NASA CONTRACTOR
REPORT

NASA CR-128997

OPTIMUM SPACEBORNE COMPUTER SYSTEM DESIGN BY SIMULATION

By T. Williams, J. E. Weatherbee, and
D. S. Taylor

Computer Sciences Corporation
Field Service Division
Aerospace Systems Center
8300 S. Whitesburg Drive
Huntsville, Alabama 35802

September 1, 1972

Prepared for

NASA-GEORGE C. MARSHALL SPACE FLIGHT CENTER
Marshall Space Flight Center, Alabama 35812

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO.	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
NASA CR-128997		

4. TITLE AND SUBTITLE	5. REPORT DATE
Optimum Spaceborne Computer System Design By Simulation	September 1, 1972
	6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT #
Dr. T. Williams, Dr. J.E. Weatherbee, and Dr. D.S. Taylor	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Sciences Corporation Field Services Division, Aerospace Systems Center 8300 S. Whitesburg Drive Huntsville, Alabama 35802	10. WORK UNIT NO.
	11. CONTRACT OR GRANT NO. NAS8-21805
12. SPONSORING AGENCY NAME AND ADDRESS National Aeronautics and Space Administration Washington, D.C. 20546	13. TYPE OF REPORT & PERIOD COVERED Contractor Report
	14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

16. ABSTRACT

A deterministic digital simulation model is described which models the Automatically Reconfigurable Modular Multiprocessor System (ARMMS), a candidate computer system for future manned and unmanned space missions. Use of the model as a tool in configuring a minimum computer system for a typical mission is demonstrated, using as an application mission the proposed Reusable Shuttle Booster (RSB) stage. While ARMMS is not a candidate for the RSB computer, this application was chosen as a vehicle for demonstrating the simulator because of the availability of a detailed description of the RSB data processing workload.

The configuration which is developed as a result of studies with the simulator is optimal with respect to the efficient use of computer system resources, i.e., the configuration derived is a minimal one. Other considerations such as increased reliability through the use of standby spares would be taken into account in the definition of a practical system for a given mission.

17. KEY WORDS	18. DISTRIBUTION STATEMENT
Configuration Modular Multiprocessor Optimum Reconfigurable Simulation	B. Hodges Bobby Hodges Computer Systems Division MSFC/Computation Laboratory Unclassified-Unlimited

19. SECURITY CLASSIF. (of this report)	20. SECURITY CLASSIF. (of this page)	21. NO. OF PAGES	22. PRICE
Unclassified	Unclassified	37	NTIS

MSFC - Form 3292 (May 1969)

Table of Contents

ACKNOWLEDGEMENT

The authors wish to acknowledge the project monitor, Mr. B. Hodges (S&E-COMP-C) for technical contributions and liaison work with Astrionics Laboratory and also his helpful discussions throughout the duration of this project.

Thanks is also due to Dr. J.B. White (S&E-ASTR-CA) for his continued cooperation and encouragement during the course of this work.

SUMMARY

A deterministic simulator is described which models the Automatically Reconfigurable Modular Multiprocessor System (ARMMS), a candidate computer system for future manned and unmanned space missions. Its use as a tool to study and determine the minimum computer system configuration necessary to satisfy the on-board computational requirements of a typical mission is presented.

The effectiveness of a simulation model of a computer system as a design tool depends on the accuracy and fidelity of the definition of the specific data processing load. Hence, the simulation techniques are described in relation to a specific spacecraft performing a specific mission, namely the Resusable Shuttle Booster (RSB) stage which had been considered in the early phases of the shuttle program as the first stage of the transport from earth to low orbit.

While ARMMS is not a candidate for the RSB computer, this application was chosen because of the availability of a detailed description of the RSB data processing workload in which the computational requirements of the various RSB subsystems have been identified by mission phase and are used as parametric inputs to the deterministic model of the complete data processing system.

This report describes how the computer system configuration is determined in order to satisfy the data processing demand of the various Shuttle Booster subsystems. Various assumptions have been made which may be unrealistic from the standpoint of the ARMMS presently being designed, but were used in order to define a basic system which could be simulated and allow the capabilities of the model to be demonstrated.

The configuration which is developed as a result of studies with the simulator is optimal with respect to the efficient use of computer system resources, i.e., the configuration derived is a minimum one, other considerations such as increased reliability through standby spares would be taken into account in the definition of a practical system for a given mission.

I. INTRODUCTION

The advent of third generation computing concepts has created problems in the selection or design of a computer to process a given workload. For example, multiprogramming enables the computer system to process more than one task concurrently, allowing more efficient utilization of system resources such as central processing unit and input-output subsystems; this concept, however, has made the selection or design of a computer system no longer the relatively simple task it once was.

A great amount of effort has been expended on this problem of matching a modern computer system to its data processing workload resulting in many different approaches such as simulation, mathematical modeling, etc. (NIELSEN, N.R., 1967), (GAVER, D.P., 1967). In order to accomplish such an optimized match, however, detailed knowledge of both the proposed system and workload is required.

This report presents an approach to the optimization of the Automatically Reconfigurable Modular Multiprocessor System (ARMMS) to a well-defined data processing workload. ARMMS is presently being designed by Astrionics Laboratory of the Marshall Space Flight Center, Huntsville, Alabama, and is addressing the anticipated requirements for both higher computing capacity and reliability which may characterize spaceborne computers in the late 1970's to mid-1980's. ARMMS is intended to achieve both of these objectives through a highly modular computer architecture which can be configured as a multiprocessor for maximum computing speed or as a triple modular redundant (TMR) system with standby spares for extremely high reliability. Moreover, the configuration will be dynamic in that it will be possible to change the configuration in real time as needed by various mission phases or events.

The workload used in this study was that for the Reusable Shuttle Booster (RSB) Stage which had been considered in the early phases of the shuttle program as the first stage of the transport from earth to low orbit. While the ARMMS is not intended for the RSB computer, this application was chosen because of the availability of a detailed description (Univac Report, 1971) of the RSB data processing workload. This combination of the ARMMS system description and the RSB workload description is used to illustrate the optimization procedure through a method of digital simulation.

The RSB data processing workload is typical of most aerospace applications in that it consists of a number or repetitive tasks which iterate at various rates. In the RSB application, however, many of these tasks must operate in a high reliability, i.e., TMR, mode. Hence, the situation exists where such a task will use three CPU's for a few milliseconds and then release them for use by other tasks which may be required to operate in the simplex mode, i.e,, single CPU, for a further few milliseconds. These tasks will update system data elements, which in turn will be used by other executing program modules.

In an environment of such complexity, simulation of the system is imperative in order to determine the values of system resources such as number of CPU's, width of data busses, number of input/output channels, etc.

