MICROPROCESSOR FEASIBILITY STUDY AND PRELIMINARY DESIGN FOR AN ARTILLERY FIRE CONTROL APPLICATION

By

DAVID ERNEST WEST

Bachelor of Science in Electrical Engineering

University of Oklahoma

Norman, Oklahoma

1974

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
May, 1976

MICROPROCESSOR FEASIBILITY STUDY AND PRELIMINARY DESIGN FOR AN ARTILLERY FIRE CONTROL APPLICATION

Thesis Approved:

Edward L. Shreve
Thesis Adviser

Craig S Sims

Paul A. McCollum

N N Durham
Dean of the Graduate College

PREFACE

A project is underway at Oklahoma State University to develop an artillery fire control system. A preliminary study has been concluded which demonstrated the feasibility of the proposed system approach by computer simulations. This is a similar preliminary study as to the feasibility of using a microprocessor as the basis of the hardware implementation as an alternative to completely discrete components.

I wish to express my thanks to Dr. Edward Shreve, my thesis adviser, for his valuable guidance. Also, I would like to thank the remaining committee members, Professor Craig S. Sims and Dr. McCollom, for their assistance in the preparation of the final manuscript.

Finally, I would like to express my gratitude to Mrs. Janice Cronch for her excellent typing of the many drafts and the final copy.

TABLE OF CONTENTS

Chapter		Page
I.	INTRODUCTION	1
	1.1 Background	1
	1.2 System Description	1
	1.3 Objectives	5
	1.4 Analysis Procedure	6
II.	DETERMINATION OF MICROPROCESSOR REQUIREMENTS	8
	2.1 Throughput	9
	2.2 Input/Output	17
	2.3 Special Constraints	18
III.	OVERVIEW OF MICROPROCESSORS	20
	3.1 Introduction	20
	3.2 Basic Definitions	20
	3.2.1 Microprocessor Slice	23
	3.2.2 Monolithic Processor	25
	3.3 Integrated Circuit Technologies	25
	3.3.1 MOS Technologies	26
	3.3.2 Bipolar Technologies	27
	3.3.3 Technology Comparison	28
	3.4 Monolithic and Slice Comparison	31
	3.5 Previous Work in Evaluation of Microprocessors for Military Applications	35
	3.6 Reduction of Microprocessor Spectrum for Analysis	38
	3.7 Conclusion	39
IV.	FINAL SELECTION FROM A SURVEY OF AVAILABLE MICROPROCESSORS	40
	4.1 Microprocessor Analysis	40
	4.2 Multiplication with Peripheral Hardware	42
	4.3 Fixed Data Path Microprocessors	46
	4.3.1 Feasibility of Using the IMP-16C	49
	4.4 Variable Data Path Microprocessors	53
	4.4.1 Intel 3000 Series Microprocessor	54
	4.4.2 9400 Series Macrologic	58
	4.4.3 Am2900 Series	60
	4.4.4 M10800 Microprocessor System	64
	4.4.5 Comparison	65

TABLE OF CONTENTS (Continued)

Chapter		Page
V.	APPLICATION OF THE Am2900 SERIES	71
	5.1 Introduction	71
	5.2 System Specification and Design	71
	5.2.1 Memory System	74
	5.2.2 Branch Control	81
	5.2.3 Firmware Register Control	85
	5.2.4 Pipeline Register Operation	87
	5.2.5 Shift and Rotate Functions	91
	5.2.6 Multiplication Hardware	93
	5.2.7 Macrocode and Microcode Summary	96
	5.3 Approximate Instruction Execution Times and Package Count	99
VI.	CONCLUSIONS AND RECOMMENDATIONS	104
	6.1 Summary and Conclusions	104
	6.2 Project Continuance	105
BIBLIOGRAPHY		108
APPENDIX A - IMP-16 INSTRUCTION SET		111
APPENDIX B - MICROPROGRAMED MICROPROCESSOR INSTRUCTION SETS		116

LIST OF TABLES

Table		Page
I.	Instruction Mix for Extended Kalman Filter Implementation	12
II.	Required Constants for Implementation of Special Functions	16
III.	Technology Comparison	30
IV.	Summary of the Differences Between the Two Types of Microprocessors	34
V.	Instruction Mix for Avionics Fire Control	36
VI.	Available Microprocessor Performance	37
VII.	Am25S05 Configurations	44
VIII.	16-Bit Microprocessors	47
IX.	Hardware Support	48
X.	Software Support	48
XI.	Microprocessor Functional Summary	67
XII.	Microprocessor Comparison	70
XIII.	System Elements	72
XIV.	Input/Output Operations	78
XV.	Descriptive Symbols	87
XVI.	Firmware Execution with Pipeline Register	88
XVII.	Execution of Conditional Branch Instructions	91
XVIII.	Shift Control Code	93
XIX.	Instruction Execution Time Summary	100
XX.	Package Count	102

LIST OF TABLES (Continued)

Table		Page
XXI.	Memory Reference Instructions	112
XXII.	Register Reference Instructions	113
XXIII.	Input/Output, Flag and Halt Instructions	114
XXIV.	Transfer of Control Instructions	114
XXV.	Extended Instruction Set	115
XXVI.	Intel 3001	117
XXVII.	Intel 3002	118
XXVIII.	Explanation of Symbols	120
XXIX.	Instruction Set for the 9404	121
XXX.	I-Field Assignment	123
XXXI.	Instruction Set for the 9406	124
XXXII.	Instruction Set for the 9407	125
XXXIII.	2901 Microcode	126
XXXIV.	MC10800 ALU Function Set	127

