporary index.
IDPTR - Memory map array to hold UST pointer.
INCORE - Number of UST's in core (N/S).
INDDRL - Flag set by keyboard I/O derails. 1 = input, 0 = output.
INITCORE - Input data value of initial TSS swap area size.
INITIME - Input data value of time at which statistic counters are to be reinitialized via event REINIT.
INTMEAN - Input data value of mean interrupt interarrival time for exponential distribution.
J - Temporary index.
JOBNO - UST attribute, subsystem index number.
KEYIN - Global count of keyboard inputs.
KEYOUT - Global count of keyboard outputs.
KIIAT - UST attribute, keyboard input interarrival time.
KILL - UST attribute, remaining cpu time until termination.
KIODUR - Input data distribution, keyboard input duration.
KOIAT - UST attribute, keyboard output interarrival time.
KOSWAP - Global count of swaps due to keyboard output activity after subsystem termination.
KSWITCH - See OKS.
KUST - Temporary location for UST in KIOCC.
LNSF - Maximum number of swap files (4).
LSFLG - Line service flag.
LSIZE - UST attribute, program size.
LSPTS - UST attribute, subsystem processor time used.
LSTIO - UST attribute, disk I/O count.
LTCW - UST attribute, current time state.
LTC21 - UST attribute, keyboard input count.
LTC22 - UST attribute, keyboard output count.
LTC31 - UST attribute, force swap count.
LTC32 - UST attribute, swap count.
LTIN - UST attribute, response timer.
LTMRS - UST attribute, accumulated response time.
LTMWT - UST attribute, working timer.
LTM0 - UST attribute, non useful core residency time.
LTM0SS - Global sample of LTM0 (M/S).
LTM1 - UST attribute, swap time.
LTM1SS - Global sample of LTM1 (M/S).
LTM2 - UST attribute, useful core time.
LTM2SS - Global sample of LTM2 (M/S).
LTM3 - UST attribute, out of core time.
LTM3SS - Global sample of LTM3 (M/S).
LTM4 - UST attribute, waiting for core time.
LTM4SS - Global sample of LTM4 (M/S).
LTM5 - UST attribute, waiting for memory after force swap time.
LTM5SS - Global sample of LTM5 (M/S).
MAPTM - Time of last entry to MAP.
MEMALOCOK - Return value from memory buffer allocator (MBA).
0 = unsuccessful, 1 = successful.
MEMQ - Memory queue, ordered by increasing LSIZE.
MSDELAY - Input data value, minimum of uniform distribution for interrupt delay.
MPACT - Special memory action flag.
MPWF - MAP work flag.
"MS." - Short for "UNITS".
MSR240 - "No users" flag in LINSV.
MSWITCH - See DMS.

MXDELAY - Input data value, maximum of uniform distribution for interrupt delay.

N - Argument for ATCHG, new time state of UST.

NEXT - Cpu time interval allocated to subsystem in RETSSX.

NODIO - UST attribute, number of disk I/O's.

NOKIN - UST attribute, number of keyboard inputs.

NOKOUT - UST attribute, number of keyboard outputs.

NXTCC - Time until next courtesy call.

NXTDIO - UST attribute, cpu time until next disk I/O.

NXTINT - Cpu time until next interrupt.

NXTKIN - UST attribute, cpu time until next keyboard input.

NXTKOUT - UST attribute, cpu time until next keyboard output.

NXTUST - Real time until next subsystem arrival.

OUTCC - UST attribute, pointer to pending output courtesy call.

OUTMEAN - Input data value, mean output duration.

PERUSED - Percent core utilization (M/S).

PRED - Memory map array of backward pointers.

PROGSIZE - Samples of program sizes in memory map (M/S).

QSWITCH - See DQS.

RDIAG - Short for "FOR RS=1 TO RSWITCH PRINT 1 LINE THIS".

RESP - Global sample of response time calculated for individual key I/O operations (M/S).

RS - Temporary index.

RSWITCH - See DQS.

R6 - Temporary UST pointer in MAP.

SCSWITCH - See DSCS.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFUSE</td>
<td>Number of UST's swapping (M/S).</td>
</tr>
<tr>
<td>SHOLE</td>
<td>Memory map array to hold hole values.</td>
</tr>
<tr>
<td>SHOW</td>
<td>Short for &quot;PRINT 1 LINE WITH TIME.V&quot;.</td>
</tr>
<tr>
<td>SIZEDIST</td>
<td>Input data distribution of program sizes.</td>
</tr>
<tr>
<td>SIZEINCR</td>
<td>Number of TSS swap area size increases.</td>
</tr>
<tr>
<td>SIZERED</td>
<td>Number of TSS swap area size reductions.</td>
</tr>
<tr>
<td>SJOB</td>
<td>Memory map array to hold program sizes.</td>
</tr>
<tr>
<td>SPMFR</td>
<td>Time of last size increase request.</td>
</tr>
<tr>
<td>SSCPU</td>
<td>Global sample of .LSPTS (M/S).</td>
</tr>
<tr>
<td>SSDIO</td>
<td>Global sample of LSTIO (M/S).</td>
</tr>
<tr>
<td>SFSWAP</td>
<td>Global sample of LTC31 (M/S).</td>
</tr>
<tr>
<td>SSKIN</td>
<td>Global sample of LTC21 (M/S).</td>
</tr>
<tr>
<td>SSKOUT</td>
<td>Global sample of LTC22 (M/S).</td>
</tr>
<tr>
<td>SSRESP</td>
<td>Global sample of response time calculated by dividing the total response time by total keyboard I/O's (M/S).</td>
</tr>
<tr>
<td>SSSWAP0</td>
<td>Global sample of LTC32 (M/S).</td>
</tr>
<tr>
<td>SSWITCH</td>
<td>See DSS.</td>
</tr>
<tr>
<td>STOPTIME</td>
<td>Input data value, simulation duration.</td>
</tr>
<tr>
<td>SUC</td>
<td>Memory map array for forward pointers.</td>
</tr>
<tr>
<td>SUST1</td>
<td>Temporary variable to hold UST pointer in 1ALLCC.</td>
</tr>
<tr>
<td>SUST2</td>
<td>Same as SUST1 for 2ALLCC.</td>
</tr>
<tr>
<td>SWAPDUR</td>
<td>Input data distribution, swap channel service time.</td>
</tr>
<tr>
<td>T</td>
<td>Temporary variable.</td>
</tr>
<tr>
<td>TAAUG</td>
<td>Number of urgent users detected.</td>
</tr>
<tr>
<td>TACOR</td>
<td>TSS swap area size.</td>
</tr>
<tr>
<td>TAGMI</td>
<td>Minimum time between size increases (14000 ms.).</td>
</tr>
<tr>
<td>TAGPT</td>
<td>Cumulative cpu time used by TSS since last relinquish to GCOS.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TAGTC</td>
<td>Number of interrupts.</td>
</tr>
<tr>
<td>TACTU</td>
<td>Sum of subsystem cpu time.</td>
</tr>
<tr>
<td>TAHOL</td>
<td>New requested TSS swap area size.</td>
</tr>
<tr>
<td>TAIL</td>
<td>Pointer to last entry in memory map.</td>
</tr>
<tr>
<td>TALCT</td>
<td>Time of last size change attempt.</td>
</tr>
<tr>
<td>TALPP</td>
<td>Large program penalty multiplier for wait factor calculation in SDP3 (4).</td>
</tr>
<tr>
<td>TALPS</td>
<td>Largest program detected in LINSV size reduction logic.</td>
</tr>
<tr>
<td>TALUT</td>
<td>Longest urgent user wait time.</td>
</tr>
<tr>
<td>TAMAW</td>
<td>Maximum wait time for core before rejection of program (150000 ms.).</td>
</tr>
<tr>
<td>TAMII</td>
<td>Minimum memory size increment (7K).</td>
</tr>
<tr>
<td>TAMIS</td>
<td>Maximum size of programs considered under high priority (36K).</td>
</tr>
<tr>
<td>TAMMS</td>
<td>Maximum TSS swap area size (58K).</td>
</tr>
<tr>
<td>TAMRI</td>
<td>Minimum time between size reduction attempts (300000 ms.).</td>
</tr>
<tr>
<td>TAMRU</td>
<td>Minimum value of TAPMU required to prevent size reduction (75%).</td>
</tr>
<tr>
<td>TAPMU</td>
<td>Moving average percent core utilization.</td>
</tr>
<tr>
<td>TASCIF</td>
<td>Minimum time between size increase requests due to urgent users (30000 ms.).</td>
</tr>
<tr>
<td>TASID</td>
<td>Damper for urgent user size increases (1000 ms.).</td>
</tr>
<tr>
<td>TASIO</td>
<td>Count of swap outs due to key I/O.</td>
</tr>
<tr>
<td>TASMS</td>
<td>Minimum TSS swap area size (20K).</td>
</tr>
<tr>
<td>TASRI</td>
<td>Memory size reduction amount (5K).</td>
</tr>
<tr>
<td>TASRT</td>
<td>Time of last entry to size reduction logic.</td>
</tr>
<tr>
<td>TASWF</td>
<td>Divisor to convert size into time units for wait factor calculations in SDP3 (.008).</td>
</tr>
</tbody>
</table>
TASWT - Minimum wait time before swap decision logic is invoked (3000 ms.).

TATMC - Maximum time allowed for size change to occur (60000 ms.).

TATMD - Delay before scheduled size reduction request is completed (30000 ms.).

TATMN - Total program size belonging to urgent users.

TAKRG - Current number of urgent users (M/S).

TAUSE - Temporary variable to add up size being used.

TCDEL - Subsystem cpu time slice (25 ms.).

TEMP - Temporary variable.

TKILL - Number of subsystem terminations.

TLFLG - Allocator work flag.

TLST - Time of last entry to LINSV.

TLNAA - Count of TSS idles.

TLNLM - Interval between line service functions (3000 ms.).

TLOLD - Time of last entry to idle status check in LINSV.

TLTLM - Minimum time between scan for new UST's in LINSV (500 ms.).

TOTCOR - Temporary variable to accumulate TSS core size in MMV.

TSIRC - Number of times size increase was requested because of waiting program.

TSRRC - Number of size reduction requests.

TSRT - Number of subsystem starts.

TSWAP - Number of program swaps.

TSWPK - Total core swapped.

TUST - Temporary UST pointer storage.

URCT - Urgent user wait time.