The remainder of the report describes such a simulation model together with the results of a series of simulation runs in which various system parameter values were established for the ARMMS system in its hypothetical role as the on-board computer for the Reusable Shuttle Booster.

II. SYSTEM DESCRIPTION

A. ARMMS Baseline

The basic ARMMS configuration is shown in Figure 1. It consists of a Central Processing Element (CPE) composed of a number of central processing units (CPU's) which execute programs contained in a random access memory (RAM) backed up by bulk storage. External subsystems communicate with the computer via the Input/Output Element (IOE). Various data busses interconnect the system modules, as shown in Figure 1.

The system operates under the control of a dedicated executive module, the Block Organizer and System Scheduler (BOSS).

The number of CPU's, size of RAM, number of input/output processors and the widths of the various data busses will be dictated by the data processing and reliability requirements of each particular mission.

B. Workload Description

A detailed data processing workload analysis is essential in order to perform a realistic design of the associated data processing system. Such an analysis has been performed for the Reusable Shuttle Booster (Univac Report, 1971) and was used to define the requirements of an on-board computing system. This workload description consists of the identification of a number of discrete mission time-phases during which specific program modules are executed and interact with specific data elements. Each program module is defined in terms of the amount of memory required, the number and type of instructions executed, the input and output data rates and the program iteration rate. The program modules interact with a number of data elements, each of which is defined by its size, the update iteration rate,

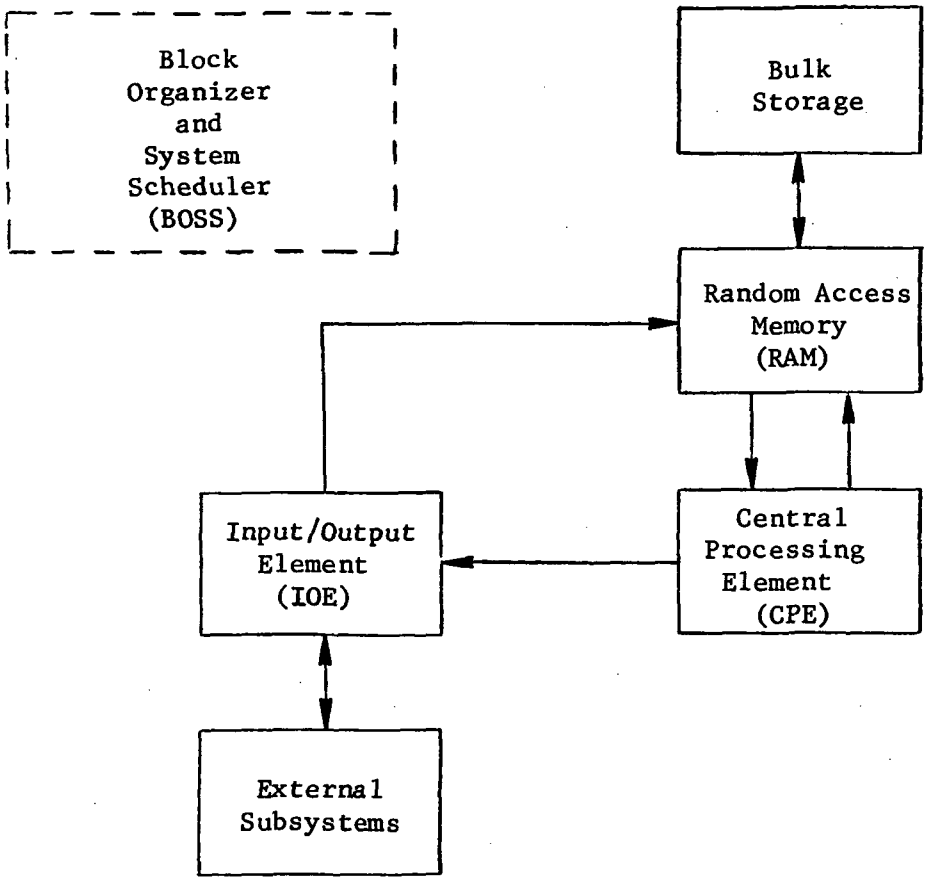


Figure 1. Basic System Configuration

and function, i.e., whether it is a source of or a destination for data associated with the program modules and external subsystems.

Shuttle Booster Mission Phases

The various phases of the Shuttle Booster Stage have been identified and are shown in Figure 2. Note that this timeline refers to a reusable booster stage which is intended to re-enter the earth's atmosphere after each launch, then cruise and land like a conventional aircraft.

Program Module Description

A sample program module description is shown in Figure 3. The program module described is called BIGP (Boost Iterative Guidance Program) and is defined in terms of its total number of instructions, the number of long and short instructions executed in a normal iteration, the required data space, the number of times the program executes per second, the reliability requirement, and the mission phases during which the program executes.

This description allows an exact allocation of memory space to be made to each module, consisting of an instruction area and a data area. Also, knowing the CPU execution speed, the amount of CPU time per iteration may be found directly from the total number of instructions executed per iteration.

The reliability requirement determines whether one or three CPU's are required each time the program executes.

Data Element Description

A data element is defined by the number of times it is used per second, its size in bits, its sources of updated data and the program elements and/or subsystems which use its data.

0	BEGINNING OF PRELAUNCH CHECKOUT
1	START UP OF THE VEHICLE ELECTRICAL GENERATORS
2	INTERNAL ELECTRICAL POWER ACHIEVED
9	NEAR END OF PRELAUNCH CHECKOUT (GO-INERTIAL)
10	BEGINNING OF LAUNCH
11	INITIATION OF PITCHOVER
18	END OF BOOST SIGNIFICANT ATMOSPHERE
19	BEGINNING OF THRUST TERMINATION
20	BEGINNING OF COAST
30	BEGINNING OF REENTRY
32	ENTRY INTO SIGNIFICANT ATMOSPHERE
34	END OF SURVIVAL PHASE
40	BEGINNING OF CRUISE
50	BEGINNING OF LANDING
51	INITIATE LOWER LANDING GEAR
52	LANDING GEAR DOWN
60	TOUCHDOWN
61	END OF TAXI
70	FERRY TAKE OFF
71	LANDING GEAR UP
80	BEGINNING OF BOOST ABORT
81	END OF BOOST ABORT
82	PILOT INITIATION
83	PROGRAM COMPLETION

Figure 2. Shuttle Booster Mission Phases

PROGRAM NAME	BIGP
TOTAL INSTRUCTIONS SHORT INSTRUCTIONS EXECUTED LONG INSTRUCTIONS EXECUTED	707 1344 376
CONSTANT (24-32 BIT) CONSTANT (12-16 BIT) VARIABLE (24-32 BIT) VARIABLE (12-16 BIT)	0 21 8 60
ITERATIONS/SEC	2
REDUNDANCY	YES
INITIAL SCHEDULING PHASE DESCHEDULING PHASE	18 20

Figure 3. Sample Program Module

A number of such data elements are described in Figure 4: for example, ABEFU (Accelerometer Bias Error Functions) and BPAYAC (Boost Pitch and Yaw Attitude Commands) are both single source/single destination; ABEFU being updated by the Navigational Subsystem (NAV) and used by program module SDSU (Strapdown Start Up Program), while BPAYAC is updated by program module BIGP (Boost Iterative Guidance Program) and is used by the Flight Control Subsystem (FCS). Other examples are shown in Figure 4 for single source/multiple destination, multiple source/single destination and multiple source/multiple destination elements.