LIST OF FIGURES

Figure		Page
1.	Fire Control System	1
2.	Fire Control Program Structure	3
3.	Extended Kalman Filter Implementation	10
4.	Flowchart of Extended Kalman Filter	11
5.	Decimal Cordic Computer	15
6.	Basic Computer	21
7.	Elements Common to Most Microprocessors	22
8.	16-Bit Microprocessor	24
9.	8-Bit Microprocessor	24
10.	Speed/Power/Complexity Comparison	29
11.	Basic Microprogrammable System	32
12.	Parallelogram Configuration	45
13.	IMP-16C Card Assembly Block Diagram	50
14.	RALU Block Diagram	51
15.	Block Diagram of a Typical System	55
16.	Intel 3002 Block Diagram	58
17.	Am2901 Block Diagram	61
18.	Am2909 Block Diagram	62
19.	Typical System Using 2900 Family Elements	63
20.	M10800 Family - Pipeline Processor Example	65
21.	Basic Block Outline	73
22.	Memory System	75

LIST OF FIGURES (Continued)

Figure		Page
23.	I/O Bi-Directional Bus System	77
24.	Am2905 Block Diagram	77
25.	Input/Output Control Circuit	80
26.	Input/Output Bus Circuit	82
27.	Microcycle Timing	83
28.	Branch Control Circuit	84
29.	Firmware Control Using Two Bits of Microcode	86
30.	Firmware Control Using Three Bits of Microcode	86
31.	Clock Timing During Conditional Branch	90
32.	Hardware for Clock Control for Conditional Branch	90
33.	Shift Control Circuit	92
34.	Multiplication Hardware Modification	95
35.	Macrocode Summary	97
36.	Microcode Summary	97
37.	Program Development Chart	107

CHAPTER I

INTRODUCTION

1.1 Background

This document involves the application of microprocessors to a tactical artillery fire control system. It is part of an overall program to develop a field portable fire control system. Feasibility was established by previous work done by the Electrical Engineering Department of Oklahoma State University. A simulation of the system has been implemented on the IBM 360/65 with encouraging results. Further work is required in software development, so that a flexible hardware implementation is desirable. Thus, microprocessors shall be studied in the hope that they can produce a reasonable alternative to completely hardwired logic.

1.2 System Description

The fire control system must predict the point of impact and then calculate necessary corrections to the firing azimuth and elevation to achieve the desired impact position. These functions must be performed in real time so that additional rounds may be fired if necessary before the sensor round has impacted. The advantage of the system is that the meteorological effects from temperature, density and winds do not have to be precisely known before firing to enable compensation. The projectile itself serves as a sensor which enables automatic compensation

without individual attention to each parameter.

The overall system can best be described using Figure 1. The radar tracks the projectile to provide position and range rate data. This data is fed into the fire control system at a rate of 20 samples/second. The fire control system then uses this data to estimate the position and velocity of the projectile. These are used to predict the impact point and adjust the firing parameters of the artillery for predicted impact errors.

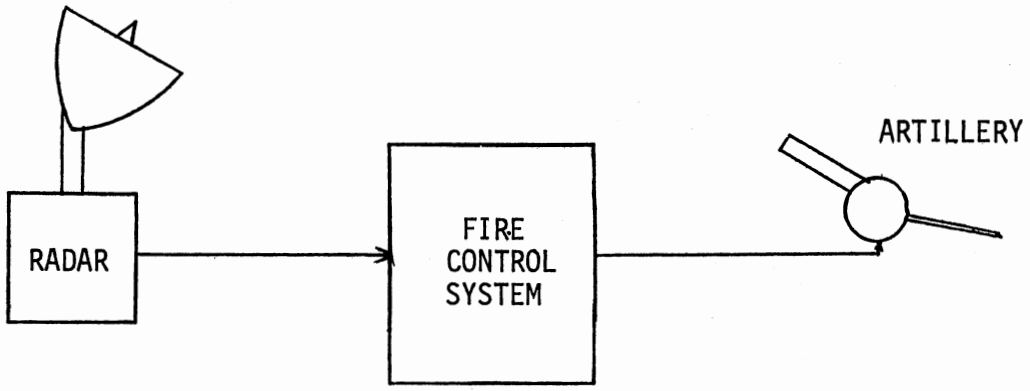


Figure 1. Fire Control System

The general program structure which implements the fire control system is shown in Figure 2. Required operations for this process include coordinate transformations, input data filtering, integration of the state transition matrix and control.

This report will look exclusively at the problem of implementing the Kalman filter and coordinate transformations for the following reasons: the microprocessor analysis must begin by carefully determining