USEDSIZE - Amount of TACOR being used by programs (M/S).
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UST</td>
<td>Global pointer to UST currently being served by the allocator.</td>
</tr>
<tr>
<td>USTIAT</td>
<td>Input data distribution, UST interarrival times.</td>
</tr>
<tr>
<td>USWITCH</td>
<td>See DUS.</td>
</tr>
<tr>
<td>VCPUDUR</td>
<td>Sampled values of CPUDUR (M/S).</td>
</tr>
<tr>
<td>VDIOIAT</td>
<td>Sampled values of DIOIAT (M/S).</td>
</tr>
<tr>
<td>VKIIAT</td>
<td>Sampled values of KIIAT (M/S).</td>
</tr>
<tr>
<td>VKOIAT</td>
<td>Sampled values of KOIAT (M/S).</td>
</tr>
<tr>
<td>VSIZE</td>
<td>Sampled values of SIZEDIST (M/S).</td>
</tr>
<tr>
<td>VUSTIAT</td>
<td>Sampled values or USTIAT (M/S).</td>
</tr>
<tr>
<td>WAITCOR</td>
<td>Current number of users waiting for core (M/S).</td>
</tr>
<tr>
<td>WAKET</td>
<td>Time to reenter TSS after relinquish to GCOS.</td>
</tr>
<tr>
<td>WTFAC</td>
<td>Wait factor calculated for subsystem.</td>
</tr>
<tr>
<td>IACRWT</td>
<td>UST pointer to urgent user for whom core fence is established.</td>
</tr>
<tr>
<td>ITALRT</td>
<td>Pointer to UST that has been urgent the longest.</td>
</tr>
<tr>
<td>ZACRWT</td>
<td>Program size of UST for which core fence is established.</td>
</tr>
</tbody>
</table>
## Appendix 2

### Flowcharts

<table>
<thead>
<tr>
<th>Routine</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>1</td>
</tr>
<tr>
<td>MAIN</td>
<td>2</td>
</tr>
<tr>
<td>PDIAG</td>
<td>2</td>
</tr>
<tr>
<td>REINIT</td>
<td>3</td>
</tr>
<tr>
<td>CORSAMP</td>
<td>3</td>
</tr>
<tr>
<td>MQPRINT</td>
<td>4</td>
</tr>
<tr>
<td>CQPRINT</td>
<td>4</td>
</tr>
<tr>
<td>STEEM</td>
<td>5</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>5</td>
</tr>
<tr>
<td>SNAP.R</td>
<td>6</td>
</tr>
<tr>
<td>ALLOCI</td>
<td>7-8</td>
</tr>
<tr>
<td>MAP</td>
<td>9-12</td>
</tr>
<tr>
<td>SDP</td>
<td>13-14</td>
</tr>
<tr>
<td>SDP3</td>
<td>14-17</td>
</tr>
<tr>
<td>SDP4</td>
<td>18-19</td>
</tr>
<tr>
<td>SDP5</td>
<td>20</td>
</tr>
<tr>
<td>SDP6</td>
<td>21-22</td>
</tr>
<tr>
<td>SDP7</td>
<td>22</td>
</tr>
<tr>
<td>SPMACT</td>
<td>23-26</td>
</tr>
<tr>
<td>MLA</td>
<td>26-28</td>
</tr>
<tr>
<td>MBA3</td>
<td>29</td>
</tr>
<tr>
<td>MBD</td>
<td>30-31</td>
</tr>
<tr>
<td>MSW</td>
<td>31-32</td>
</tr>
<tr>
<td>SWOUT</td>
<td>33</td>
</tr>
<tr>
<td>SWIN</td>
<td>33</td>
</tr>
<tr>
<td>SWPLD</td>
<td>34</td>
</tr>
<tr>
<td>SSFINI</td>
<td>35-36</td>
</tr>
<tr>
<td>BUFDM</td>
<td>36</td>
</tr>
<tr>
<td>KIOSRT</td>
<td>37-38</td>
</tr>
<tr>
<td>Routine</td>
<td>Page</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>START</td>
<td>39-50</td>
</tr>
<tr>
<td>RETSSX</td>
<td>50</td>
</tr>
<tr>
<td>SACT</td>
<td>51</td>
</tr>
<tr>
<td>ATCHG</td>
<td>41</td>
</tr>
<tr>
<td>EXENTR</td>
<td>42</td>
</tr>
<tr>
<td>EXEACT</td>
<td>2-3</td>
</tr>
<tr>
<td>KONDRL</td>
<td>5</td>
</tr>
<tr>
<td>KOTBRL</td>
<td>4</td>
</tr>
<tr>
<td>DRLDIO</td>
<td>44-45</td>
</tr>
<tr>
<td>DRLRET</td>
<td>5</td>
</tr>
<tr>
<td>KIOCC</td>
<td>46-47</td>
</tr>
<tr>
<td>DIOCC</td>
<td>47</td>
</tr>
<tr>
<td>'ALLCC</td>
<td>8</td>
</tr>
<tr>
<td>2ALLCC</td>
<td>29</td>
</tr>
<tr>
<td>LINSV</td>
<td>50-51</td>
</tr>
</tbody>
</table>
INIT

Read input parameters

Read CHECK

CHECK = 9999 ?

no → Print error message → STOP

yes → Reserve arrays for memory map

Initialize memory map

Sample next arrival time of subsystem, and interrupt

Schedule entry to allocator, simulation termination, diagnostic routine and reinitialization event

RETURN
MAIN

Call INIT and then release the routine

Start the simulation clock and proceed to the first event

STOP

PDIAG

Is this the first call to PDIAG?

yes

Set internal switches to the values read in routine INIT

no

Clear all diagnostic switches

Destroy the PDIAG event notice

RETURN
ITI

Call the Simscript generated routines to reset all counters for ACCUMULATE and TALLY variables

Reset all explicit counters to zero

RETURN

CORSAMP

Sample the program size and hole size of all entries in the memory map. (Except the program size of the first dummy entry.)

Add program sizes in TAUSE and program and hole sizes in TOTCOR

Sample USED SIZE as TAUSE

Calculate percent core used

RETURN
MQPRINT

For each UST in the MEMQ print the UST index, flag bits, state and size

RETURN

CQPRINT

For each UST in the CPUQ print the UST index, flag bits, state and size

RETURN
STERM

Call OUTPUT to print results

RETURN

OUTPUT

Print memory statistic report

Print global counts averaged per hour

Print statistics for average subsystem execution

Print statistics on queue lengths

STOP
SNAP.R

Print arrival time of next event for each event class

Print arrival of next UST, interrupt and derails for current UST

Print arrival time of each pending courtesy call

Call OUTPUT to print results

STOP
1/7

Clear allocator flag

Increment count of allocator idles

LINSV

2/7

Increment count of programs selected for processor allocation

Is this job scheduled for force swap?

yes

SOP7

no

Clear "new subsystem" flag

RETSSX
MAP

Increment count of entries to MAP

Clear MAP work flag

Special memory action waiting?

yes

no

mark last time through here in MAPTM

Urgent user waiting?

yes

no

Will it fit in fenced area?

yes

no

Been waiting more than AMPTM ms?

yes

no

call SPMACT

\[1\]

\[1\]

\[1\]

\[2\]

\[2\]
Increment number of times core fence limit was exceeded

Take down core fence

Set pointer to head to MEMQ

MAP.3
Get next entry in memory queue

End of MEMQ?

no

Program waiting for core?

yes

Program force swapped?

no

Did we already find such a program?

yes

no

Remember this one
MAP. 3B

Did we detect a force swap?

no

Is special memory action waiting?

no

ALLOC

MAP. 4

Call MBA to obtain memory

no

Set size required by program

yes

Call SWIN to swap job in

SDP

MAP
Take down core fence

Program needs core?

UST in memory queue?

\( \frac{1}{9} \)

\( \frac{2}{10} \)

\( \frac{2}{11} \)
Did we find a job eligible to swap?

- Yes: Call SWOUT to swap out program.
  - Clear swap program UST pointer in AMNI.
  - SDP
  - SDP3

- No: SDP3

Has it been more than ASD31 ms. since the last time through here?

- No: Increment ASD3C.
- Yes: Clear ASD3C.

1/3
Mark current time in ASD3T

Zero out number of urgent users, total memory needed and longest wait time

Set pointer to head of MEMQ

Get next UST in MEMQ

End of MEMQ?

User waiting for memory?

as he waited more than TASWT ms.?

Get his program size and add it to the total memory needed by urgent users

Page 15
Program size \(
\begin{align*}
\text{TAMIS} & \quad \text{yes} \\
\text{no} & \quad \text{no}
\end{align*}
\) \\

Compute wait factor by dividing by TASWF \\

Wait factor > TASWT ms. \\

Wait time less than wait factor \\

Was program force swapped \\

Use TASWT to prevent small programs from showing excessive urgency
1. Clear force swap bit

2. Call ATCHG to change state from "awaiting memory after force swap" to "awaiting memory"

2. Increment count of total and currently urgent users

2. Compute time user has been urgent

Is this ≥ longest wait time?

- no → 2.  
- yes → Make this the longest wait time, save the UST pointer in ITALUT

2.  

Page 17
was an urgent user detected?

Urgent user waited at least TASID ms.?

Has it been at least TASCF ms. since last size change?

Size increase already waiting?

Let new memory size request be current size plus TAMII

Is request > TAMMS?
1/10

Set new size request to current size

Is new request > TAMMS?

yes → SDP5

no

Let new request be TAMMS

2/10

Increment count of size
increase requests

Set special memory
action flag

SDP5
Set UST to urgent users pointer saved in ITALUT

Will program fit in current TSS memory?

Set up core fence for urgent user. Save UST pointer in 1AURWT and size needed in 2AURWT

Set time core fence was established

SDP6

SDP5

Is core fence already up?

no

yes

Call MBA3 to try size increase

SDP5

SDP6
Is this a new subsystem or is it already scheduled for a force swap?

Mark for force swap

Disk or key I/O in progress?

SDP7

Call SWOUT to swap user out

Increment count of force swaps for system and this UST

Move UST from CPUQ to MEMQ

ALLOCI
\begin{align*}
\frac{1}{23} & \\
\text{Is new size requested - 3K?} & \\
\text{Is hole size of last map entry big enough for reduction?} & \\
\text{Has request been pending more than TATMC ms.?} & \\
\text{Has request been pending less than TATMD ms.?} & \\
\text{Increment count of size reactions} & \\
\frac{1}{25} & \\
\frac{2}{25} & \\
\frac{4}{25} & \\
\frac{4}{25} & \\
\frac{4}{25} & \\
\frac{4}{25} & \\
\frac{1}{25} & 
\end{align*}
SPM.4
Update hole size of last entry to accomplish either size change

SPM.4A
Clear size change request

Clear special memory action flag and time of reduction request refusal

SPM.4B
Set time of last size change

RETURN

Has change been refused before?

no

So: time of change refusal

yes

SPM.4

RETURN
Is time since first refusal \( \geq \) TATMC ms.?

- If yes, go to step 2/25.
- If no, go to step 3/25.

**MBA**

- Clear success flag
- Increment count of entries to MBA
- Set pointer to head of memory map

**MBA.1**

- If yes, go to step 1/27.
- If no, go to step 2/27.
Print error message

STOP

Is the hole entry large enough for current UST's size?

Get next entry in memory map

End of memory map?

Increment number of times a fit was found

Is this the urgent user?