Data Base Description

The above information concerning program modules and data elements is combined into a single data base entry for each executing program. A sample entry for BIGP is shown in Figure 5.

DATA ELEMENT	ITERATION RATE	SIZE (BITS)	* UPDATED BY	* USED BY
ABEFU	16	576	NAV	SDSU
BPAYAC	2	32	BIPG	FCS
INEVAP	32	192	SD64	SD32 SD16 SD02 BGFP COGP MPST
AILSDO	1	4	AILC AILM	NAV
OCTRUL	4	7	EGSP TCMP TGSP	EGS EGCP

Figure 4. Sample Data Element Description

* Four letter acronyms are program modules; three letter acronyms are subsystems.

PROGRAM NAME	IBANK	DBANK	SINST	LINST	PRATE	TMR	INPUT	IRATE	OUTPUT	ORATE
BIGP	707	97	1344	376	2	1	INEVAP TPTNCT	32 2	BPAYAC	2

Figure 5. Sample Data Base Entry

III. SIMULATOR DESCRIPTION

A. General Description

The simulator to be described allows the interaction of the various program modules, data elements and external subsystems to be analyzed. A portion of this complex interaction is shown in Figure 6, where the basic ARMMS system of Figure 1 has had superimposed upon it some of the interacting program modules and data elements described above. Figure 6 shows the program module BIGP in execution at a rate of 1344 short and 376 long instructions two times per second. BIGP requires 707 words of instruction space and 97 words of data space. It requires updated information from data elements INEVAP and TPTNCT which are themselves updated at rates of 32 and 2 per second respectively. BIGP supplies information to data element BPAYAC, which is used by the Flight Control Subsystem (FCS).

Figure 6 represents a snapshot of an instant in time; when BIGP finishes a particular iteration it releases its CPU for use by another program module. Should any program module fail to iterate at its required rate, due say to priority conflicts with other modules, the simulator will flag the event as real time violation. It is the object of the simulation to have all program modules and data elements executing at their predefined rates within a minimum system under the constraint of no real time violations.

B. System Simulator

The system simulator is a program which implements the functions described in Figure 6. Its flowchart is shown in Figure 7.