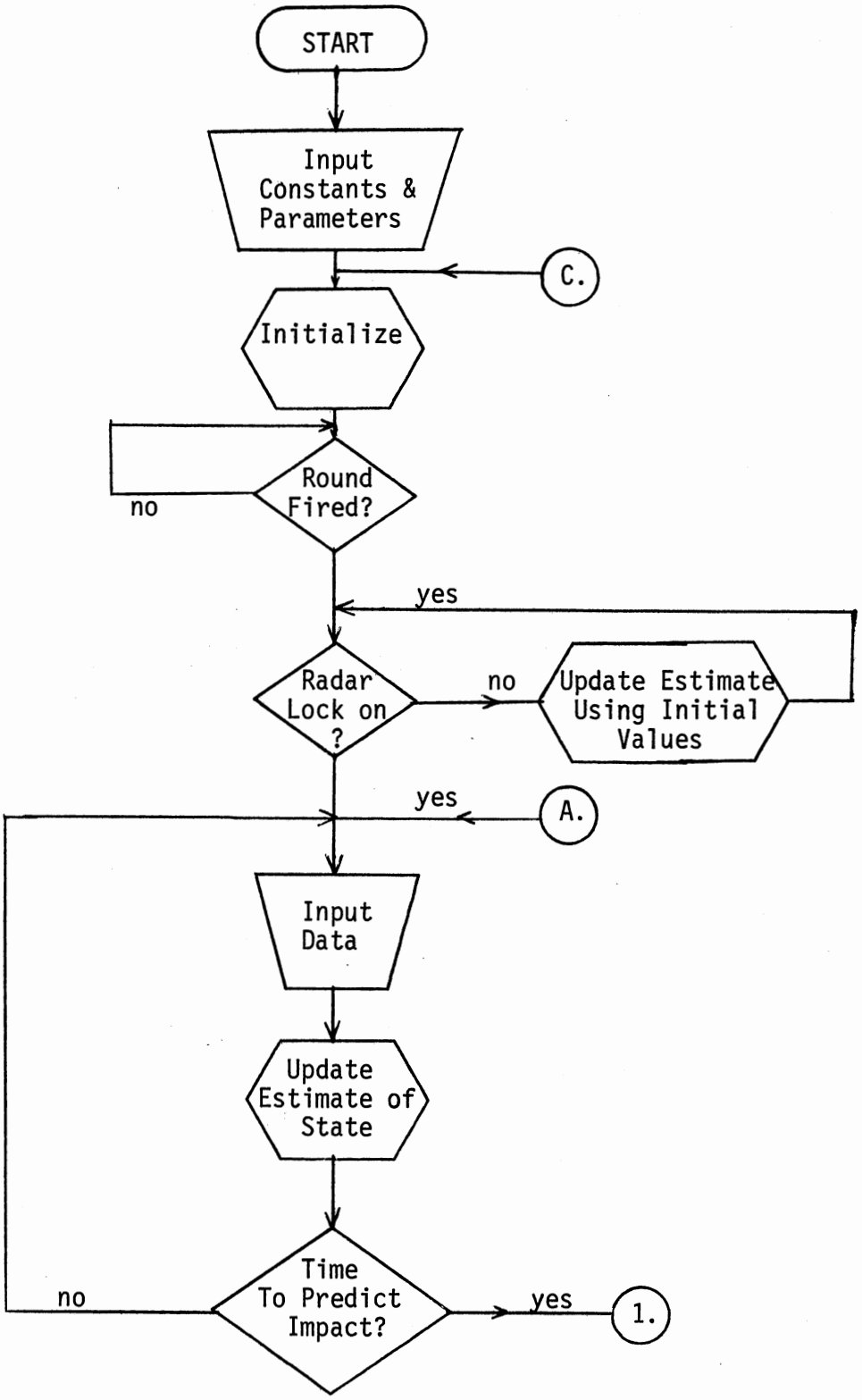


Figure 2. Fire Control Program Structure

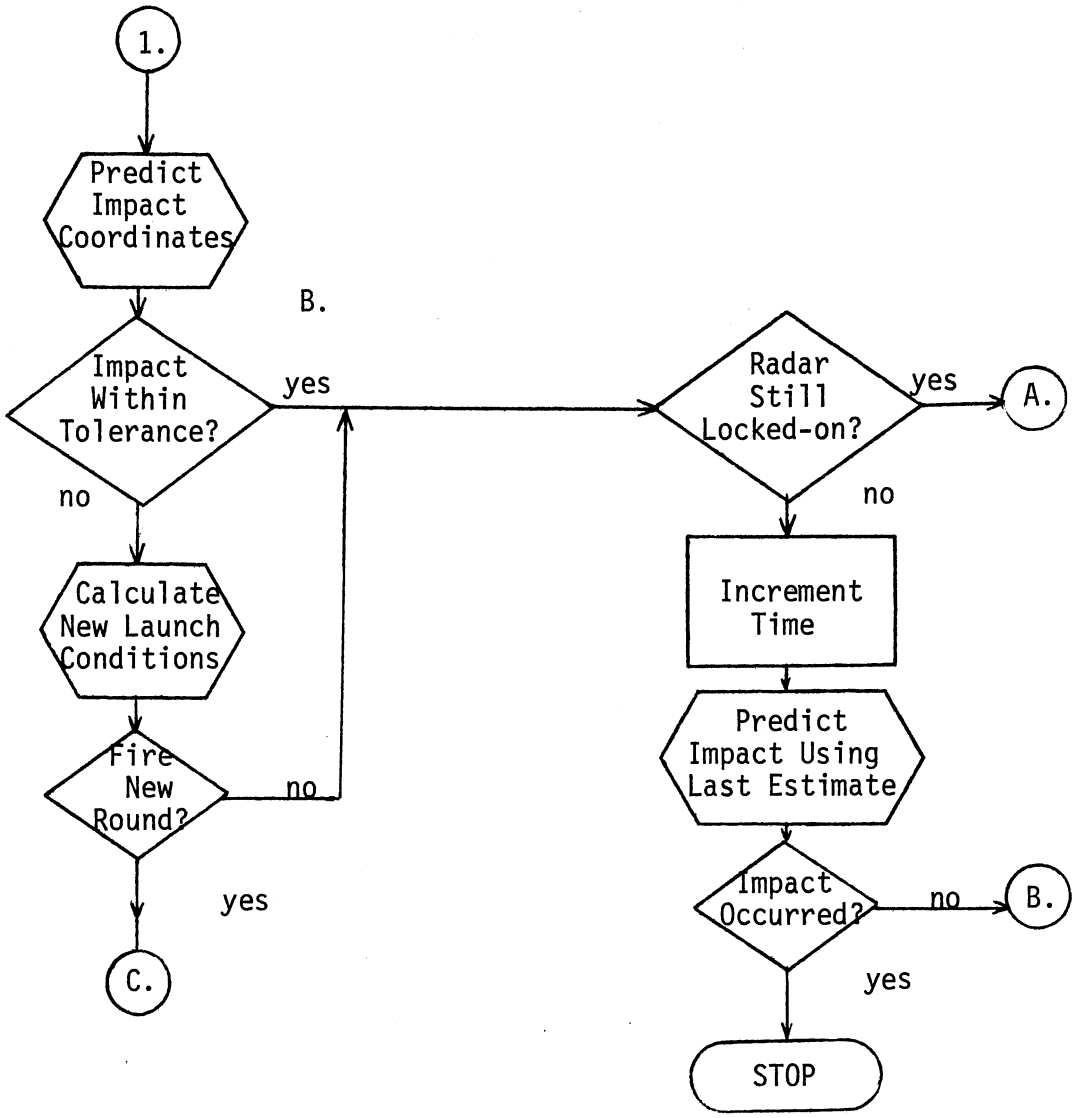


Figure 2. (Continued)

the system specifications, which include power requirements, speed, input/output (I/O) and software. Before these can be determined, the overall system must first be studied to find the subsystem which presents the greatest requirements. The Kalman filter processor has been deemed the most critical not only for numerical accuracy, but also for required speed. The Kalman filter determines the system's accuracy and it must complete calculations within the 20 sample/second data rate. Thus, the microprocessor requirements such as speed and required instructions will be derived from analysis of the Kalman filter and coordinate transformation algorithms.