Take down core fence

Was the hole at the tail end of core?

Will the urgent user still fit in the remaining hole?
Add new entry into memory map

Zero hole for entry from which core was obtained and set hole of new entry to remaining hole

Set successful allocation flag

Call MMV to verify memory map

RETURN

Will program fit in current TSS core size?

yes

RETURN

no

Call MBA3 to attempt size increase

RETURN
MBA3

Has user waited more than TAMAW ms.?

no

Is current request for size increase ≥ program size?

no

Set new request of core size to size of waiting program

Was the change necessary ≥ TAMII?

no

Is current upper limit + TAMII ≤ TAMMS?

no

Use TAMMS as new size request

yes

RETURN

RETURN

RETURN

RETURN
MBD

Set pointer to head of memory map

MBD.1A

Is this the last entry to be deallocated?

yes

Get next entry in memory map

no

Last entry of memory map?

yes

RETURN

no

Remove entry from the map

Add the removed core and hole size to the hole of the previous entry

Was program ≥ 2K?

yes

Set MAP flag

no

Page 30
130

Call MMV to verify memory map

RETURN

MMV

Zero TOTCOR, total core size counter

Set pointer to head of memory map

NEXT

Add its program and hole sizes to TOTCOR

Get next entry in memory map

End of memory map?

no

yes

1/32
Take sample of core size from TOTCOR

Is TOTCOR = current TSF core size?

yes

Count up and sample number of UST's in MEMQ that are awaiting memory

Count up and sample number of UST's in CPU, that are eligible to use the CPU

no

Print error message

STOP

RETURN
SWOUT

Clear "in core" flag

Set "swapping out" flag

Increment count of system swaps

Is key I/O in progress?

yes

Set "I/O roadblocked" flag

call SWPLD to swap job out

RETURN

SWIN

Set "swapping in" flag

Call SWPLD to swap job in

Increment number of users in core

RETURN
SWPLD

Increment count of swaps for this UST

Call MATCHG to change state to "swapping"

Increment number of jobs swapping

Is swap in or out?

out

Schedule the swap out courtesy call and file it in the CCQ

return

in

Schedule the swap in courtesy call and file it in the CCQ

return

Add the program size to the total core swapped
SSFINI

Increment count of subsystem terminations

Decrement number of UST's in core

Call ATCHG to change state to "out of core"

Call MBD to deallocate memory

Remove the UST from either the CPUQ or MEMQ

Is key output in progress?

yes

Destroy the output courtesy call

Increment counts of swaps for system and UST

Add 1K to total core swapped and add a sample of swap duration to UST's swap timer.

no
Call SHCT to take accounting

Destroy the UST

RETURN

BUFDMF

Is this an output operation?

yes

no

Is the UST in the CPUQ?

yes

no

Move the UST from the CPUQ to the MEMO

Set allocator and MAP flags

Call ATCHG to change state to "non useful core residency"

Set "I/O roadblocked" flag

Call KIOSRT to begin I/O operation

RETURN
KIOSRT

Is an output already in progress?

yes

Cancel the output courtesy call

no

Set "data in transmission" flag?

yes

Is operation an output?

no

KIO.2

Increment count or UST's outputs

Is response timer < current timer?

yes

Collect response time for subsystem

no

Set response timer to the current time

Add output duration sample time to response timer

1/38

2/38

Page 37
1/37

Schedule output courtesy call and file it in the CCQ

RETURN

2/37

Increment count of UST's inputs

Is response timer < current time?

yes

Collect response, time for subsystem

no

Set response timer to current time

Schedule input courtesy call and file it in the CCQ

RETURN
START

Increment count of subsystem starts

Call CORSAMP to sample memory values

Create a new UST

Sample values for program size and cpu time

Sample values for disk I/O's, keyboard outputs and inputs for this UST and calculate their respective interarrival times

Set cpu time until arrival of first disk I/O, key input and output operation

Set "new UST" flag

Assign UST index number

Initialize response and state timers

Set Allocator and MAP flags

1

Page 39
Call ATCHG to set initial state to "awaiting memory"

RETURN

RETSSX

Find cpu time until first interrupting event

Set time of dispatch to subsystem

Subtract cpu time quantum found from interarrival time of each event

Schedule the interrupting event after the cpu quantum found. In the case of an external interrupt add a uniformly distributed delay

RETURN
SACT

Assign each UST attribute counter to its corresponding global sample variable

Calculate the response time per keyboard operation

RETURN

ATCHG

Add the time since the previous call to ATCHG for this UST to the state timer for the previous state

Set the new state index

Set the time of this last call to ATCHG

RETURN
EXENTR

Set the allocator flag

Move the UST to the end of the CPUQ

Call EXACT to take accounting

If the interrupt was due to an I/O interrupt, sample a new interrupt arrival time

ALLOC I

EXACT

Increment count of interrupts

Add cpu quantum used by subsystem to totals for UST and TSS

Was MAP last called more than ASD31 ms. ago?

Set MAP flag

1/43

Page 42
Was LINSV last called more than TITLM ms. ago?

- Yes: Set LINSV flag
- No: RETURN

KONDRL

Set flag to indicate input operation

Call EXACT to take accounting

Increment global count of inputs

Call BUFDMF

Reinitialize arrival time of next input

ALLOCI
KOTDRL

Set flag to indicate output operation

Call EXEACT to take accounting

Increment global count of outputs

Call BUFDMP to move UST to MEM

Reinitialize arrival time of next output

ALLOCI

DRLDIO

Set "disk I/O in progress" flag

Call EXEACT to take accounting

Increment global and UST's counts of disk I/O's

Reinitialize arrival time of next disk I/O
Is DIOQ empty?

yes

Schedule the disk I/O courtesy call and file it in the CCQ

no

File the UST last in the DIOQ

ALLOCI

DRLRET

Call EXEACT to take accounting

Did subsystem use all its allocated cpu time?

no

Print error message

yes

Call SSFINI to terminate subsystem

ALLOCI
Set LTIN to current time

Clear "data in transmission" flag

"I/O roadblock" flag set?

yes

Clear "I/O roadblocked" flag

no

Program in core?

yes

Set allocator flag

no

Set MAP flag

Set allocator flag

Move UST from MEMQ to top of CPUQ

Call ATCHG to change state to "useful core residency"

Call ATCHG to change state to "awaiting memory"

147
Remove this KIOCC from the CCQ and destroy it

RETURN

DIOCC

Remove the first UST from the DIOQ

Clear "disk I/O in progress" flag

Set allocator flag

Move the UST to the top of the CPUQ

Remove this DIOCC from the CCQ

yes

DIOQ empty?

no

Schedule a courtesy call for the first UST in the DIOQ and file it in the CCQ

RETURN

Destroy this DIOCC

RETURN
Decrement number of jobs in core

Clear "swapping out" flag and set MAP flag

If key I/O swaps

Call ATCHG to change state to "out of core"

Call ATCHG to change state to "awaiting memory"

Data in transmission?

Was program force swapped?

yes no

yes

CALLCHG to change state to "awaiting memory after force swap"

no

Increment count of key I/O swaps

ALLCCA

Call MBD to deallocate core

Decrement number of jobs swapping

Remove this IALLCC from the CCQ and destroy it

RETURN
In the core flag and clear "swapping in" and "force swapped" flags

Set allocator flag

Move the UST from the MEMQ to the top of the CPUQ

Call ATCHG to change state to "useful core residency"

Decrement number of jobs swapping

Remove this 2ALLCC from the CCQ and destroy it

RETURN
LINSV

Clear line service flag and "no users" flag

Set time of last entry into LINSV

Has it been > TLNLMS ms. since last time through next part?

no

yes

Mark last time through here

Are both CPUQ and MEMQ empty?

no

yes

Set "no users" flag

Is TSS size at minimum?

no

yes

...
Clear "no users" flag

Have TAMRI ms. passed since last time through here?

Calculate current percent core utilization and average it with the previous value

Find the largest program size in the system

Are there any users?

Have TAMRI ms. passed since last time through here?
Set last time through here

Has it been urgent user detected since the last time through here?

Has it been at least TATMD since last size change?

Has anybody been waiting for core more than TASWT ms. since last time through here?

Is the average percent core utilization > TAPMR?
Is current size minus TASRI greater than minimum TSS size?

- Yes: Set new size request to current size minus TASRI
- No: Set new request at minimum TSS size

Is size change already waiting?

- Yes: Increment count of size reduction requests
- No: Is new request less than largest program found?

- Yes: Increment count of size reduction requests
- No: Set special memory action flag
Set delay until reentry to a locator

ALLOCI
Appendix 3

Program Listing

BEST AVAILABLE
**Simulation Model for TSS Allocator**

**Preamble**

- Normally, TSS is integer and dimension is 0
- Define EVT as 6 for ES=1 to EVSTCH
- Define NR. to NEQ NOUT
- Define show to Flag with Time
- Define Print as 1 line thus
- Define itl as a feasible routine

**Definition of System Cues and Random Variables**

*Permanent Entities*

- The system has a CPU, a MEM, a CCC and a DIO
- The system has a CPU, a RAND, a LIN, and a RAND
- The system has a CPU, a STREAM and a RAND
- The system has a CPU, a STREAM and a RAND
- The system has a CPU, a STREAM and a RAND
- The system has a CPU, a STREAM and a RAND
- The system has a CPU, a STREAM and a RAND
- The system has a CPU, a STREAM and a RAND
- The system has a CPU, a STREAM and a RAND
- The system has a CPU, a STREAM and a RAND

*Temporary Entities*

- Every NEQ has a L1T, L1T2, L1T3, L1T4, L1T5, L1T6, L1T7, L1T8, L1T9, L1T10
- L1T1, L1T2, L1T3, L1T4, L1T5, L1T6, L1T7, L1T8, L1T9, L1T10
- L1T11, L1T12, L1T13, L1T14, L1T15, L1T16, L1T17, L1T18, L1T19, L1T20
- L1T21, L1T22, L1T23, L1T24, L1T25, L1T26, L1T27, L1T28, L1T29, L1T30