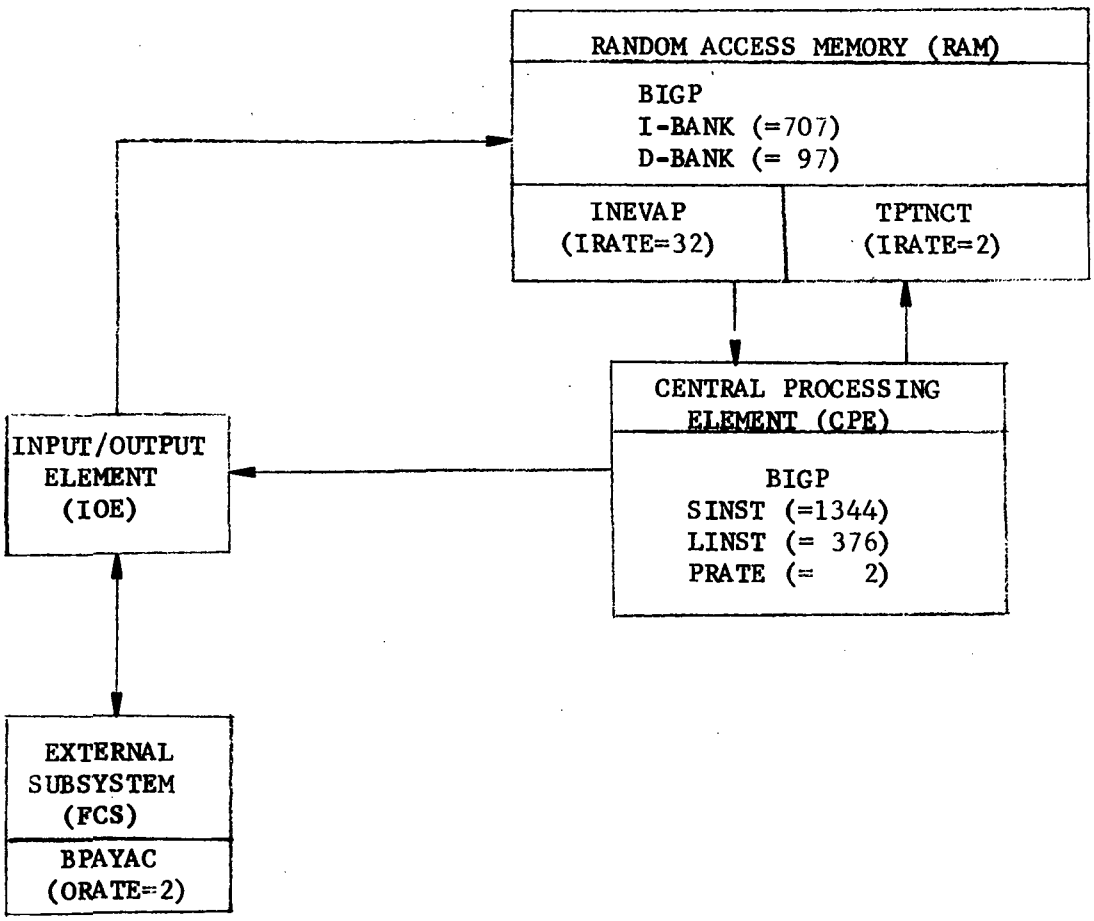


Figure 6. Portion of ARMMS Baseline Data Flow Simulation

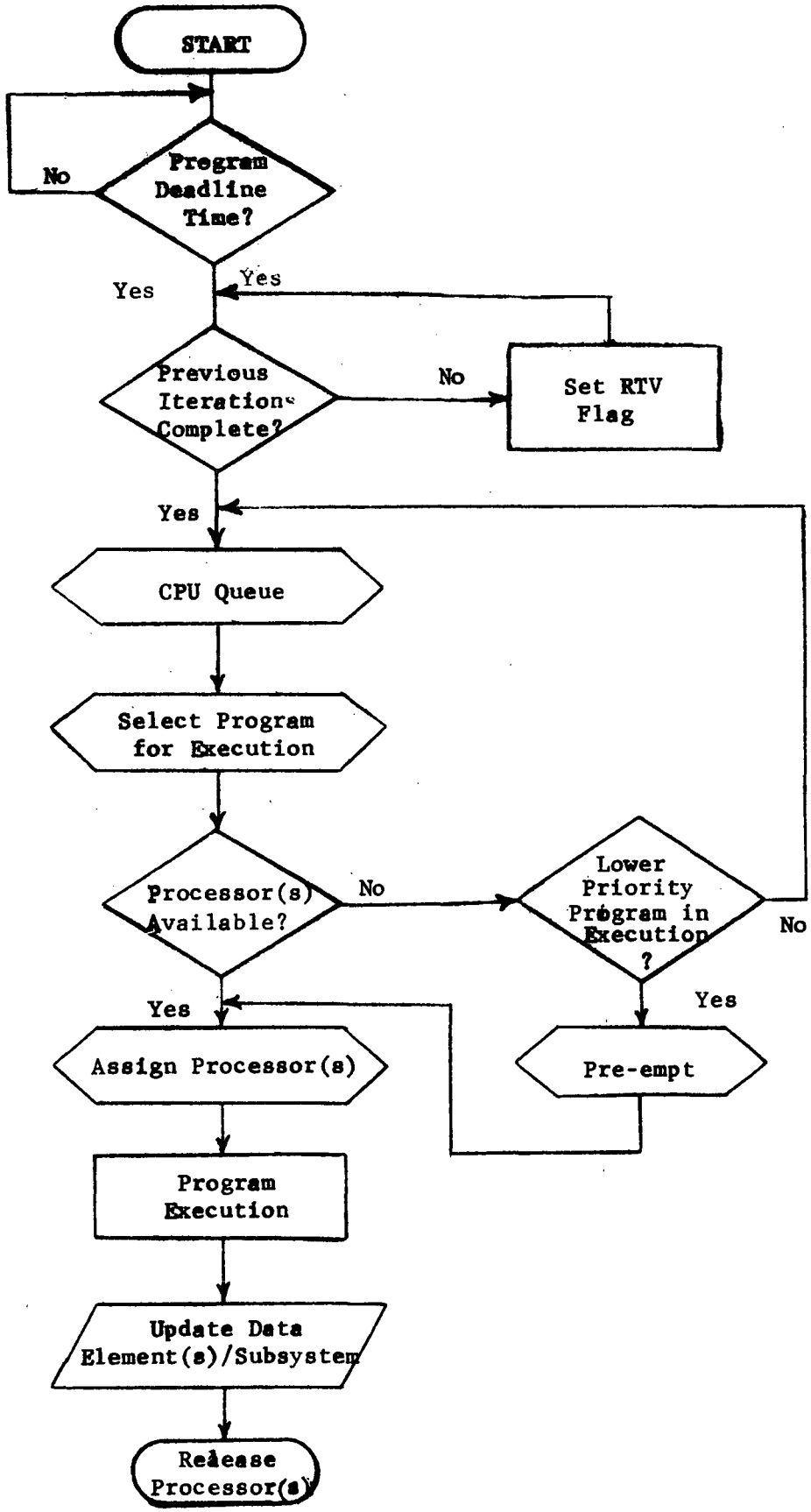


Figure 7. System Simulator Flowchart

Each program has a pre-assigned priority based upon its iteration rate and reliability requirement. The priority assigned to each program is given in Table 1, where a high figure implies a high priority; the letters T and S refer to the TMR and simplex modes of operation, e.g., 16/T describes a program which iterates at a rate of 16 times per second in the TMR mode.

Itn. Rate/Mode	64/T	64/S	32/T	32/S	16/T	16/S	8/T	8/S	4/T	4/S	2/T	2/S	1/T	1/S
Priority	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Table 1. Program Priority Scheme

From the iteration rates shown, it is obvious that each program has a deadline time which must be met in order to avoid a real-time violation. Hence for no real-time violations, programs having priorities 13 or 14 must execute every 1/64 sec., those with priorities 11 or 12 every 1/32 sec., etc.

When a program's deadline time arrives, a check is made to see if the previous iteration of that program is complete: if it is, the program is entered in the CPU queue; if not, a real-time violation (RTV) flag is set. Program modules for which the RTV flag is set are denied entry into the CPU queue until the previous iteration is complete.

Programs are selected from the CPU queue in order of their priority, higher priority programs having the ability to preempt executing programs of lower priority. Execution takes place for a time determined by the number of long and short instructions to be executed per iteration. This execution allows data elements and/or subsystems to be supplied with updated data.

After each execution iteration, the processor(s) are released for use by other programs of lower priority which are currently resident in the CPU queue.

C. Interfaee Simulator

As was stated in Section III.B, the duration of each iteration cycle of each executing program moduel is determined by the number of short and long instructions executed per iteration. Examples of short instructions are ADD, SHIFT, JUMP; while MULTIPLY, DIVIDE are typical long instruction types. The execution times for each instruction is specified for the ARMMS CPE and hence, by assuming a Gibson mix (SMITH, J. M., 1968) for all programs, an average execution time may be determined for both long and short instructions. Referring to Figure 1, the basic ARMMS configuration, which represents a system having a number of CPU's interacting with a random access memory consisting of a number of modules, a situation can arise where more than one CPU will try to access a given memory module. When this occurs, interference takes place with a resultant delay in the execution of one of the conflicting programs. In an environment such as this, therefore, it is no longer valid to determine the execution time of any given program on the basis of average instruction execution time alone.

In order to take the effect of memory interference into account, it was necessary to simulate the interference at the CPE/RAM interface. This was done by means of the Interface Simulator.

The flowchart of Figure 8 illustrates the function of this simulator: it is initialized by specifying all input parameters such as number of CPU's, number of memory modules, data bus widths, number of instructions fetched per access, etc.