1.3 Objectives

The major objectives of this analysis are first to make a selection of the required microprocessor and second to begin the application of this processor to the fire control system. This report is part of a preliminary project; therefore, the selection process will be well documented and written on such a level as to enable re-evaluation of the material in the event of system requirement changes. Also, material as to what is termed next generation processors will be included due to the uncertain time schedule of the project. In the event of a significant time delay in the system design, this material may become useful since faster systems are being designed which may become available commercially.

The applications material will be written using a single best choice if one exists. In the event that a clear choice does not exist, application of each will be done to give an idea as to their distinct advantages. A further objective of the application material is to give an

estimate of the required manpower and resources involved to complete the project. Manpower requirements are seen as an important aspect of this problem.

1.4 Analysis Procedure

The study organization has been broken into seven logical steps as shown below:

1. System description,
2. Determination of microprocessor requirements from the system description,
3. Overview of available microprocessors,
4. Reduction of microprocessor spectrum for analysis,
5. Study of available microprocessors within reduced spectrum of analysis,
6. Detailed comparison and selection and
7. Microprocessor application.

The study will begin with the determination of the microprocessor requirements to implement the Kalman filter and coordinate transformation algorithms. This will include determining the required number of bits of accuracy, the time limit for execution and the instruction set. Included in the software requirements will be the necessary subroutine functions.

With the system requirements in mind, an overview of available microprocessors will be presented including an introduction to microprocessors and integrated circuit technologies. From this overview one can reduce the spectrum of microprocessors to those with the most potential for this application. A study of particular processors on the market, which from preliminary analysis satisfy the basic require-

ments, will be made. After the detailed study of individual processors, a comparison and selection process can begin. Finally after the selection is made, Chapter V will be dedicated to the application of the chosen microprocessor. Application material will include hardware system alternatives, hardware multiplication and subroutine implementation if necessary, software organization, instruction definition and programming examples. Manufacturer-supplied software support will be described along with required software support which must be developed.

CHAPTER II

DETERMINATION OF MICROPROCESSOR REQUIREMENTS

Before a search for the best microprocessor can begin one must first investigate the system requirements and working environment of the processor. This chapter will try to document the basic requirements and any special constraints which the chosen microprocessor must meet. Also a look at the problem in detail will give better understanding of the required firmware functions. Preliminary work has been done at Oklahoma State University in the area of software simulations. The basic Kalman filter equations have been studied and some different algorithms for implementing required functions have been documented. These studies will be used to determine the microprocessor requirements.

This chapter will basically draw from previous work to determine specific requirements and will not try to specify the fire controller functions. Specification of the fire control system is presently being done and is not yet in a concrete form. A complete specification of the fire controller is not necessary at this point since one of the basic reasons for selecting a microprocessor system is the added flexibility of firmware.

The system characteristics will be studied to determine three major areas of interest: throughput, input/output (I/O) parameters and special constraints. Throughput requirements are based on the number and type

of functions which the processor must execute in a given amount of time. The I/O parameters specify such requirements as the type of data, required bit-accuracy of various operations and external control or communication requirements outside of the controller system. Finally the special constraints will be specific characteristics which will necessarily constrain selection of the microprocessor.

2.1 Throughput

The firmware implementation of the fire control system has not been completed. The basic variable seems to be in the type of data filter needed to smooth the radar tracking data. Two different methods have been simulated, a polynomial filter and a Taylor series filter, with encouraging results. It is thought that a third method, the extended Kalman filter, would give even better results (38). Thus the extended Kalman filter method, which was the subject of an independent study will be used to determine the throughput requirements.

The time limitation for the execution of this state estimation loop is 0.05 second. This is the time to when the next radar data will be available. Thus the extended Kalman filter calculations must have been concluded so that real time estimation can be obtained of the test projectile path. Figure 3 is a block diagram of the extended Kalman filter prospective implementation. In this diagram the loop of interest includes the integration of the model and transition matrix and the extended Kalman filter. Although the integration routine is shown separately it is actually part of the extended Kalman filter as shown in Figure 4. Figure 4 shows the extended Kalman filter in detail where block two, COMPUTE THE PREDICTED STATE, includes the Runge-Kutta integration routine.