*Variable Definitions*

- Define NEQ as a set ranked by LCONV

---

**Page 116**
DEFINITION OF EVENTS

EVENT NOTICES

INCLUDE ALLOC, MAP, SDF, SDF3, SDF4, SDF5, SDF6, SDF7, KORDL

KORDL, DRLIO, DRLII, LINSV, EVENT, RESSX, STEM, REINIT

AND PEDAG

EVERY TALLOC HAS A PICTU, BELONGS TO THE CCQ,

HAS A S. CCQ IN LOK, 7, AND HAS A P. CCQ IN WOD 8

EVERY TALLOC HAS A SUST1, BELONGS TO THE CCQ,

HAS A S. CCQ IN WOD 7, AND HAS A P. CCQ IN WOD 8

EVERY TALLOC HAS A PICTU, BELONGS TO THE CCQ,

HAS A S. CCQ IN WOD 7, AND HAS A P. CCQ IN WOD 8

DEFINE CCQ AS A SET RANKED BY LOC TIME, A WITHOUT L, N AND M Attributes

DEFINITION OF MEMORY MAP ARRAYS AND DIAGNOSTIC SWITCHES

DEFINE HEAD, TAIL, AVAILABLE, USWITCH, USWITCH, SCSWITCH,

KSWITCH, ES, ES, TSMEM, VMEM, DEVS, DSS, DSS, DMEM

DEFCS, T, USWITCH, KSWITCH, DSS, DSS, SCSWITCH, DSS AS VARIABLES

DEFINITION OF FLAGS, VARIABLES AND COUNTERS

DEFINE LSFLG, TRAP, ALLOC, TLFLG, APART, APART, A.MAP,

MAPCT, I.RPM, 2UPXT, 3APRT, ASIP, ASIP, ASP, ASP, ASP,

ASS, TPUS, TATH, TALPUS, TPUS, TATH, TALPUS, TAIL

DEFINE MLLOC, MVHA, MPHAN, MAER, MCDRL, MAUS, MTEMP

DEFINE TAILLOC, PCLONE, PMVHA, PMPHA, PMBASE, MPKOR, PMAUS, PMTEMP

DEFINE MMAP, MMAP, MMAP, MMAP, MMAP, MMAP, MMAP, MMAP

DEFINE IMAP, IMAP, IMAP, IMAP, IMAP, IMAP, IMAP, IMAP

DEFINE TMAP, TMAP, TMAP, TMAP, TMAP, TMAP, TMAP, TMAP

DEFINE LMAP, LMAP, LMAP, LMAP, LMAP, LMAP, LMAP, LMAP

DEFINE ITMAP, ITMAP, ITMAP, ITMAP, ITMAP, ITMAP, ITMAP, ITMAP

DEFINIT TMAP, ITMAP, ITMAP, ITMAP, ITMAP, ITMAP, ITMAP, ITMAP

DEFINE CSS, CSS, CSS, CSS, CSS, CSS, CSS, CSS

DEFINE CSS, CSS, CSS, CSS, CSS, CSS, CSS, CSS

DEFINE CSS, CSS, CSS, CSS, CSS, CSS, CSS, CSS

DEFINE CSS, CSS, CSS, CSS, CSS, CSS, CSS, CSS

DEFINE CSS, CSS, CSS, CSS, CSS, CSS, CSS, CSS

DEFINE CSS, CSS, CSS, CSS, CSS, CSS, CSS, CSS

DEFINITION OF VARIABLES FOR COLLECTING STATISTICS

DEFINE KEYOUT, KEYIN, DISKIO, TSWAP, SIZEINC, SIZEDER

DEFINE VDIR, VOCTN, VDOR, VDIR, VDIR, VDIR, VDIR, VDIR

DEFINE VDIR, VOCTN, VDOR, VDIR, VDIR, VDIR, VDIR, VDIR

DEFINE VDIR, VOCTN, VDOR, VDIR, VDIR, VDIR, VDIR, VDIR

DEFINE VDIR, VOCTN, VDOR, VDIR, VDIR, VDIR, VDIR, VDIR

DEFINE VDIR, VOCTN, VDOR, VDIR, VDIR, VDIR, VDIR, VDIR

DEFINE VDIR, VOCTN, VDOR, VDIR, VDIR, VDIR, VDIR, VDIR

VARIES, VDIR, VOCTN, VDOR, VDIR, VDIR, VDIR, VDIR

DEFINITE LTMSS, LTMSS, LTMSS, LTMSS, LTMSS, LTMSS, LSRES, SRES

SSECU AS REAL VARIABLES

DEFINE SSOUT, SSDIO, SSTSWAP, SSSWAPP AS VARIABLES

TALLY WPLOGSIZE AS THE MEAN AND HOSLOGSIZE AS THE STD OF HOSLOGSIZE

TALLY HOSLOGSIZE AS THE MEAN AND HOSLOGSIZE AS THE STD OF HOSLOGSIZE
ROUTINE TO INITIALIZE SIMULATION

136 1 " " ROUTINE TO INITIALIZE SIMULATION
137 2 " " ROUTINE TO INITIALIZE SIMULATION

BEGINNING OF SIMULATION AT "***"."***

142 6 " " READ INPUT PARAMETERS, CONSTANTS AND DISTRIBUTIONS

144 8 " " READ INPUT PARAMETERS, CONSTANTS AND DISTRIBUTIONS

145 9 READ STOP TIME, DEVR, DES, DGS, DMS, DCS, BUS, DKS, DSS, DBEGIN,
146 10 PRINT, INIT

147 11 LIST STOP TIME, DEVR, DES, DGS, DMS, DCS, BUS, DKS, DSS, DBEGIN,
148 12 PRINT, END

149 13 PRINT 1 LINE, THUS

FACTOR NUMBER SEEDS
151 14 FOR J = 1 TO 10 DO
152 15 READ SEED.V(J) PRINT 1 LINE WITH SEED.V(J) THUS

153 16 ELSE

154 17 LOGIC

155 18 READ LPM, INSP, TART, TNMS, TLF, TASS, TASID, TTNLM,
156 19 TASA, TAIM, ATIC, TAGY, TATC, TTMD, TANM, TTLM,
157 20 TASS, TARS, TAPI, TASI, TASC, AANG, TDDEL,

158 21 LIST LPM, INSP, TART, TNMS, TLF, TASS, TASID, TTNLM,
159 22 TASA, TAIM, ATIC, TAGY, TATC, TTMD, TANM, TTLM,
160 23 TASS, TARS, TAPI, TASI, TASC, AANG, TDDEL,

151 16 FOR J = 1 TO 10 DO
152 15 READ SEED.V(J) PRINT 1 LINE WITH SEED.V(J) THUS

161 24 PRINT 1 LINE WITH "TICOME, TICOE, TICOME, TICOE, TICOME, TICOE"
162 25 PRINT 1 LINE WITH "TICOME, TICOE, TICOME, TICOE, TICOME, TICOE"
163 26 PRINT 1 LINE WITH "TICOME, TICOE, TICOME, TICOE, TICOME, TICOE"

164 26 CHECK
165 27 IF CHECK NE 9999 PRINT 1 LINE, THUS

166 28 = INPUT PATTERN INCORECT

167 29 STOP

168 30 ELSE

169 31 " " RESERVE ARRAYS FOR MEMORY MAP

170 32 " " RESERVE ARRAYS FOR MEMORY MAP

171 33 RESERVE SJOB, SHOLT, SUC, PRED, IDPFE AS 50

172 34 " " INITIALIZE MEMORY MAP

173 35 " " INITIALIZE MEMORY MAP

174 36 " " INITIALIZE MEMORY MAP

175 37 LET HEAD=1 LET TAIL=1
176 38 LET SJOB(1)=ITCORE LET SJOB(1)=0 LET TACOR=INICORE
177 39 LET TAVL=2 LET SU(1)=0 LET PRED(1)=0
178 40 FOR TM=2 TO 5 DO
179 41 LET SU(1)=1
180 42 LOOP

181 43 LET SU(5)=1

182 44 LET TAP=1 LET CORR=INICORE

183 45 " " TAKE FIRST SAMPLE OF UST ARRIVAL AND INTERRUPT

184 46 " " TAKE FIRST SAMPLE OF UST ARRIVAL AND INTERRUPT

185 47 LET XUST=USTFAT
186 48 LET XUST=USTFAT
188 49 LET VUSTAT=VUSTAT
189 50 LET XSTIT=EXPONENTIAL(Y(1)/EAK,9)
190 51 ""
191 52 "" SCHEDULE ENTRY TO ALLOCATOR AND TERMINATION EVENTS
192 53 SCHEDULE A ALLOC NOW LET LSFLG=1
193 54 SCHEDULE A STEEN AT STOPTIME
194 55 SCHEDULE A DTOG AT DEBEGIN "" SCHEDULE EVENT TO PRINT DIAGNOSTICS
195 56 SCHEDULE A DTOG AT INITIME "" SCHEDULE EVENT TO RE-INITIALIZE COUNTERS
196 57 RETURN END
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>197</td>
<td>2</td>
<td>MAIN ROUTINE</td>
</tr>
<tr>
<td>198</td>
<td>3</td>
<td>MAIN</td>
</tr>
<tr>
<td>200</td>
<td>4</td>
<td>MAIN</td>
</tr>
<tr>
<td>201</td>
<td>5</td>
<td>PERFORM INIT RELEASE INIT</td>
</tr>
<tr>
<td>202</td>
<td>6</td>
<td>START SIMULATION</td>
</tr>
<tr>
<td>203</td>
<td>7</td>
<td>STOP</td>
</tr>
<tr>
<td>204</td>
<td>8</td>
<td>END</td>
</tr>
</tbody>
</table>
EVENT TO START AND STOP PRINTING OF DIAGNOSTICS

205       1        "                    
206       2        "                    
207       3        "                    
208       4        "                    
209       5        "                    
210       6        "                    
211       7        "                    
212       8        "                    
213       9        "                    
214      10        "                    
215      11        "                    
216      12        "                    
217      13        "                    
218      14        "                    
219      15        "                    
220      16        "                    
221      17        "                    

RETURN END
222 1 "" EVENT TO RE-INITIALIZE COUNTERS FOR COLLECTION OF STATISTICS
223 2 "" EVENT RESET
224 3 ""
225 4 EVENT RESET
226 5 CALL R.PROGSIZE CALL R.VERSIZE CALL R.CORESIZE CALL R.USEDSIZE
227 6 CALL R.PROFUSED CALL R.VITESIZ CALL R.VKOSIZ CALL R.VUSIZ
228 7 CALL R.VDIEZ I CALL R.VSIZE CALL R.VCUDUR CALL R.VSIZOSS
229 8 CALL R.VLIMSS CALL R.VT12SS CALL R.VT33S CALL R.VLIMSS
230 9 CALL R.LT65SS CALL R.L55ESP CALL R.LSCPU CALL R.RESP
231 10 CALL R.L55SS CALL R.L55ESP CALL R.LSCPU CALL R.RESP
232 11 CALL R.L55SS CALL R.L55ESP CALL R.LSCPU CALL R.RESP
233 12 CALL R.L55SS CALL R.L55ESP CALL R.LSCPU CALL R.RESP
234 13 ""
235 14 LET KEYOUT=0 LET KEYIN=0 LET DISKOUT=0 LET TSWAP=0 LET TSIOM=0
236 15 LET AS:w=0 LET SIGNUM=0 LET SIZE=0 LET TSIZE=0
237 16 LET TAI=0 LET CTALL=0 LET APAF2=0 LET TAIAGE=0 LET ATAGM=0
238 17 LET ATAGM=0 LET APAF1=0 LET TSWP=0 LET TSIOM=0
239 18 RETURN END
This routine takes a snapshot of the memory map to output values for program size, hole size, and percentage core used.