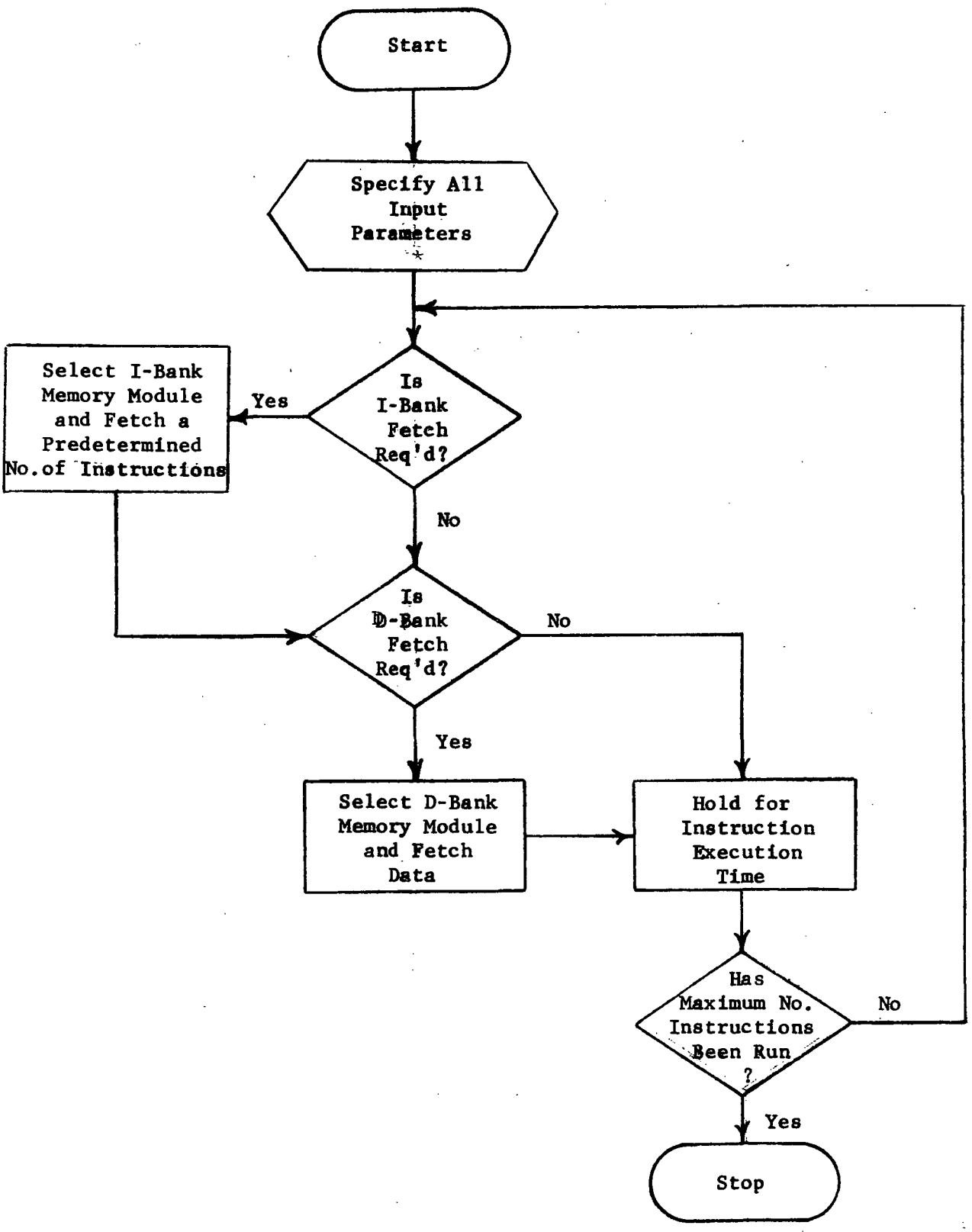


Figure 8. Interface Simulator Flowchart

The Interface Simulator operates under the assumption that there is always a non-zero set of instructions awaiting execution by each CPU; this represents a worst case condition. Note from Figure 8 that an instruction fetch and data fetch are not initiated for all instructions executed: for example, an instruction fetch is not always necessary if a multiple instruction fetch is incorporated in the simulated CPE; similarly, a data fetch is not required for every instruction executed, e.g., JUMP. Values of functions defining the fetch/no fetch ratios are supplied as inputs to the Interface Simulator. This fetch/no fetch ratio is a function of the system, being simulated and the data fetch/no fetch ratio is a function of the instruction mix; in the present simulation, a Gibson mix was assumed.

The Interface Simulator computes the total execution time for both long and short instructions by simulating the execution of a large number of instructions. This number, which is supplied to the simulator as input, is presently set at 3000, giving a variance of less than 1 percent on the computed times.

IV. SYSTEM SIMULATION

A. Introduction

This section describes how the simulation models were used to determine values for those parameters necessary to specify the design of a minimum computer system capable of performing the data processing for all mission phases of the Reusable Shuttle Booster.

In the work to be described, certain system parameter values were fixed due to constraints other than data processing throughput; in addition, certain assumptions were made. These are:

(a) The maximum number of tasks which can process concurrently is three. This figure was derived from consideration of the load placed upon the executive module, BOSS.

(b) There are two one-way busses for communication from the CPE to RAM, and another two connecting RAM to the CPE. These busses are time shared by the three instruction streams assumed under (a); two busses were considered necessary for the purpose of reliability.

(c) The processing speed of the CPE is fixed: This arises due to the fact that ARMMS is an extension of the Space Ultrareliable Modular Computer (SUMC) which is presently under development at MSFC, and it is anticipated that a version of SUMC will be the processor module of the ARMMS system.

(d) The RAM cycle time is assumed to be 750 n.sec: This is available using present-day passive memory technology and is close to the memory speed proposed for ARMMS. Destructive readout is assumed.

(e) There is no triplication of stored data for the TMR operation: All memory transfers will be parity checked and the data fanned out into three CPU's where TMR is required.

(f) A single address instruction format is assumed; in general, one operand is fetched from RAM and the other from a bank of general registers located in each CPU.

(g) The bus transfer time is 100 n.sec.

(h) The time taken to assign a task to a processor or pre-empt an operating task is 100 μ sec.

(i) The basic machine word length is 32 bits.

The simulation models were then used to determine:

(a) The effect of multiple instruction fetches per memory access;

(b) The number of CPU's;

(c) The number of RAM modules;

(d) The widths of the system busses.

The approach taken was to inspect the workloads for each mission phase and select the one which appeared to place greatest demands upon the computer system. Mission Phase 18 appeared to be the most stringent in its requirements and was selected as the workload on which the initial investigations were performed. This phase was characterized by the requirement that all tasks were required to operate in the TMR mode, constraining the system to operate with a multiple of three CPU's.

B. Determination of Number of CPU's

Since the actual average long and short instruction execution times are functions of the system configuration and the resulting memory/CPE interface conflicts, a set of runs was made in which the execution times were varied over a wide range and any real-time violations noted. Figure 9 shows the results of such a set of runs as simulated for three CPU's; it can be seen that, if the long instruction execution time exceeds 5 μ sec for a 1 μ sec short instruction execution time, or 3 μ sec for a 1.5 μ sec short instruction execution time, real time violations occur. The corresponding times for the SUMC processing unit are known to be of the order of 6 and 2 μ sec, respectively, and, hence, a single TMR processor set is inadequate for the task of processing the Mission Phase 18 workload.