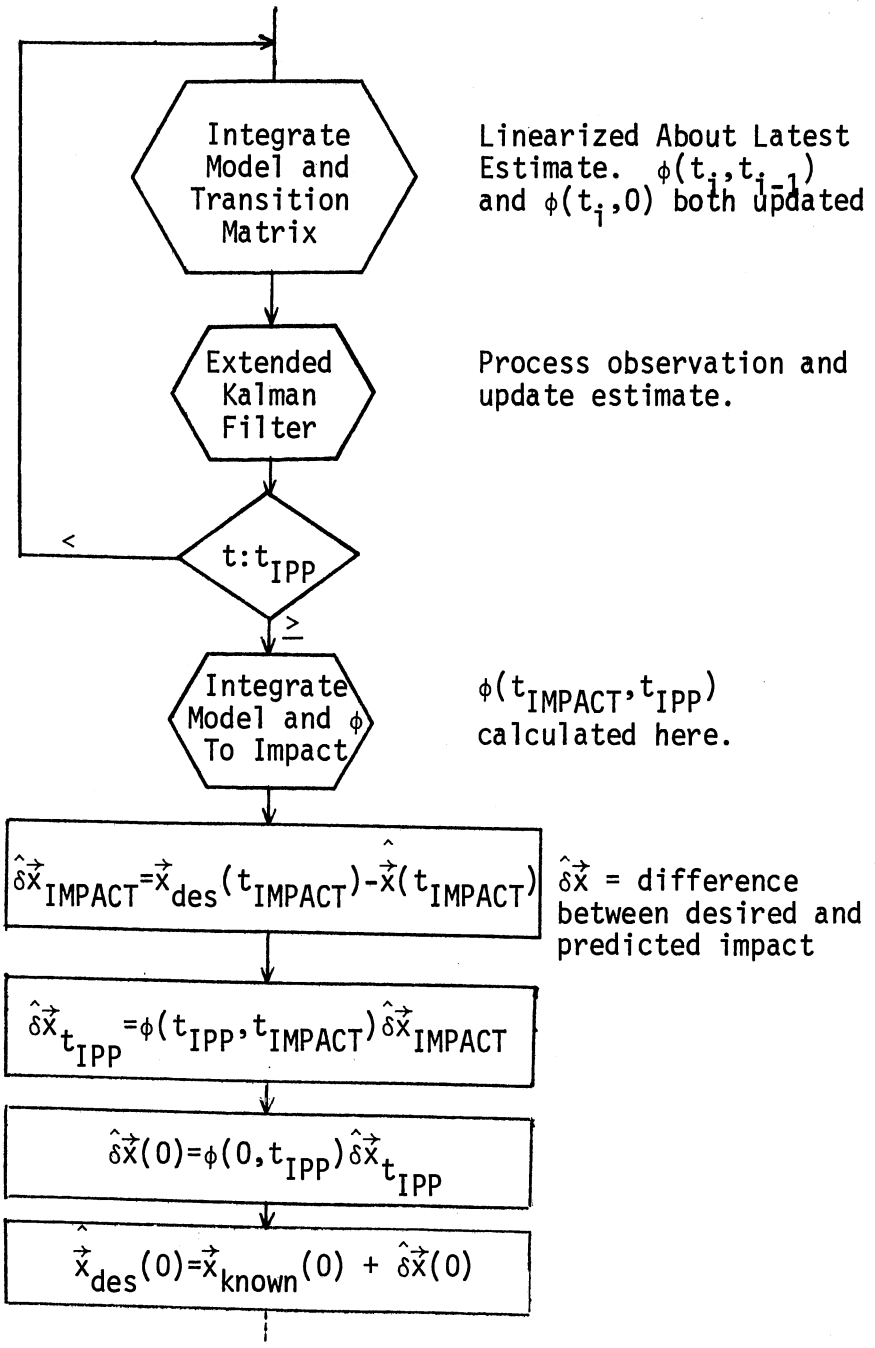


Figure 3. Extended Kalman Filter Implementation

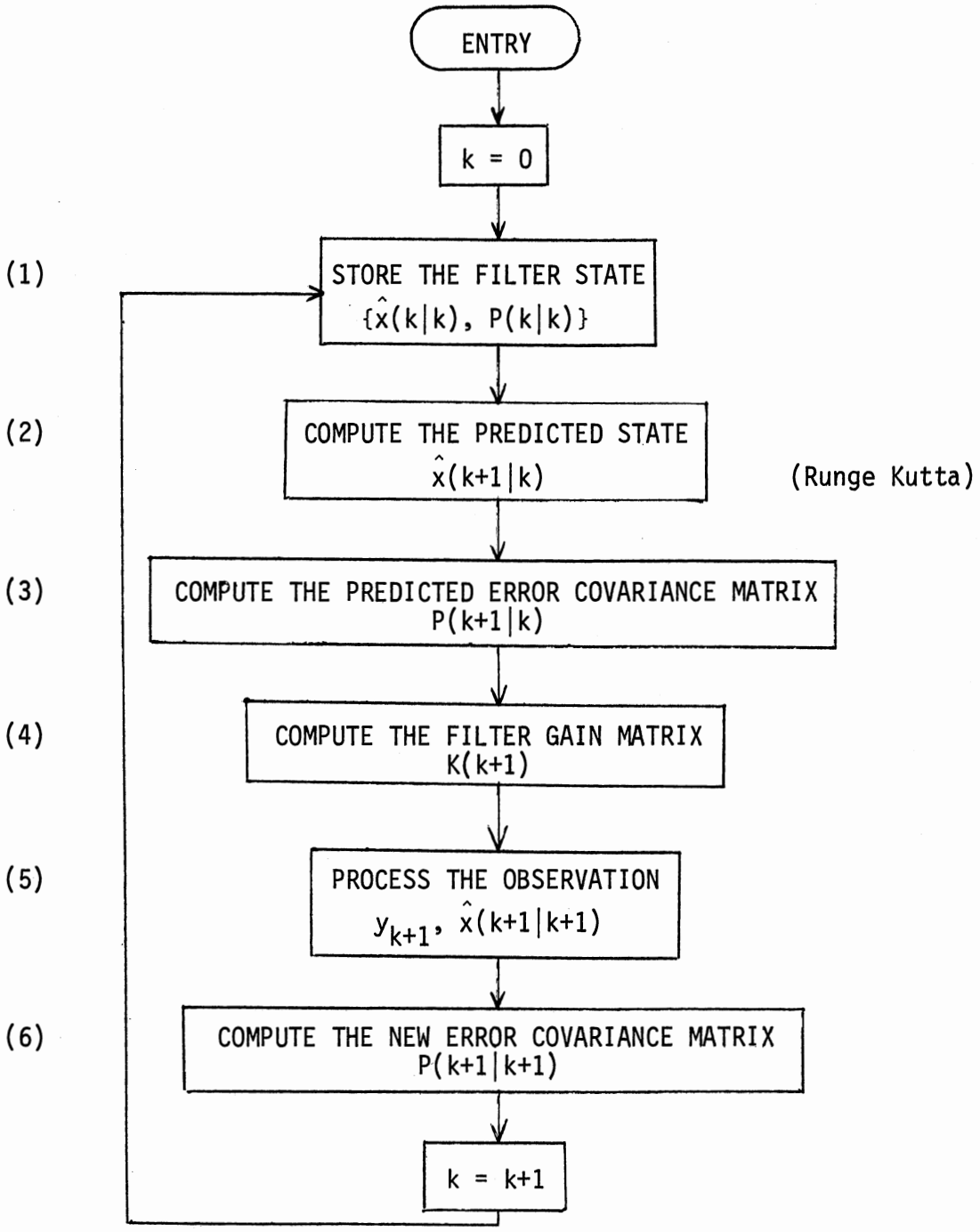


Figure 4. Flowchart of Extended Kalman Filter (28)