DO NOT LOOK AT DUMMY PROG SIZE

LET TOICOR = (TOICOR + JCPB(I)) / SHOLE(I)

LET TAUS = TAUS + JCPB(I)

LET I = JUC(I)

IF I NE 0 GO TO NEXT

LET USER_SIZE = USER_SIZE + USE / TOICOR

AT ********** TOICOR = USER_SIZE / (MEAN**X)** (MEAN**X, X)
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>258</td>
<td>2</td>
<td>PRINT MEMO FOR DIAGNOSTIC AID</td>
</tr>
<tr>
<td>259</td>
<td>3</td>
<td>ROUTINE MOPRINT</td>
</tr>
<tr>
<td>260</td>
<td>4</td>
<td>PRINT 2 LINES THUS</td>
</tr>
<tr>
<td>261</td>
<td>4</td>
<td>TO STATE SIZE</td>
</tr>
<tr>
<td>262</td>
<td>4</td>
<td>FOR EACH I OF MEMO PRINT 1 LINE WITH J0190(I), FL18(I), FL19(I),</td>
</tr>
<tr>
<td>263</td>
<td>4</td>
<td>FL20(I), FL21(I), FL22(I), FL23(I), FL24(I), FL25(I),</td>
</tr>
<tr>
<td>264</td>
<td>4</td>
<td>LTOV(I), LSIZE(I) THUS</td>
</tr>
<tr>
<td>265</td>
<td>4</td>
<td>RETURN END</td>
</tr>
</tbody>
</table>

125
<table>
<thead>
<tr>
<th>Line</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>270</td>
<td>PRINT CPUG FOR DIAGNOSTIC AID</td>
</tr>
<tr>
<td>272</td>
<td>PRINT 2 LINE THIS</td>
</tr>
<tr>
<td>278</td>
<td>FOR EACH I OF CPUG PRINT 1 LINE WITH JORKO(I), FL18(I), FL19(I),</td>
</tr>
<tr>
<td>279</td>
<td>FL22(I), FL23(I), FL24(I), FL25(I), FL26(I), FL28(I), FL29(I),</td>
</tr>
<tr>
<td>281</td>
<td>RETURN END</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>282</td>
<td>1</td>
</tr>
<tr>
<td>283</td>
<td>2</td>
</tr>
<tr>
<td>284</td>
<td>3</td>
</tr>
<tr>
<td>285</td>
<td>4</td>
</tr>
<tr>
<td>286</td>
<td>5</td>
</tr>
</tbody>
</table>

**CAZL SINSCRIPT II., FOR HIS 600/6000 USAF RELEASE 9.**

282 
283 EVENT TO END SIMULATION
284 
285 EVENT STEM
286 CALL OUTPUT RETURN END
ROUTINE TO PRINT SIMULATION RESULTS.

SIMULATION TERMINATION AT **********.

PRINT CORE STATISTICS

SKIP 2 OUTPUT LINES

PRINT 6 LINES WITH MPROG SIZE, SPROG SIZE, MHOST SIZE, SHOST SIZE.

SKIP 2 OUTPUT LINES

OUTPUT LINES

PRINT MEAN RATES AND COUNTS

CUT-PADDY READ-OUTS (CPU JOUR)
<table>
<thead>
<tr>
<th>SUBSYSTEM STATISTICS</th>
<th>MEAN</th>
<th>STANDARD DEVIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM IAT</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>KEY INPUT IAT</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>KEY OUTPUT IAT</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>DISK I/O IAT</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>PAGE SIZE</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>CPU TIME (SAMPLED)</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>CPU TIME (USED)</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>RESPONSE TIME (R79)</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>RESPONSE TIME (INDIVIDUAL)</td>
<td>*****</td>
<td>*****</td>
</tr>
</tbody>
</table>

**STATE**

- **CPU USEFUL**
  - SWAP
  - CPU USEFUL
  - CPU OUT
  - CPU READY
  - CPU MEMORY AFTER FS
- **CPU KEY**
  - CPU KEY
  - CPU KEY OUT
  - CPU DISK I/O'S
- **CPU SWAPS**
  - CPU SWAPS

**PRINT QUEUE STATISTICS**

<table>
<thead>
<tr>
<th>QUEUE</th>
<th>MAN</th>
<th>STANDARD DEVIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCSTOP QUEUE</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>&quot;ENTRY QUEUE&quot;</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>&quot;EXIT QUEUE&quot;</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>USER STOPPING</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>USER TIMEOUT</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>USER WAITING FOR CPU</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>USER ELIGIBLE FOR CPU</td>
<td>*****</td>
<td>*****</td>
</tr>
</tbody>
</table>

**EXIT**

- STOP
1. **THIS ROUTINE IS CALLED IF THE SIMULATION IS TERMINATED**

2. **ABnormally FOR ANY REASON, IT PRIntS CERTAIN VALUABLE DATA FOR DEBUGGING AND LOCATING A POSSIBLE ERROR.**

3. **ROUTINE SNAP.I**

4. **PRINT EVENT TIMES FOR EVERY EVENT**

5. **FOR EVERY EVENT, V DO**

6. **PRINT A LINE WITH TIME,A(F.EV.S(I)) THUS**

7. **LIST TIME, V, NXTUST, NXTDIO, NXTKIN, NXTKOUT, KILL, NXTINT**

8. **PRINT ALL WAITING COURTESY CALL TIMES**

9. **FOR EACH I OF CTO PRINT A LINE WITH TIME,A(I) THUS**

10. **PRINT SIMULATION OUTPUT UPTO THIS POINT.**

11. **CALL OUTPUT**

12. **STOP END**
ENTRY TO ALLOCATOR

EVENT ALLOC

ALLOC AT *********

IF LSFQ EQ 1 SCHEDULE A LINSV IN .006 MS. RETURN

ELSE IF UPMF EQ 1 SCHEDULE A MAP IN .01 MS. RETURN

SCHEDULE NEXT SUBSYSTEM IN CPUG

ELSE 'PAP' ADD 1 TO ALOCI

LET TILFLG=1 LET EXT=.04

'PAP.1' FOR EACH UST OF CPUG DO

ADD .02 TO EXT

IF FL18+FL19+FL21+FL23 EQ 0

ADD 1 TO APAP2

IF FL34 EQ 1 SCHEDULE A SF7 IN EXT+.022 MS. RETURN

ELSE SCHEDULE A PRTSSX IN EXT+.01 MS.

LET FL24=0

RETURN

ALLOCATOR IS IDLE. CALL LIKE SERVICE

LET TILFLG=0

ADD 1 TO APAP1

SCHEDULE A LINSV IN EXT MS.

RETURN END
**Memory Allocator Process**

**EVENT MAP**

**ADD 1 TO MAP LET EXT=0**

**LET MPF=1**

**IF SPECIAL MEN ACTION WAITING**

**LET MPF=1 CALL MPXACT**

**ELSE LET MPF=TIME.V**

**CHECK UP ON URGENT USTR IF ANY**

**IF USTR DO ADD .025 TO EXT GO TO MAP,3**

**ELSE IF PL37=0 GO TO MAP,3**

**ELSE ADD 1 TO MAP2**

**LET MPF=0**

**FIND USERS TO SWAP OUT**

**FOR EACH USTR OF "MEM, DO,**

**IF FL19=PL21+PL22+PL23 EQ 0**

**ADJ,03 TO EXT**

**IF FL34 EQ 0 GO TO MAP,4**

**ELSE IF FL EQ 0 LET RE=UST**

**ELSE ELSE LOOP**

**IF WE DETECT A FORCETIME SWAP**

**LET MPF=1 CALL MPXACT EQ 0 IF MPF**

**SCHEDULE AN ALLOC IN EXT=0,01 MS.**

**RETURN**

**LET ANM=1**

**RETURN**

**ELSE LET UST=RE**

**IF TO GET CORE FOR JOB**

**CALL MBA**

**SUCCESSFUL, CALL SWAP DECISION PROCESSOR**

**IF UNSUCCESSFUL, CALL SWAP DECISION PROCESSOR**

**ADJ,1 TO ASDP**

**LET ANM=1 SIZE**

**RETURN**

**SUCCESSFUL, SWAP JOB IN**

**ELSE CALL SWP**

**RESCHEDULE A MAP IN EXT=0,03 MS.**

**RETURN**

**URGENT USER GET CORE, TAKE DOWN CORE FENCE**

**LET UST=1AUAMT**
482  52  ADD  0.33 TO EXT
483  53  LET 優惠 50
484  54  LET 2AUPFTE
485  55  IF  FL19+FL21+FL22+FL23 NE 0  GO TO MAP,3
486  56  ELSE IF  CST IS NOT IN MENU  GO TO MAP,3
487  57  ELSE  GO TO MAP,4
488  58  END
SWAP DECISION PROCESSOR

SWAP ALL JOBS > OR = SIZE REQUIRED

EVENT SDP

EVNDG SHOQ THUS

SDP AT **********

IF SDPUST GT 2*INSF IF NEAT EQ 0 SCHEDULE AN ALLOC IN 0.04 MS.

RETURN

CASE ELSE LET AMN1=0

SEARCH MEMORY QUEUE FOR KEY I/O BLOCKED JOBS

LET EXT=0, IS FOR EVERY UST OF MEMQ DO

IF 3.19-1122 EQ 2 AND 1121+1123 EQ 0

ADD 24 TO EXT

IF LSIZE_GE AMN2 GO TO SDP_ZE

ELSE LET AMN1=UST

ELSE LOOP

LET 2H FIND ANY JOB TO SWAP OUT

IF AMN1 EQ SCHEDULE AN SDP IN EXT MS.

RETURN

SWAP JOB FOUND

ELSE LET UST=AMN1

SWITZ CALL SWOUT

LET AMN1=0

SWAP FOR MORE JOBS

SCHEDULE AN SDP IN EXT MS.

RETURN END
**SCAN MEMORY QUEUE FOR URGENT USERS**

**EVENT SRP3**

**SYSDIG** SHOW THUS

AT *********

IF TMEVL-ASD3=0

ALO 1 TO ASD3C

IF ASD3C LET'S SCHEDULE AN ALLOC IN .077 MS.

RETURN

ELSE LET ASD3C=0

LET ASD2=TIME,V

LET ASD1=0

LET TIME,V=0

SCAN MEMORY QUEUE

LET EXT=0,00

FOR EVERY 3ST OF MEMORY DO

IF S=14+FL34+FL35+FL36 NE 0 GO TO SRP3X

ELSE ADD 0.257 TO EXT

IF TMEVL-LTHT LE TASS GO TO SRP3X

ELSE

CALCULATE JOB WAIT FACTOR DEPENDING

OF STAT AND TYPE ALREADY WAITED

IF S=14+FL34+FL35+FL36 LET WTPAC=1ST.

IF S=14+FL34+FL35+FL36 LET WTPAC=1ST+2ND.

IF S=14+FL34+FL35+FL36 LET WTPAC=1ST+2ND+3RD.