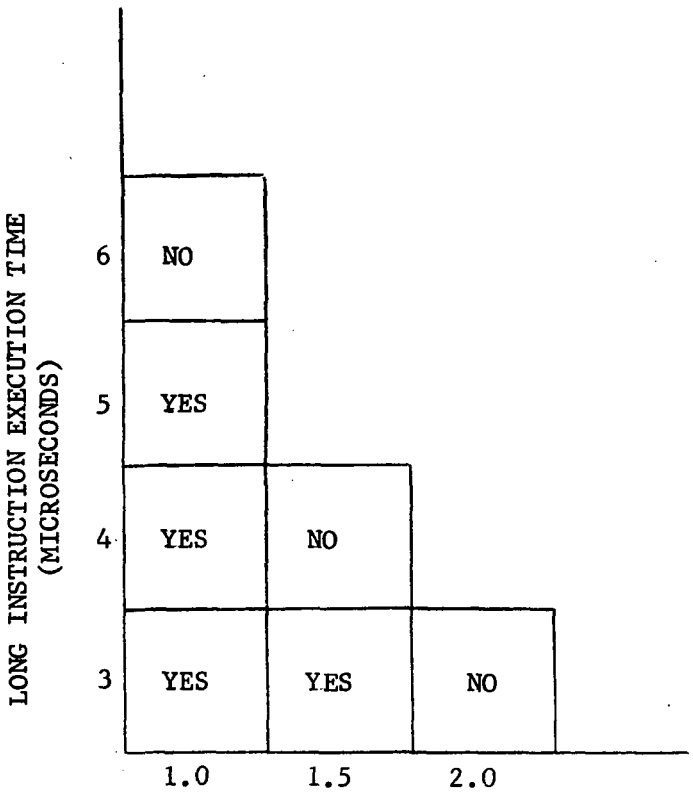
Figure 10 shows a similar chart for six CPU's, i.e., two TMR sets. It can be seen that this configuration will readily process the Mission Phase 18 workload under the constraint of the SUMC processing times.

Note that these tests were carried out on a system having a single input/output subsystem and a single bit CPE-IOE bus width. These parameter values were chosen after it was established that they imposed no constraints on the system due to the relatively light amount of I/O activity to and from the external subsystems.

Hence, it is concluded that the ARMMS system will require six CPU's in order to meet the data processing and reliability requirements of Mission Phase 18.

Having established the need for six CPU's, the Interface Simulator was then used to determine the effect upon instruction execution times of

(a) varying the number of RAM modules



YES - All program deadlines met
NO - Some program deadlines not met

Figure 9. Results from System Traffic Model Runs with 3 CPU's

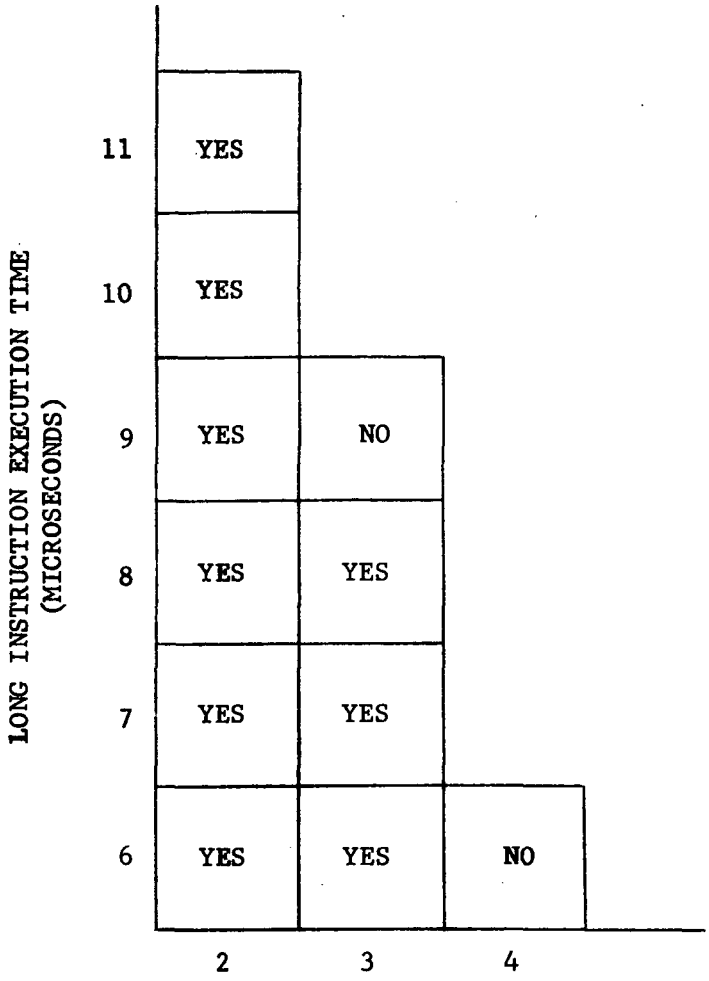


Figure 10. Results from System Traffic Model Runs with 6 CPU's

(b) multiple instruction fetch

(c) variation of the width of the RAM/CPE busses

All these tests were performed with relation to Mission Phase 18 in that two instruction streams were simulated representing the two TMR sets required by that Mission Phase.

Also, an analysis of the proposed instruction set as mapped into a Gibson mix revealed that 16 percent of all instructions required no D-bank access, and this figure was used in all experiments performed.

The RAM/CPE busses were assumed to be 32 bits wide.

C. Determination of Number of Memory Modules

A set of tests was carried out on the Interface Simulator to determine the effect, upon instruction execution times, of varying the number of RAM modules assigned separately to the storage of instructions and data. Table 2 shows the instruction execution times obtained from these tests as a function of the number of I-bank and D-bank modules.

Module Size (K Words)	8	4	2
No. I-Bank Modules	4	8	16
No. D-Bank Modules	8	16	32
SIT (μsec)	2.02	2.01	1.99
LIT (μsec)	6.20	6.19	6.16

SIT = Short Instruction Time
LIT = Long Instruction Time

Table 2. Instruction Execution Time as a Function of Number of Memory Modules

It can be seen from Table 2 that the amount of memory interference is small when greater than 12 memory modules are used. The total memory requirement for Mission Phase 18 is approximately 32K of I-bank and 64K of D-bank; hence, Table 2 represents the effects of varying module size from 8K to 2K.

From the standpoint of reliability, it is unlikely that an 8K module would be used since it represents too large a "throwaway" module. Hence, the 4K module was chosen and used in all further studies to be described.