The required functions for the execution of the extended Kalman filter system including the Runge-Kutta integration routine are shown in Table I. This instruction mix was part of the results obtained by a separate study (28). Table I was derived by looking at the functions which must be implemented in the extended Kalman filter loop. These are the functions which must be executed during the 0.05 second sample time. The primary purpose of this instruction mix will be to compute the execution time of the critical Kalman filter loop for prospective microprocessors. The number of required executions of each function shown in Table I does not include the use of one function to execute another function.

TABLE I

INSTRUCTION MIX FOR EXTENDED KALMAN FILTER IMPLEMENTATION

Function	Number Required
Add-subtract	900
Multiply	1100
Divide	14
Sin, Cos, Tan	7
Square root	10
Exponential	1
Miscellaneous	4000

The functions of Table I are only the major functions which must be

available as either an instruction or a subroutine. They do not include the many memory reference instructions, register manipulation instructions and jump instructions which will also be required. As a rough estimate 4000 miscellaneous instructions of these types will be included in the instruction mix. This is about twice the number of the special functions. These types of instruction have roughly the same execution times especially when compared to the more complex special functions. As a measure of the basic throughput requirements, the execution time of the extended Kalman filter loop will be calculated based on the instruction mix of Table I. This type of throughput calculation is very rough and only intended as a first approximation. A more accurate instruction mix would require programming the application on each processor being considered. In many cases the execution times of the functions will even have to be estimated. These estimates will be based on the performance of a similar microprocessor with the same function.

One has the alternative of using hardware or firmware for the implementation of these special functions. The only functions of the ones shown in Table I which are generally offered by most microprocessors are the add and subtract functions. The multiply and divide functions are sometimes offered but they are no more than internal firmware implemented requiring long execution times. The remaining functions must be implemented either in hardware or as subroutines. The minimization of hardware would lead to the software implementation of these functions depending on the speed of the chosen microprocessor. It is doubtful that a total firmware solution would have the required throughput. Thus a tradeoff between hardware and firmware implementations will probably be necessary. Although the determination of this tradeoff is part of the

system design, two alternatives will be discussed here.

The first alternative is to use Cordic in the implementation of these functions. Cordic algorithms are implemented using both firmware and hardware. A general decimal Cordic computer is shown in Figure 5. The basis of Cordic is the use of a look-up table in read only memory (ROM) for certain constants which are then used by hardware in the calculation of the required function. The process requires a great deal of control and timing hardware, but it is a fast method of implementation (37).

The other alternative is to allocate the hardware resources to a single function which can be used together with firmware control to implement the remaining functions. All the special functions could be implemented with the addition and subtraction instructions as a base. But a better solution is to increase the number of base functions by hardware implementing the multiply, the most used function. Not only is multiplication a heavily used instruction in the system, but it is also required in the firmware implementation of the remaining functions. Furthermore firmware implemented multiplication would require at least thirty times the execution time of those instructions under the miscellaneous heading in Table I. This means the multiplication function would account for roughly 85 percent of the execution time while only accounting for 18 percent of the total number of instructions. Thus, hardware implementation of the multiply instruction could be used to increase the speed of the firmware implemented functions and increase the overall throughput. The final decision as to which method to use is dependent upon the microprocessor selected.