IF S=14+FL34+FL35+FL36 LET WTPAC=1ST+2ND+3RD+4TH.

IF S=14+FL34+FL35+FL36 LET WTPAC=1ST+2ND+3RD+4TH+5TH.

IF S=14+FL34+FL35+FL36 LET WTPAC=1ST+2ND+3RD+4TH+5TH+6TH.

IF S=14+FL34+FL35+FL36 LET WTPAC=1ST+2ND+3RD+4TH+5TH+6TH+7TH.

ELSE IF TMEVL-LTHT LE TASS LET WTPAC GO TO SRP3X

ELSE LET TMEVL-LTHT LE TASS LET WTPAC=0.

IF FL33=0 LET FL33=4

CALL ATC1(4)

ADD ... TO EXT

GO TO SRP3X

ELSE

REGISTER AS AN URGENT USER.

ADD 1 TO T

LET UNT=ULT-1ST-VLT=VTPAC

IF NSWITCH NE O THEN 1 LINE WITH JOBLST URG TMEVL THUS

FOUND URGENT FOR ********** MS. AT **********

ELSE LET URG TMEVL-1ST-2ND TASS

LET TASS=0 LET TASS=0

LET URG ADD 0.32 TO EXT

'SRP3X LOOP
79507 01 08-26-75 17.504 CACT SCRIPT II.5 FOR HIS 600/6000
USAF RELEASE 9.

572 51 LET TAURGE
573 52 SCHEDULE AN SDPW IN EXT RS; RETURN
574 53 END
1. **DECIDE WHETHER TO GROW TSS SIZE**

2. **EVENT SDP**

3. **EVERDIAG SDG**

4. **SDP**

5. **AT **

6. **RETURN**

7. **ELSE IF TALUT LE SCHEDULE AND SDP IN 0.005 MS**

8. **RETURN**

9. **ELSE IF TALUT LT TASID SCHEDULE AND SDP IN 0.009 MS**

10. **ELSE IF TIMESLT TALCF SCHEDULE AND SDP IN 0.017 MS**

11. **RETURN**

12. **ELSE IF TALOL GE TACOR SCHEDULE AND SDP IN 0.028 MS**

13. **RETURN**

14. **ELSE**

15. **RETURN**

16. **RETURN**

17. **LET TAHOL=TACOR**

18. **RETURN**

19. **LET TAHOL=TAMS**

20. **RETURN**

21. **ELSE**

22. **RETURN**

23. **ELSE**

24. **ADD 1 TO TAMS**

25. **LET \( FACT = 1 \)**

26. **RETURN**

27. **RETURN**
603  1  /* SET UP CORE FENCE FOR URGENT USER */
605  3  
607  4  EVENT SDP5
607  5  EVDIAG SHOW THRU
607  6  AT ****** *
607  7  LET EXT=0,0,5
607  8  IF 2HUNT OT 9 GO TO SDP,5m
610  9  IF LSUM GT FACCT CALL K313 ADD 0,037 TO EXT GO TO SDP,5b
613 10  ELSE
615 11  LET IAHP=14HST
615 12  LET 2A'KUI=14KZ
615 13  LET ALTH=TTE,Y
617 14  ADD 0,05 TO EXT
619 15  1SDP,3a SCHEDULE AN SDP6 IN EXT MS.
619 16  RETURN END
10-28-75 17:504 CACI SIMSCRIPT II.5 FOR HIS 600/6000  USAF RELEASE 9

1  "CHECK JOBS TO FORCE SWAP"

4 EVENT SDP6

5 END DIAG SHOW THUS

SDP6 AT "**********".

6 LET EXT=9.013

7 IF XPACT GT .20 TO SDP.6A

8 ELSE IF TALUT NE 0

9 LET SPROE LT 2*LN5F GO TO SDP.6A

10 IF 5E

11 ELSE SCHEDULE AN ALLOCI IN 0.015 MS.

12 RETURN

13 'SDP.6A' FOR EVERY USR OF CPUQ DO

14 ADD 9.013 TO EXT

15 IF TIME-V-LT-Wlicher TO

16 IF PL24=PL34 EQ 7 LET PL34=1

17 IF PL18=PL19 GT 0 SCHEDULE NF ALLOCI IN EXT+0.015 MS.

18 RETURN

19 ELSE SCHEDULE AN SDP7 IN EXT+0.015 MS.

20 RETURN

21 ELSE

22 ELSE LOOP

23 SCHEDULE AN ALLOCI IN EXT MS.

24 RETURN END
DUMP OUT FORCE SWAP JOBS

EVENT SDF7

EVDAG SHO:;JOBN0 THUS

AT **********, VO: ****

LET EXT=0,11

CALL S0UT

ADD 1 TO LTC31

ADD 1 TO REP7

REMOVE THIS UNIT FROM CPU

IF I/E=1 LET "P"=1 FILE THIS UNIT IN MEMO

IF COUNTER N= - CALL vsRAIS CALL COUNTER ELSE

* FORC SWAPPED AT **********

FILE SCHEDULE AN ALLOC IN EXT HR.

RETURN END
PERFORM SPECIAL MEMORY ACTION

(TSS MEMORY INCREASE OR DECREASE)

ROUTINE SPFACT

SPFACT CALLED

ADD 0.017 TO EXT

LET MPACT=0

RETURN

ELSE IF TAHOL LE TACOR GO TO SPN.3

INCREASE TSS CORE SIZE

ELSE ADD 0.017 TO EXT IF T TIMEV TACOR LT TACOL RETURN

ELSE IF TAHOL LT 3 GO TO SPN.4A

ELSE IF SHML(TAIL)=TACOR TAHOL GO TO SPN.5

ELSE LET T TIMEV SYSPRF

IF T GE TACOR GO TO SPN.5

ELSE IF T LT TACOR GO TO SPN.5

ELSE ADD 0.017 TO EXT IF T TIMEV THUS

DECREASE TSS CORE SIZE

SPN.3 IF TAHOL LT 3 GO TO SPN.4A

ELSE IF TACOR TAHOL LET TACOR TAHOL

LET T TIMEV SYSPRF

LET SYSPRF=0

LET TALT=TIMEV

RETURN

ELSE IF REQUEST HAS WAITED TOO LONG WITHOUT ACTION

ELSE LET T TIMEV SYSPRF ADD 0.017 TO EXT

IF T GE TACOR GO TO SPN.4A

ELSE GO TO SPN.4B

END
79501 01 08-28-75 17,504 CACI SIMSCRIPT II.5 FOR HIS 600/6000 USAF RELEASE 9.

760 50 ELSE LET XEQACOM1
761 51 CALL XEV
762 52 ADD 0.13 TO XEV
763 53 RETURN END
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>764</td>
<td>1</td>
</tr>
<tr>
<td>765</td>
<td>2</td>
</tr>
<tr>
<td>766</td>
<td>3</td>
</tr>
<tr>
<td>767</td>
<td>4</td>
</tr>
<tr>
<td>770</td>
<td>6</td>
</tr>
<tr>
<td>771</td>
<td>7</td>
</tr>
<tr>
<td>772</td>
<td>8</td>
</tr>
<tr>
<td>773</td>
<td>9</td>
</tr>
<tr>
<td>774</td>
<td>10</td>
</tr>
<tr>
<td>775</td>
<td>11</td>
</tr>
<tr>
<td>776</td>
<td>12</td>
</tr>
<tr>
<td>777</td>
<td>13</td>
</tr>
<tr>
<td>778</td>
<td>14</td>
</tr>
<tr>
<td>779</td>
<td>15</td>
</tr>
<tr>
<td>780</td>
<td>16</td>
</tr>
<tr>
<td>781</td>
<td>17</td>
</tr>
<tr>
<td>782</td>
<td>18</td>
</tr>
<tr>
<td>783</td>
<td>19</td>
</tr>
</tbody>
</table>
MEMORY BUFFER DEALLOCATOR

ROUTINE MBDE

MBD CALLED

SEARCH MEMORY MAP FOR UST TO BE DEALLOCATED

ADD 0.53 TO EXIT

LET I=HEAD

MBD, A IN IDSTR(I) ME UST LET I=SUC(I)

IF I TO :

RETURN

ELSIF GO TO MBDE

REPLACE UST AND UPDATE MEMORY MAP

ELSE LET SUC(PRED(I))=SUC(I)

IF I = TAIL

LET PRED(SUC(I))=PRED(I)

ELSE LET I TO T-X

IF TAIL=PRED(I)

ELSE AND SHOLE(I)=SJOB(I) TO SHOLE(PRED(I))

LET SUC(I)=AVAIL

LET AVAIL=

IF SJOB(I) GE 2 LET NEWF=1 ELSE

CALL F+

ADD 0.55 TO EXIT

RETURN END
MEMO

VERIFY MEMORY MAP AND CALCULATE CPU STATISTICS

ROUTINE MNV

MNV CALLED

ADD .019 TO EXT

IF MSWITCH FK U PRINT 2 LINES

MEMORY MAP

FLAT LET TOCOR=0
LET T="Ell ?" START SCAN OF MEMORY MAP
"NEXT" AND .013 TO EXIT
IF MSWITCH FK U PRINT 1 LINE WITH JOBS#(PTR(I))

SUCH(I), SPOOL(I) THUS

F22 13 FLSE
F23 14 LET TOCOR=TOCOR+$JOB(I)+$HOLE(I)
F24 15 IF I NE TAIL
F25 16 LET I=SUC(I)
F26 17 GO TO NEXT
F27 18 ELSE LET CORSET=TOCOR
F28 19 ... COMPARE SUM OF MEMORY MAP PROGRAM AND
F29 20 ... CORE SIZE WITH THE CORE SIZE
F30 21 ... "OUT OF MEMORY"

ERROR - MEMORY MAP DOES NOT VERIFY

STOP

FLST LET J=0 LET TUST=UST
FOR FL19+FL22+FL24=J DO 0 ADD 1 TO J
FL47 31 EL= LCF
FL48 32 LET WAITCPU= LET J=0
FOR EVERY CPU OF CPU DO
FL50 34 IF FL19+FL22+FL24=J ADD 1 TO J
FL51 35 ... ADD 1 TO J
FL52 36 EL= LCF
FL53 37 LET ELIGCPU= LET UST=TUST
FL54 38 RETURN END
```
79507 01 08-28-75 17.504 CACI SISSCRIPT II.5 FOR HIS 600/6000 USAF RELEASE 9.

855 1 **
856 2 ** SWAP OUT JOB
857 3 **
858 4 ROUTINE SWOUT
859 5 RDIA

SWOUT CALLED
861 6 LET FL22=0
862 7 LET FL21=1
863 8 ADD 1 TO TSWAP
864 9 IF FL19.XO 1 LET FL19=1
865 10 LET FL34=0
866 11 ELSE CALL SWPLD(1)
867 12 ADD 6,11 TO EXT
868 13 RETURN END
```
869  1  **  SWAP IF JOB
870  2  **

871  3  **
872  4  ROUTINE SWIN
873  5  FD;AG

875  6  LET FL23#1
876  7  CALL SUPP(.)