D. Multiple Instruction Fetch

The design contractor of the ARMMS system (Hughes Aircraft Company) has recommended the concept of a small local store within each CPU. A major function of such a local store would be to retain a small number of previously executed instructions so that, should a branch backwards within this bound occur, an instruction fetch from main memory is not required. After a detailed analysis of an extensive set of aerospace programs, Hughes state that a saving of 4 percent in the number of instructions accessed in main memory is achieved for an eight instruction retention capability. While this has a negligible effect upon instruction execution times, an eight word instruction retention capability will handle approximately 35 percent of all branch instructions (Hughes Report, 1972). The remaining 65 percent of the branch instructions are jump-ahead type, and were assumed to represent 10 percent of all other instructions.

These figures were used as the basis for a set of experiments to determine the effect of multiple instruction fetch upon instruction execution times. Table 3 shows the results of these experiments for 2, 4 and 8 instructions fetched per memory access.

Instructions per Fetch	1	2	4	8
SIT (μsec)	2.01	1.63	1.40	1.29
LIT (μsec)	6.19	5.75	5.57	5.45

SIT = Short Instruction Time
LIT = Long Instruction Time

Table 3. Instruction Execution Time as a Function of Number of Instructions per Fetch

The greatest percentage improvement is observed in going from one to two instructions per fetch, and this improvement was used in the next set of experiments as a trade-off against reduction in RAM/CPE bus widths.

Note that a two-instruction fetch implies reading a double word from memory on each memory access.

E. Determination of RAM/CPE Bus Widths

From Figure 10 it can be seen that the 6 CPU configuration can meet all real-time requirements with execution times of up to 3 μsec for short instructions and up to 8 μsec for long instructions. Table 3 shows that the two instruction fetch produces execution times of 1.63 μsec and 5.75 μsec, respectively, for short and long instructions when using 32-bit-wide RAM/CPE busses. Instruction execution speed is traded off against RAM/CPE bus widths in Table 4 which summarizes the results of a set of tests where the RAM/CPE bus widths were held equal but reduced from the 32-bit baseline value down to a width of 4 bits.

RAM/CPE Bus Widths (bits)	32	16	8	4
SIT (μ sec)	1.63	1.91	2.48	3.63
LIT (μ sec)	5.75	6.03	6.62	7.68

SIT = Short Instruction Time
LIT = Long Instruction Time

Table 4. Instruction Execution Time as a Function of RAM/CPE Bus Widths

These results show that the RAM/CPE busses can be reduced to a width of 8 bits and still satisfy the real-time requirements of Mission Phase 18.

Hence, based upon the computational requirements of Mission Phase 18 together with the constraints stated in Section IV.A, the optimum configuration of the ARMMS system is

- 6 CPU's
- 8 x 4K RAM modules for instruction storage
- 16 x 4K RAM modules for data storage
- 2 instruction fetch per memory access
- 8 bit RAM/CPE busses
- single Input/Output Element (IOE)
- single bit CPE/IOE bus

This configuration is shown graphically in Figure 11. The effects upon instruction execution time of the parameter variations performed to arrive at this configuration are illustrated in Figure 12.

The above configuration was then used in the simulation of the data processing required by Mission Phase 9, the one assessed to be the next most demanding upon data processing resources after Phase 18: Phase 9 is characterized by having mixed mode execution, i.e., both simplex and TMR.