If the multiply is implemented in hardware, the remaining functions

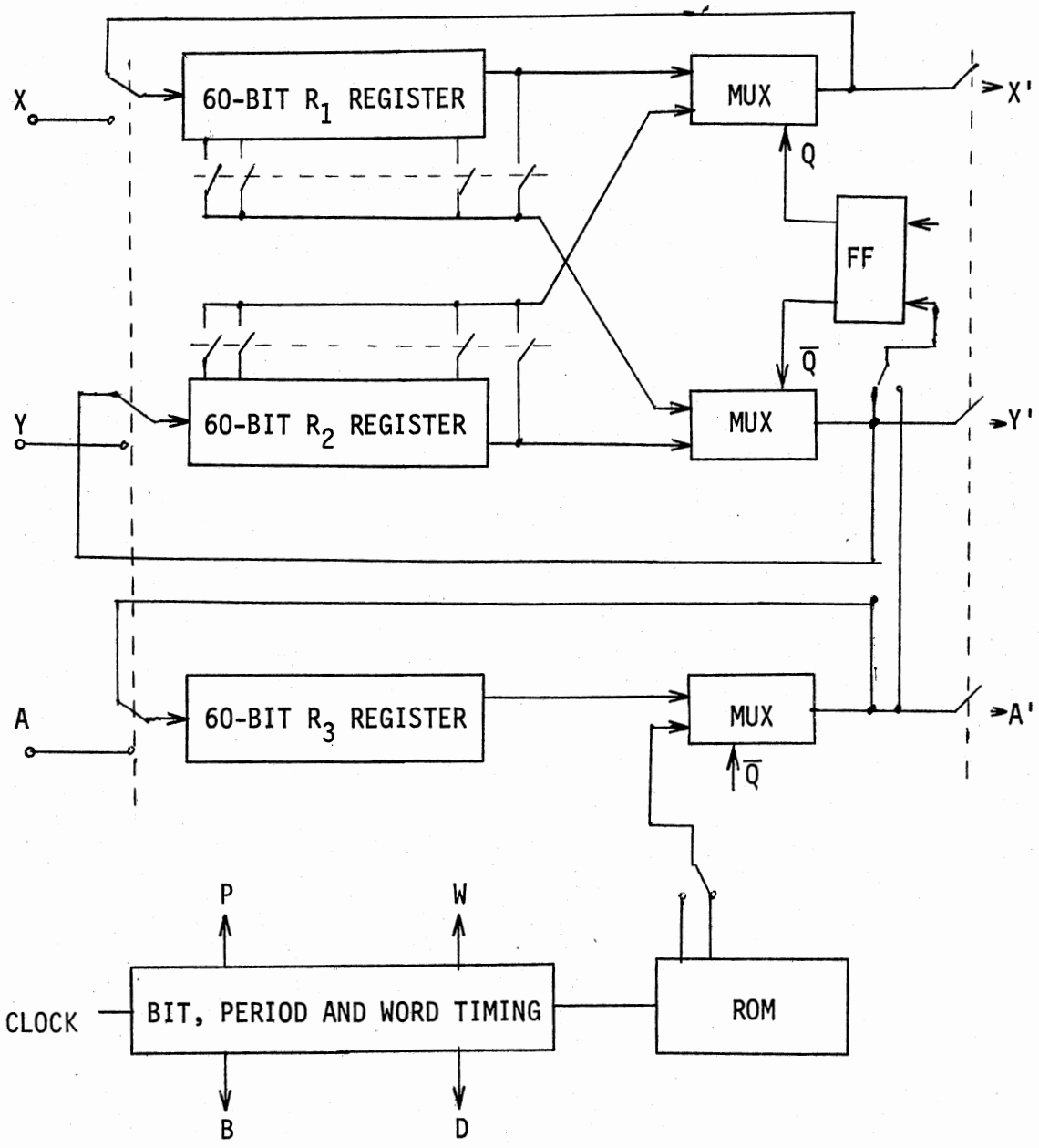


Figure 5. Decimal Cordic Computer (37)

(sin, cos, tan, exponential and squareroot) can be implemented in firmware using rational approximations. These algorithms were investigated independently (28). The implementation of the rational approximation requires a series of multiplies and adds of the argument and certain constants. Thus another required instruction to implement the rational approximations would be a load immediate function or the ability to input constants from program memory. The number of constants required by each function is shown in Table II. The additional number of multiplies is really not significant when compared to the program requirements of 1100 multiplies. For example, only eight multiplies are required for the sin algorithm execution.

TABLE II

REQUIRED CONSTANTS FOR IMPLEMENTATION OF SPECIAL FUNCTIONS

Function	Required Constants	Accuracy (Decimal digits)
Sin	4	10
Cos	6	10
Tan	0	10
Exp	10	12
Sqrt	5	25

The number of constants will have little effect on memory requirements. Each constant must be stored in memory, but either program memory or working memory may be used for storage. The constants must be

accounted for when the memory requirements are finalized.

Thus the throughput of the controller is very dependent upon the multiply instruction execution and will be important in the microprocessor selection. The previous discussion shows that the fire control system will require more general computing power than actual controller type functions.

2.2 Input/Output Parameters

The microprocessor must interface with an external memory and some type of radar data channel. The only periodic input data will be the radar data which are range, range rate, azimuth and elevation. From preliminary simulations done at Oklahoma State University, a data accuracy of at least 32 bits is desirable. The 32-bit accuracy may be obtained by double precision operation of a 16-bit processor but at the cost of reduced throughput. The tradeoff between the increased hardware required for parallel 32-bit operation and the slower but decreased hardware of a 16-bit double precision system must be explored and determined during the microprocessor selection period.

The program memory will be somewhere between 4 kilo (K) and 8K bytes as seen from the previous discussion of the instruction mix. Also the firmware will have many matrix operations requiring external data storage. The matrix operations may involve as large as a six by six matrix (37). The exact working memory requirements are hard to determine but about 128 to 256 bytes of memory will probably be necessary, each byte being 32-bits. These requirements can be determined more accurately when the software has been better defined and the microprocessor system outlined.

2.3 Special Constraints

The major system constraints come from the working environment. The final system is intended to be a field portable artillary fire control processor. Thus, the microprocessor must function over the military temperature range for field use and be relatively small in size to be portable. The power requirements will also necessarily be minimized for portability. These constraints will be observed as nearly as possible, but the operating requirements must first be met. The high speed or large throughput requirement may require a large amount of power.

The basic system will actually have the characteristics of a minicomputer, a large throughput system requiring medium-size-processor numerical ability. It will not have the characteristics of what is usually termed a controller. A powerful interrupt ability is not required. A direct memory access ability is not required since large data block transfers are not involved.

A final consideration which may become a constraint is the necessary manpower to develop a working system. This includes hardware and system design and software support. After the system has been designed, it must be transformed into the code of the microprocessor, which requires coding time and an assembler. If an assembler is not available for the chosen microprocessor, this must also be written. Debug software and special debug hardware will also be required. Thus, the project will require effort which goes beyond the implementation of the controller system. The choice of the microprocessor greatly effects the amount of manpower required for completion of the project.

General system requirements have been discussed so that an orderly microprocessor selection may begin. Actual system design would require

greater detail, but our concern is an overall picture of necessary functions and objectives.

CHAPTER III

OVERVIEW OF MICROPROCESSORS

3.1 Introduction

This chapter will introduce terminology and basic ideas relating to microprocessors. Its purpose is to characterize the different types of available microprocessors and to narrow down the spectrum which will be useful for the fire control application. Once the spectrum of useful microprocessors has been determined, the study and analysis of those which have the most potential may begin. Chapter IV will be dedicated to specific microprocessors presently on the market which meet the present needs.

3.2 Basic Definitions

A processor may be defined as a device which fetches and executes instructions. Figure 6 is a block diagram showing the primary sections of a basic computer. The four computer sections are identified as follows:

1. Memory - For storage of programs
2. Arithmetic - For performance of calculations on data and instructions
3. Input/Output - For exchange of data with external world and
4. Control - For primary control of the arithmetic section with minimal control of the other two sections.