877  8  ADD 1 TO INCORE
878  9  ADD 0.035 TO EXIT
879 10  RETURN END
PERFORM SWAP (IN AND OUT)

ROUTINE SVFLD(DIR)

SVFLD CALLED

ADD 1 TO LC32

CALL AITCH(7)

ADD 1 TO SFUSE

ADD 0.42 TO EXIT

IF DIR NO SCHEDULE AT TALCC GIVEN UST IN SWAPDUR MS.

FILE THIS TALCC IF THE CCO RETURN

ELSE SCHEDULE TALCC GIVEN UST IN SWAPDUR MS.

ADD LINE TO TEMPH

FILE THIS TALCC IF THE CCO RETURN END
1'' TERMINATE EXECUTION OF SUBSYSTEM
2'' ADD VALUE ACCOUNTING INFORMATION
3 ROUTINE SSFINI
4'' FDIAG
SSFIN CALLED
5 ADD 1 TO TRILL
6 SUBTRACT 1 FROM INCORE
7'' CALL ATCHG(3)
8 CALL NSD
9'' REMOVE JOB FROM CUPE
10 IF THIS UST IS IN CPUQ REMOVE THIS UST FROM CPUQ
11 IF SWITCH NE 1 CALL COPPINT ELSE
12 ELSE IF THIS UST IS IN MSNO REMOVE THIS UST FROM MSNO
13 IF SWITCH NE 1 CALL COPPINT ELSE
14'' KILL ALL REMAINING CORTESY CALL
15 ELSE IF FR19 EQ 1
16 LET MSNO =
17 ADD 1 TO TSNAP ADD 1 TO TASSO ADD 1 TO KOSNAP
18 AD. 1 TO LTR32 ADD 1 TO TSNPK ADD SWAPFUR TO LTR1
19 CANCEL YUE RICCO CALLED 1
20 REMOVE "IF I FROM THE CCR
21 CANCEL THIS MSNO CALLED 1
22 ELSE CALL FACT DESTROY THIS UST
23 ADD 0 TO EXT
24 RETURN END
REMOVE KEY INPUT JOBS FROM THE CPUQ

ROUTINE BUFDMQ

BUFDMQ CALLED

IF INDEP EQ 0 ADD 0.046 TO EXT GO TO BUF3B
ELSE IF UST IS IN CPUQ
REMOVE THIS UST FROM CPUQ
LET TLG=1 LET NPWF=1 FILE THIS UST IN MEMQ
ELSE LET FLG=1
ADD 0.17 TO EXT
IF GOS/TCh NE 0 CALL KOPRINT CALL CUPRINT ELSE
'BUF,3B' CALL KIOSRT
RETURN END
```
START KEY I/O AND SCHEDULE COURTESY CALL FOR INPUT

ROUTINE KIOSRT

KIOSRT CALLED

ADD 0.22 TO EXT

CHECK IF OUTPUT ALREADY IN PROGRESS.
IF SO, DESTROY THAT COURTESY CALL.

LET LOMOUTCC
CANCEL THIS KIOCC CALLED
RETURN THIS 1 FROM THE CCQ

DESTROY THIS KIOCC CALLED 1
ELSE LET FK19EQ1

PROCESS OUTPUT DETAIL

IF IDMORL EQ 0
*KICL,2* LET 1 TO LTG22

IF LTIN LE TIME.V

COLLECT RESPONSE TIME
LET TRESP-TIME.V-LTIN
LET LTIME=LET-RESP
LET LTIME=LET

ELSE ADD EXPONENTIAL.(OUTPUTN,10) TO LTIN

SCHEDULE A KIOCC GIVEN USL IN MAX.(LTIN-TIME.V,0,0)_MS.

LET TIN THIS KIOCC IN THE CCQ

IF LTIN=LTG1
PRINT ' I LIVE WITH JOEKO. TIME.V, LTIN THUS

START OUTPUT AT **********, UNTIL **********, **
ELSE RETURN

PROCESS INPUT DETAIL
ELSE ADD 1 TO LTG22
IF LIII LE TIME.V

COLLECT RESPONSE TIME
LET TRESP-TIME.V-LTIN
LET LTIME=LET-RESP
LET LTIME=LET

ELSE LET TENDUP
SCHEDULE A KIOCC GIVEN USL IN T MS.
IF RATCH NOT FEINT 1 LIVE WITH JOEKO. TIME.V, TIME.V+T THUS
START INPUT AT **********, UNTIL **********, **
ELSE FILE THIS KIOCC IN THE CCQ
RETURN END
```
```
988 1 " " START PGM UST ON ARRIVAL
989 2 " "
990 3 ROUTINE START
991 4 ADD 1 TO TSTR
992 5 " "
993 6 " "
994 7 CALL CORSAMP
995 8 " "
996 9 " " CREATE UST ENTITY AND ASSIGN ATTRIBUTES
997 10 CREATE A UST
998 11 LET USI=517; LET VSIZE=LSIZE
999 12 LET KILL=CPU; LET CPU=KILL LET CHK=FU=KILL
1000 13 LET DICTAT=TMT; LET PRIAT=RINT; LET K-1AT=AINF
1001 14 LET I=VCT+C IF Y GT C
1002 15 LET DICTAT=KILL/IF LET VDICTAT=0.5*DICIA
1003 16 ELSE LET Y=KIN IF X GT 0
1004 17 LET KILL=VILL/IF LET VKILLAT=0.5*KIA
1005 18 ELSE LET Z=POUT IF X GT 0
1006 19 LET KOTAT=VILL/IF LET VKOTAT=0.5*KOTAT
1007 20 LET X=VAX Y=VAX Z=VAX/IF LET Nz=K=0.5*KIA AT LET N=KOUT=0.5*KOTAT
1008 21 LET FL21 AT
1009 22 LET JOB=0 TSTR
1010 23 LET LI=TXT; LET EXP=TIME
1011 24 LET TIPLOT IF LET R-P=1 FILE THIS UST IN "ANO"
1012 25 IF PRINT=8*PRINT GT 0 SHOT,JOFA,C,LSIZE,KILL THIS
START AT *CISION, SIZE=CPU=*
1013 26 ELSE CALL ACHG(-)
1014 27 IF OSVICH FY CALL NPRINT ELSE
1015 28 ADD 0.13 TO *XT
1016 29 "RETURN FND"
```
DEPSSX AT ********** TO ****

```
1024 6 ' ' SEE WHAT WILL INTERRUPT IT.
1025 7 ' ' LET NEXTX=TIME.X
1026 8 LET NXTCC=TIME.C
1027 9 IF CCC IS NOT EMPTY LET NXTCC=TIME.X+TIME.C
1028 10 LET NEXTX=TIME.X
1029 11 LET NEXTX=TIME.X+TIME.C
1030 12 SECH5 1E
1031 13 IF NEXTX<=TIME.X+TIME.C
1032 14 LET NXTCC=TIME.X+TIME.C
1033 15 LET KILL=TIME.X+TIME.C
1034 16 LET NEXTX=TIME.X+TIME.C
1035 17 IF NEXTX<=TIME.X+TIME.C
1036 18 ELSE
1037 19 ' ' SCHEDULE THE INTERRUPTING EVENT
1038 20 IF NEXTX<=TIME.X+TIME.C
1039 21 ELSE IF NEXTX<=TIME.X+TIME.C
1040 22 ELSE IF NEXTX<=TIME.X+TIME.C
1041 23 ELSE IF NEXTX<=TIME.X+TIME.C
1042 24 ELSE IF NEXTX<=TIME.X+TIME.C
1043 25 RETURN END
```

154
ACCUMULATE STATISTICS AFTER SUBSYSTEM TERMINATION

ROUTINE SAQ, SACT CALLED

TALLY SUBSYSTEM STATE TIMES AND COUNTS.

LET LS=SS=LT0
LET LT1=SS=LT1
LET LT2=SS=LT2
LET LT3=SS=LT3
LET LT4=SS=LT4
LET LT5=SS=LT5
LET SSX=LT=24
LET SSOUT=LT=22
LET SSOUT=LT=20
LET SS=AP=LT=31
LET SS=AP=LT=32

CALCULATE RESPONSE TIMES.

IF LT21=LT22 E0 0.
LET LS=LT=ES=LT=19 LT=22 LET LT=21=1 LET RESP=LT=10S
ELSE LET LS=LT=5=LT=1 LET LT=221=LT=22
LET SSOUT=LT=70
IF SSOUT=LT=70 C PRINT 2 LINES WITH TIME,Y,COSMO,SSIN,SSOUT, S237 AT ******** OF ******** KOUT="**
WRITE RESP********, CPU ALLOC********, USE*********
C7 27 ELSE ALLOC FAT

155
1074  1  `'
1075  2  UPDATE SUBSYSTEM STATE TIMES
1076  3  ROUTINE ATCHG(F)
1077  4  DIAG
1078  5  ATCHG CALLED
1080  6  GO TO ATCHL,AT2,AT3,AT4 OR ATS PER LGCH
1081  7  'AT': LET LT=LETN+(TIME,V=LETV) GO TO LT6
1082  8  'AT1': LET LT=LETN+(TIME,V=LETV) GC TO LT6
1083  9  'AT2': LET LT=LETN+(TIME,V=LETV) GO TO LT6
1084 10  'AT3': LET LT=LETN+(TIME,V=LETV) GO TO LT6
1085 11  'AT4': LET LT=LETN+(TIME,V=LETV) GO TO LT6
1086 12  'AT5': LET LT=LETN+(TIME,V=LETV) GO TO LT6
1087 13  'LT5': LET LT=LET
1088 14  LET LT=LET
1089 15  AND 0.11 TO LT
1090 16  RETURN END
7950-07 08-28-75 17.50w  CACI SIGNED 11.5 FOR HIS 600/6000 USAF RELEASE 9.

1091  1 'ENTRY TO ALLOCATOR BY INTERRUPT

1092  2 '***

1093  3 'EVENT EXPR

1094  4 EVENT EXPR

1095  5 CYCLES SHOULDED THUS

1096  6 EXIT AT ******** by ****

1097  7 REMOVE UST FROM CPU

1098  8 LET RELLOC 1 FILE UST LAST IN CPU

1099  9 IF QSWITCH KS & CALL COPPIFY ELSE

1100  9 CALL EXACL

1101 10 SCHEDULE AN ALLOC IN 0.22 MS.

1102 11 """

1103 12 """SAMPLE NEXT INTERRUPT TIME.

