Example: Single Server Queue

```
        SIMULATE
        INTEGER         &LIMIT
        LET             &LIMIT=1000
*       Block Statements
        GENERATE        10,5
        QUEUE           LINE
        SEIZE           CHECKOUT
        ADVANCE         7,5
        RELEASE         CHECKOUT
        DEPART          LINE
        TABULATE        RES
        TERMINATE       1
RES     TABLE           M1,5,5,10
*       START           &LIMIT
        END
```

SIMULATE:

- is a control statement specifying that a simulation run is to be run
- it can be placed anywhere in the program
- if not present, the program is compiled and not executed

Ampervariable:

- as with many programming languages, GPSS allows the declaration of variables (called ampervariables as they are preceded by an &)
- ampervariables can be either INTEGER or REAL
- value of ampervariables can be changed using the LET and BLET (block LET) statements

GENERATE:

this blocks creates transactions and sends them into the system at random intervals

- it supports five parameters (of interest to us)

A: specifies the mean length of the generation interval
- can be replaced by functions such as RVEXPO, or RVNORM

B: for a uniform distribution, gives a plus or minus value about the mean

C: offset interval for generation of the first transaction

D: maximum number of transactions to be generated

E: priority of transaction

- example: GENERATE 10,5

- specifies that the inter-creation time of transactions is uniformly distributed between 5 and 15 (10 ± 5) units

QUEUE/DEPART:

- **these are blocks which are simply related to the gathering of statistics**
- **the parameter (in our case LINE), specifies an address of where to store the statistics (could be a number)**
- **QUEUE marks the beginning of the statistics gathering for the block while DEPART marks the end**
- **all transactions entering the QUEUE/DEPART block cause the appropriate data collection to occur**
- **the QUEUE/DEPART block has a zero-delay effect on transactions**

SEIZE/RELEASE:

- **the actual formation of queues (or line-ups) is as a result of the SEIZE/RELEASE block**
- **the SEIZE block governs the admission of transactions to a facility (server)**
- **the name (address) of the facility is given as a parameter (could be a number)**
- **at each instant of time, the facility can be in one of two states (busy or idle)**
- **a transaction can only enter the SEIZE block if the specified facility is idle**
- **if the specified facility is busy, the transaction joins a delay chain (FCFS - within Priority levels)**
- **when the facility becomes idle (as a result of the corresponding RELEASE block), the first transaction on the delay chain enters the SEIZE block**

ADVANCE:

- **the ADVANCE block is the only block in GPSS that can delay a transaction for a specified amount of time**
- **it models the service provided to a transaction**
- **it has two parameter:**
  A: specifies the mean length of holding time
  - can be replaced by functions such as RVEXPO, or RVNORM

  B: for a uniform distribution, gives a plus or minus value about the mean
- **example: ADVANCE 7,5**
- **specifies that the service time of transactions is uniformly distributed between 7 and 12 (7 ± 5) units**

TABULATE/TABLE:

- **the TABULATE block is used to collect histogram data for a particular performance measure**
- **Parameter A of this block contains the address of the TABLE definition**
- **the placement of the TABULATE is used to mark the time**
- **the TABLE block has the following parameters:**
  A: which standard numerical attribute (SNA) to tabulate
  B: upper limit of the first interval
  C: width of each interval
  D: number of intervals
- **TABULATE RES is placed after ADVANCE to consider the entire time in system**

TERMINATE:

- **the terminate block acts as a sink for transactions**
- **the parameter specifies the number of units to be deducted from the *termination count* (specified by the START block) each time a transaction enters the block**
- **if the parameter is left blank, the *termination count* is not changed but the transaction is deleted**
- **when the *termination count* reaches 0, the simulation is terminated**

START:

- **specifies the termination count as a parameter**

TRANSFER:

- the TRANSFER block moves transactions to different parts of the program based an uniform random variate
- TRANSFER prob, label1, label2
  - a uniform random variate is generated and if its value is less than *prob*, the program branches to *label2*; otherwise the program branches to *label1*
  - example: TRANSFER .4, TAB, TEL
    - specifies that a transaction entering this block is to be transferred to statement with labels TAB and TEL with probability 60% and 40% respectively
  - this is a probabilistic transfer and not a conditional transfer (like an IF statement)
  - the TRANSFER block can also be used for an unconditional transfer (i.e. TRANSFER ,label)

ENTER/LEAVE:

- similar to SEIZE/RELEASE except that the service facility may have multiple servers
- first parameter in both the ENTER and LEAVE blocks refers to the address of the storage facility and second indicates the number of servers required
- a transaction is allowed to enter the ENTER block only if the number of servers it requires is less than or equal to the number of servers in that storage facility that are idle
- otherwise, the transaction is blocked and forced to join a delay chain

STORAGE:

- the STORAGE definition statement specifies the total number of servers for each storage facility
- example: STORAGE S1,6/S2,3

End of Simulation:

- a simulation can be terminated after a pre-specified amount of time (instead of after a pre-specified number of customers)
- use two GENERATE blocks: one for customer transactions and the second for the end of simulation transaction
- the TERMINATE block that deletes the customer transactions does not contain a parameter implying the transaction is deleted but the termination count is not decrement
- the second GENERATE block generates a transaction at the end of simulation time
- this transaction immediately enters a TERMINATE 1 block
- the START block specifies a termination count of 1 causing the simulation to terminate

ASSIGN:

- the ASSIGN block is used to modify the value stored in a transaction parameter (local)
- these parameters can be used to store information such as the transaction's arrival time, service time, etc.
- the SNA reference to the jth parameter is Pj
- all transaction parameters are initialized to 0
- variations of the ASSIGN block

  ASSIGN 1,3 stores 3 into P1

  ASSIGN 1+,3 increments P1 by 3

  ASSIGN 1-,3 decrements P1 by 3

  ASSIGN 1,RVEXPO(1,20)
  - stores exponential variate with mean 20 into P1

Indirect Addressing:

- assume transaction parameter P2 contains the numeric address of the server to visit next
- Assume P2 set to 1 prior to entering the QUEUE block
- QUEUE *2 is an example of *indirect addressing* to specify the queue address
- the "*2" in the statement results in the value stored in P2 being used
- thus, the QUEUE block updates the statistics for Queue 1
- indirect addressing can be used in the SEIZE, DEPART, ENTER, LEAVE, ADVANCE, GENERATE and RELEASE blocks (to name a few)

Savevalues:

- storage locations accessible to all transactions (global)
- SNA for the jth savevalue is Xj
- the INITIAL statement is used to initialize the value of the savevalues
- ADVANCE RVEXPO(1,X*2) generates an exponential variate based on the service time stored in savevalue stored in P2
- the value of a savevalue can be modified by a SAVEVALUE block
- SAVEVALUE is analogous to the ASSIGN block in terms of modifying the value of a transaction parameter

  SAVEVALUE 1,3 stores 3 into X1

  SAVEVALUE 1+,3 increments X1 by 3

  SAVEVALUE 1-,3 decrements X1 by 3

  SAVEVALUE 1,RVEXPO(1,20)
  - stores exponential variate with mean 20 into X1

LOOP:

- the LOOP block implements a do-loop
- Example: LOOP 1,NEXT
- when a transaction enters the LOOP block, the contents Parameter A (P1) is decremented by 1
- if P1 > 0, the transaction is moved to the block specified in Parameter B (NEXT)
- otherwise the transaction is moved to the next sequential block

TEST:

- like an IF statement
- the condition is tested and if it is met, the transaction moves to the next sequential block
- if the condition is not met, it moves to the block specified in Parameter C
- example, TEST G P1,20,LAST
  - compares transaction parameter P1 to 20 and if it is greater than (G), it moves to the next block
  - if P1 is less than or equal to 20, the transaction moves to the block with label LAST
- the other TEST operators are GE, E, NE, LE, and L

Variables:

- **variables are used to compute arithmetic expressions**
- **the SNA for the jth variable is Vj**
- **ASSIGN is used to assign a value to a transaction parameter**
- **example: ASSIGN 2,V1**
- **this would compute the value of V1 and assign it to transaction parameter P2**
- **associated with each variable is a definition statement which specifies a arithmetic expression composed of SNA's**
- **example: 1 VARIABLE P2-20)**
- **when a variable is referenced (such as in ASSIGN 2,V1), the corresponding arithmetic expression is evaluated**

Other GPSS Features:

MARK Block:

- we have already introduced the SNA M1 which is the transit time of a transaction
- M1 is defined to be current time - mark time
- mark time is initially given by creation time
- mark time can be changed by sending a transaction through a MARK block which will set mark time to the current clock

Other GPSS Features:

SELECT Block:

- **this block is used to find a GPSS entity (e.g. queue, facility, storage, etc.) that satisfies a specific condition**
- **it can be used with three types of operations:**

1) Logical

U - facility in use	NU - facility not in use
SE - storage empty	SNE - storage not empty
SF - storage full	SNF - storage not full

1) Conditional

L, LE, E, NE, GE, G

3) Special

MIN, MAX

Examples:

SELECT NU 5,1,3,,,NEXT

- **facilities 1 to 3 are tested and the address of the first not in use facility is stored in P5**
- **the transaction is moved to the next sequential block**
- **if all facilities used, transaction moved to label NEXT**

SELECT E 5,1,3,X1,Q,NEXT

- **length of queues 1 to 3 are tested and the address of the first queue with length equal to X1 is stored in P5**
- **the transaction is moved to the next sequential block**
- **if no queue with length = X1 is found, transaction moved to label NEXT**

SELECT MIN 5,1,3,,Q

- **queues 1 to 3 are examined and address of the queue with minimum length is stored in P5**