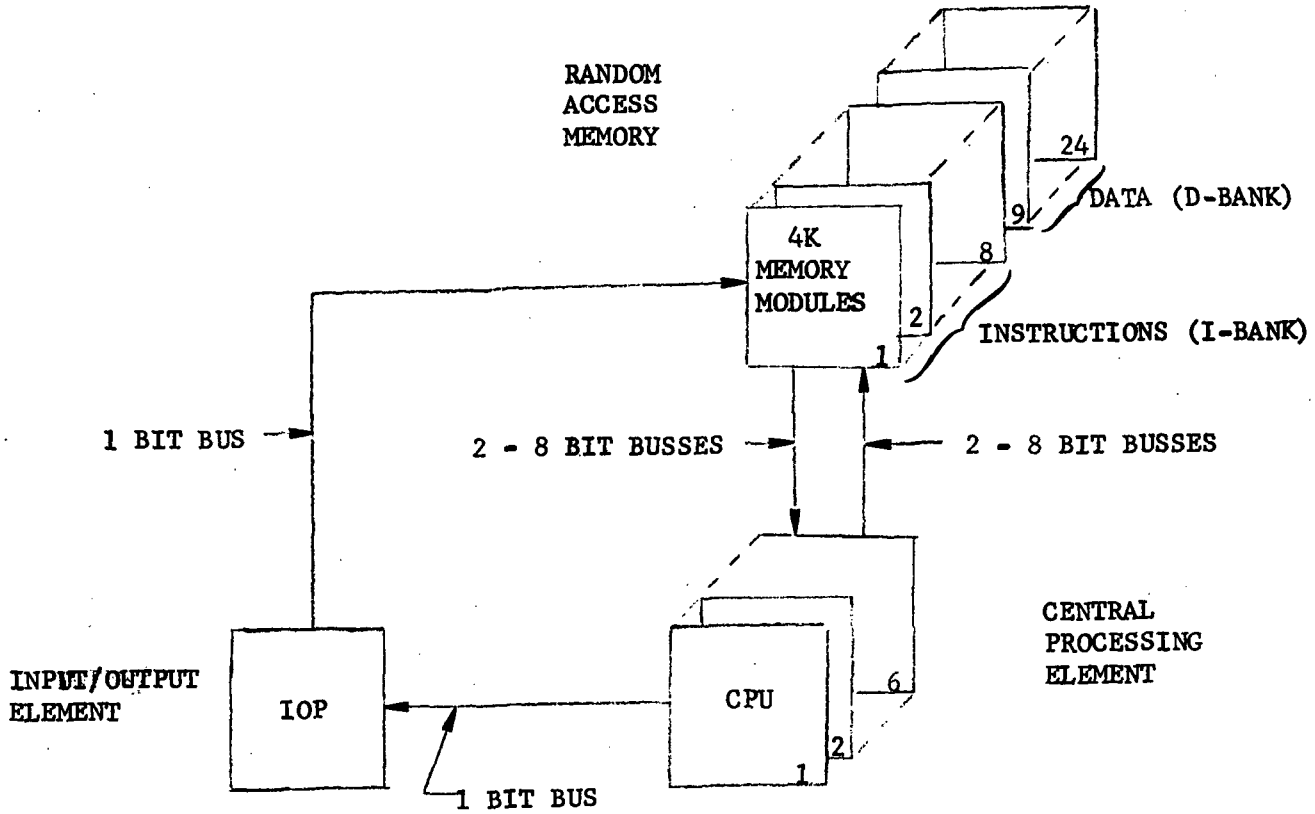


Figure 11. Optimum Configuration

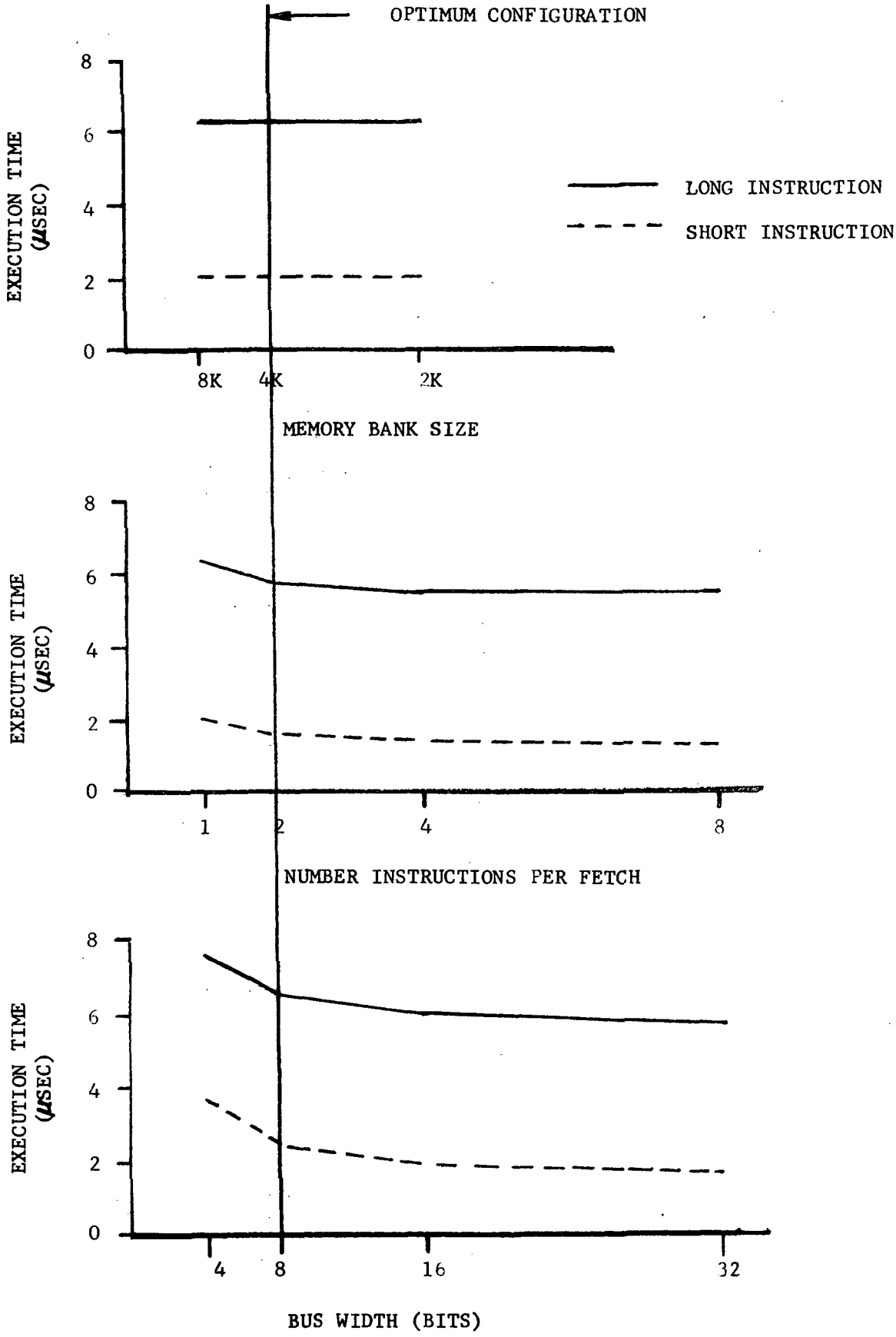


Figure 12. Instruction Execution Time as a Function of System Parameters

The instruction execution times produced by the Interface Simulator were 2.53 μsec and 6.73 μsec, respectively, for short and long instructions. These times were derived under worst case conditions where a three instruction stream environment was assumed for the whole of Mission Phase 9. Since this Phase is mixed mode, two instruction stream operation will occur part of the time when two TMR programs are executing concurrently.

Using the above instruction execution times, all Phase 9 programs were simulated without producing any real-time violations.

The simulation of Mission Phase 9 was repeated with the same configuration, but this time under the assumption that all programs executed in the TMR mode. Instruction execution times of 2.48 and 6.62 μsec, respectively, were used as derived for the case of two instruction streams. Again, no real-time violations occurred.

Checks on several other Mission Phases revealed no real-time violations when using the two TMR set configuration; hence, for the Reusable Shuttle Booster application, all programs should execute in the TMR mode, thereby relieving the Executive Module BOSS of the task of scheduling for a mixed mode operation.

V. CONCLUSION

A deterministic simulator has been described and its application as a design tool illustrated by the hypothetical example of defining a minimum Automatically Reconfigurable Modular Multiprocessor System (ARMMS) which can process the data pertaining to the Reusable Shuttle Booster mission.

It has proved possible to define the required ARMMS hardware in terms of

(a) number of CPU's;

(b) number of RAM modules; and

(c) width of the system busses.

This definition was based upon a data processing load description which was broken down into a number of mission phases, each mission phase being defined in terms of the executing program modules and associated data elements. The program module description consisted of the total number of instructions, the number of long and short instructions executed per normal interation, the required data space, the number of times the program executes per second, and the reliability requirements. Each data element was defined by the number of times it is used per second, its size, and its sources of updated data and the program elements and/or subsystems which use its data.

It is concluded that, provided the data processing requirements can be defined to the above level of detail, a minimum hardware configuration can be derived. Further, in an environment where high reliability is a requirement for some programs, this type of model can be used to determine the effect of executing all programs in a high-reliability mode, with the attendant advantage of relieving the Executive System of the task of mixed mode scheduling.

REFERENCES

NIELSEN, N.R., 1967, "The Simulation of Time Sharing Systems," Communications of the ACM, Vol. 10, pp. 397-412.

GAVER, D.P., 1967, "Probability Models for Multiprogramming Computer Systems," Journal ACM, Vol. 14, pp. 423-438.

UNIVAC Report No. PX 6550-1, 1971, "Space Shuttle Booster Data Management System (DMS) Requirements Analysis," prepared under Contract NAS8-30186.

SMITH, J.M., 1968, "A Review and Comparison of Certain Methods of Computer Performance Evaluation," The Computer Bulletin, Vol. 11, pp. 13-18.

HUGHES AIRCRAFT CO., 1972, "Design of a Modular Digital Computer System," prepared under Contract NAS8-27926.