1104 13 IF NEXTNF LF 0 LET NEXTINT=EXPOERMNTAL,F(INTELEM,9)

1105 14 ELSE RETURN END
1106   1 "" TAKE ACCOUNTING AFTER INTERRUPT
1107   2 "" EXACT CALLED
1108   3 "" EXACT CALLED
1109   4 ROUTINE
1110   5 RESTART
1111   6 ADD 0.11 TO EXT
1112   7 ""
1113   8 "" ACCUMULATE CPU TIME USED BY SUBSYSTEM AND TSS.
1114   9 ADD 1 TO TIG
1115  10 ADD NEXT TO LSPTS
1116  11 ADD NEXT TO TASS
1117  12 ADD NEXT TO TASK
1118  13 IF TIVS V\--TIVS <= LSPD4 LET NEXT=1
1119  14 ELSE IF TIVS V\--TIVS <= LLE4 LET LSFLG=1
1120  15 ELSE RETURN INC.
1122 1 ** KEY INPUT REFAIL
1123 2 **
1125 4 EVENT KONDIL
1126 5 FVXIAG SHON,JOFNO,THUS
KONDIL AT ********** BY ****
1128 6 LET INDURANT
1129 7 CALL EXACT
1130 8 ADD 1 TO KFIM
1131 9 CALL BUFFER
1132 10 **
1133 11 ** SAMPLE NXT INPUT TIME
1134 12 LET NXTIM=VXITAT LET VXITAT=NXITLIN
1135 13 SCHEDULE AN ALLOC IN 0.2 MS.
1136 14 RETURN END
1137 1 ** KEY OUTPUT DETAIL
1139 3 **
1140 4 EVENT KOTDRL
1141 5 EVISAG SKOL, JOHN THUS
KOTDRL AT ********** BY ****
1143 6 LET KOTDRL=1
1144 7 CALL EXECK
1145 8 ADD 1 TO KEYOUT
1146 9 CALL BUFUN
1147 10
1148 11 ** SAMPLE NXT OUTPUT TIME
1149 12 LET VEXTOUT=VEXT LET VEXTAT=NXTKOUT
1150 13 SCHEDULE AS ALLOC TO C22 KS.
1151 14 RETURN End
1152 1 LET FL18=1
1153 2 CERAIL TO PERFORM DISK I/O
1154 3
1155 4 EVENT DLDELIO
1156 5 ENDING SHOW JOBNO THUS
1157 6 DLDELIO AT ********* BY ****
1158 7 LET FL18=1
1159 8 CALL EXACT
1160 9 ADD 1 TO DISQIO
1161 10 ADD 1 TO LSTQIO
1162 11 LET NXTDIO=DISQIO LET VDIO=VXTRIO
1163 12 FILE REQUEST IN DISQ; IF DISQ WAS EMPTY
1164 13 START TO NOW ADD SCHEDULE DISQ COURTESY CALL.
1165 14 IF DISQ IN E PNT
1166 15 SCHEDULE A DISQGIVEN UST IN EXPONENTIAL,F(RIGHT-10) MS.
1167 16 FILE THIS DISQ IN THE CCQ
1168 17 ELSE FILE THIS UST LAST IN DISQ
1169 18 SCHEDULE AN ALLOC IN 0.35 MS.
1170 19 RETURN END
1172 1 ** DERRAIL TO TERMINATE SUBSYSTEM
1173 2 **
1174 3 **
1175 4 EVENT DERRT
1176 5 CYCLES SHOWN, JOB NO THUS DERRT AT ********** BY ****
1177 6 CALL EXIT
1178 7 **
1179 8 ** CHECK CPU USAGE OF SUBSYSTEM.
1180 9 IF ASELV(CHKCPU,LSPTS) GT 1
1181 10 PRINT 1 LINE WITH JOB NO, CHKCPU, LSPTS THUS
1182 11 ELSE CALL SSINIT
1183 12 SCHEDULE AN ALLOC IN 0.176 MS.
1184 13 RETURN END
162
KEY INPUT AND OUTPUT COURTESY CALLS

1187 1 ..
1188 2 ..
1189 3 ..
1190 4 EVENT: KIACC SAVING THE EVENT NOTICE
1191 5 LET UST=UST: LET UPP=UST
1192 6 EVENT# SHOW, JCC THIS
1193 7 IF KIACC *** THEN CALL CQ PRINT 1 LINE WITH JORNO, TIME, V THUS
1194 8 ELSE LET LTV=TV:V
1195 9 LET FK13=
1196 10 ..
1197 11 ..
1198 12 IF FL17 EQ
1199 13 GO TO <KCOC1
1200 14 ELSE LET FLG=3
1201 15 IF FL21 EQ
1202 16 IF FL22 EQ 1
1203 17 REMOVE THIS UST FROM MEMO
1204 18 LET FLF3=FLF THIS UST FFPST IX CPUQ
1205 19 CALL H-QJ(2)
1206 20 CALL Mprint CALL MPRINT ELSE
1207 21 GO TO <KCOC1
1208 22 ELSE LET UPP=1
1209 23 CALL MALLOC(4)
1210 24 ELSE
1211 25 KIACC, I LET UST=UST REMOVE THIS KIACC FROM THE CCQ
1212 26 DESTROY THIS KIACC
1213 27 RETURN END
DISK I/O COURTESY CALL

EVENT DIQCC SAVING ENDO EVENT NOTICE
LET USE TUST LET USE TUST
EVENT SHOCK, SHOCK TUST
DIQCC AT ******* OF ****
REMOVE FIRST USE FROM DIQCC
LET FL1=1

MOVE HIM TO TOP OF THE CPUSQ.
REMOVE THIS USE FROM THE CPUQ
LET TUPQ=1 FILE THIS USE FIRST IN THE CPUQ
IF SWITCH AT / CALL COMPINT ELSE

LET USE TUST

SEE IF ANY MORE DISK I/O'S ARE WAITING.
REMOVE THIS DIQCC FROM THE CPUSQ
IF DIQCC IS NOT EMPTY
RETURN
RETURN CPUSQ
SWAP OUT COURTESY CALL

EVENT TALCC SAVING THE EVENT NOTICE
LET TUST=UST LET UST=SUST
EVING SHOK,JOENO THUS
TALCC AT *********, OF ****
SUBTRACT 1 FROM INCORE
LET FL19=
LET MWFP=1
... SEE WHAT HE GOT SWAPPED OLT FOR.
IF FL19 EQ 1
IF FL34 EQ 1
CALL ACHG(5)
GO TO TALCC
ELSE CALL ACHG(4)
GO TO TALCC
ELSE ADD 1 TO TALCC
CALL ACHG(3)
TALCC: CALL END
SUBTRACT 1 FROM SPUST
LET UST=UST REMOVE THIS TALCC FROM THE CCO
RETRY THIS TALCC
RETRY END
1264 1 ""
1265 2 "" SWAP IN COURTEST CALL
1266 3 ""
1267 4 EVENT CALLCC SAVING THE EVENT NOTICE
1268 5 LET UST=UST1 LET UST=UST2
1269 6 EVENT SHOW; SHOW THUS
1270 CC AT ******** DE****
1271 7 LET UST=1
1272 8 LET PL=PL
1273 9 LET PL3=PL
1274 10 LET PL3=PL1 FILE THE UST FIRST IN THE CPUQ
1275 11 LET PL3=PL1 FILE THE UST FIRST IN THE CPUQ
1276 12 CALL ARCUT(2)
1277 13 IF PL3=PL1 CALL YPRINT CALL CPRINT ELSE
1278 14 USTEAR ACV CYTE FIRST
1279 15 LET UST=UST RELOVE THIS CALLCC FROM THE CCQ
1280 16 RETDUC THIS CALLCC
1281 17 RETURN AND
1202  1  ""  LINE SERVICE
1203  2  ""
1204  3  ""  EVENT LSHE
1205  4  ""  EVENT HIGH THUS
1206  5  ""  EVENT HIGH THUS
1207  AT  ""  EVENT HIGH
1208  6  ""  EVENT HIGH
1209  7  ""  EVENT HIGH
1210  8  ""  EVENT HIGH
1211  9  ""  EVENT HIGH
1212  10  ""  EVENT HIGH
1213  11  ""  EVENT HIGH
1214  12  ""  EVENT HIGH
1215  13  ""  EVENT HIGH
1216  14  ""  EVENT HIGH
1217  15  ""  EVENT HIGH
1218  16  ""  EVENT HIGH
1219  17  ""  EVENT HIGH
1220  18  ""  EVENT HIGH
1221  19  ""  EVENT HIGH
1222  20  ""  EVENT HIGH
1223  21  ""  EVENT HIGH
1224  22  ""  EVENT HIGH
1225  23  ""  EVENT HIGH
1226  24  ""  EVENT HIGH
1227  25  ""  EVENT HIGH
1228  26  ""  EVENT HIGH
1229  27  ""  EVENT HIGH
1230  28  ""  EVENT HIGH
1231  29  ""  EVENT HIGH
1232  30  ""  EVENT HIGH
1233  31  ""  EVENT HIGH
1234  32  ""  EVENT HIGH
1235  33  ""  EVENT HIGH
1236  34  ""  EVENT HIGH
1237  35  ""  EVENT HIGH
1238  36  ""  EVENT HIGH
1239  37  ""  EVENT HIGH
1240  38  ""  EVENT HIGH
1241  39  ""  EVENT HIGH
1242  40  ""  EVENT HIGH
1243  41  ""  EVENT HIGH
1244  42  ""  EVENT HIGH
1245  43  ""  EVENT HIGH
1246  44  ""  EVENT HIGH
1247  45  ""  EVENT HIGH
1248  46  ""  EVENT HIGH
1249  47  ""  EVENT HIGH
1250  48  ""  EVENT HIGH
1251  49  ""  EVENT HIGH
1252  50  ""  EVENT HIGH
1253  51  ""  EVENT HIGH

167
ADD 0.44 TO EXT

SEE IF ANY NEW SUBSYSTEMS (UST's) NEED TO
BE STARTED.

'MSR301' IF 'XTUST-1: TIME.V CALL START

LET VUST=1: VUST

LET HTUST=XTUST+VUST

GO TO MSR301

ELSE IF TLPL=1: IF GE 1 SCHEDULE AN ALLOCATE EXT MS. RETURN

PACKER TO GOES UNTIL NEXT INTERRUPT.

SUBSYSTEM APPEND TO COURTESY CALL.

IF 1 TO LPLAA LET TAGPT=0

IF TIME.V=A'UT' OF ADD3 LET MPY=1

ELSE LET PCCPRINT.C

IF CCQ IS ICP ENTRY LET K.TCC=TIME.A(F:CCQ)


SCHEDULE AN ALLOCATE AT PAKET+EXT

RETURN PAKET
References


MISSION
of
Rome Air Development Center

RADC is the principal AFSC organization charged with planning and executing the USAF exploratory and advanced development programs for information sciences, intelligence, command, control and communications technology, products and services oriented to the needs of the USAF. Primary RADC mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, and electronic reliability, maintainability and compatibility. RADC has mission responsibility as assigned by AFSC for demonstration and acquisition of selected subsystems and systems in the intelligence, mapping, charting, command, control and communications areas.