Those sections which fetch and execute the instructions are the arithmetic and control sections, and thus constitute the processing unit.

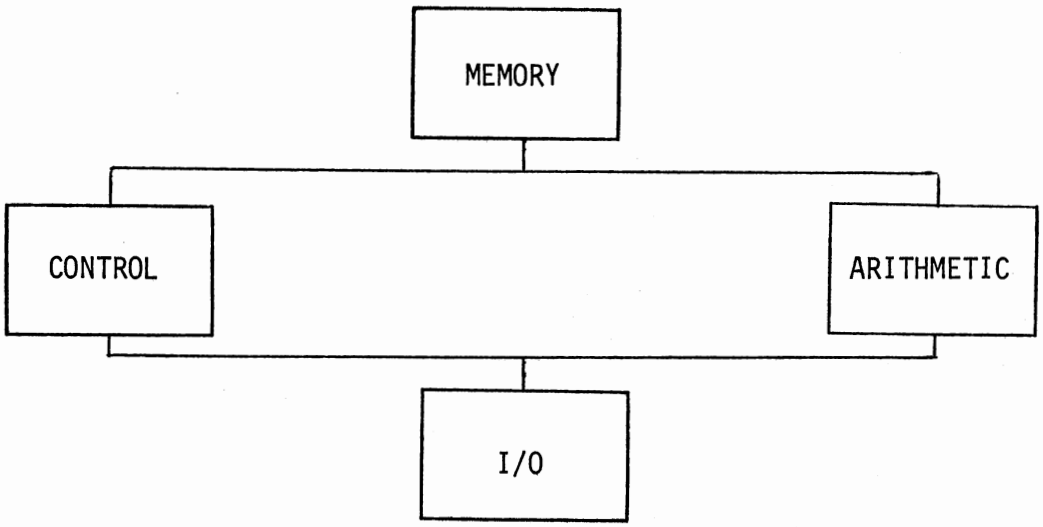


Figure 6. Basic Computer

A microprocessor is a set of circuits necessary and sufficient to perform the functions of a general purpose processor. The number of circuits is relatively small and the packaging density of most of the circuits within this set is in the category of large scale integration (LSI). A detailed block diagram of the elements common to most microprocessors is shown in Figure 7. First a means must be provided to address the memory. Once the instruction has been sent from memory to the processor, it is loaded in the instruction register. Also, instruction decode circuitry must be provided to determine what the instruction is

and to generate the signals required to implement the instructions. Some instructions require an arithmetic or logical operation on data. Thus, the processor must contain an arithmetic logic unit (ALU). Since other instructions may involve temporary storage of intermediate results, various service registers are also provided. Furthermore, a method must be provided to sequence addresses through memory, which means that a program counter is necessary. Finally, address modification, such as indexing or subroutine location, requires various address registers or a register stack. All these elements, together with a timing and control section including I/O drivers and receivers, make up the major blocks of a microprocessor.

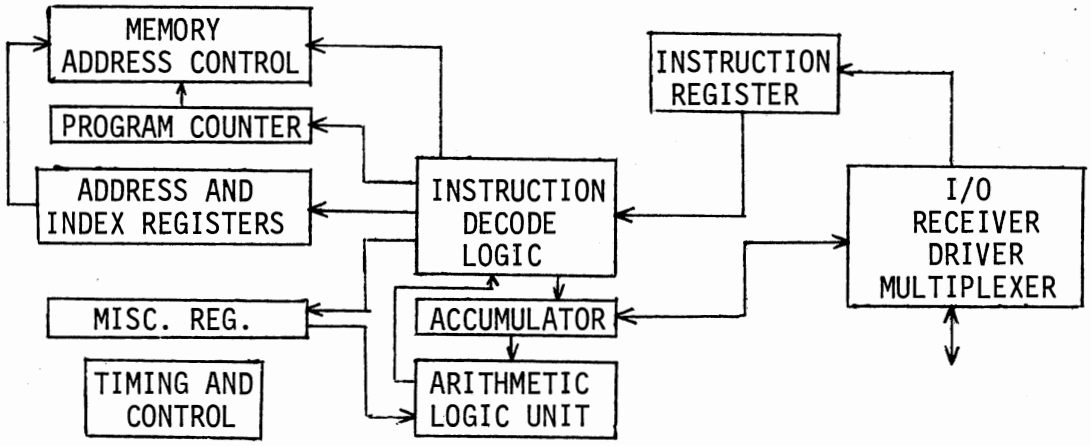


Figure 7. Elements Common to Most Microprocessors

Thus, a microprocessor is a very small and inexpensive processor contained in a single integrated circuit chip or a set of integrated

circuits. The set of integrated circuits refers to the microprocessor slice configuration in which 2-bit or 4-bit "slices" are paralleled to form larger data path systems. The single chip is called a monolithic microprocessor and it has a fixed width data path.

3.2.1 Microprocessor Slice

The microprocessors which have been designed to allow a modularity of data paths and whose control sections can support this modularity are considered to be in the "slice" configuration (14). The basic set of chips which together produce a processor include:

1. Processor Slice - Arithmetic unit which may be used in parallel to achieve the necessary data path width (ALU).

2. Look-Ahead Carry Generator - Capability of multilevel look-ahead for high-speed arithmetic operations over large word lengths.

3. Microprogram Sequence or Control Unit - Control unit containing the instruction register, program counter, etc. depending on the particular manufacturer's system arrangement.

4. Read Only Memory (ROM) - Memory where the microprogram is stored.

These chip sets may also include such extras as timing function chip, slice/memory interface (34), cyclic redundancy check (CRC) generator/checker, serial/parallel first in-first out (FIFO) buffer, data path switch (DPS), p-stack, data access register (DAR) (25), multi-mode latch buffer, priority interrupt control unit, and inverting bi-directional bus driver (30).

Figures 8 and 9 show 16-bit and 8-bit microprocessor chip level architecture where both microprocessors utilize the same chip set

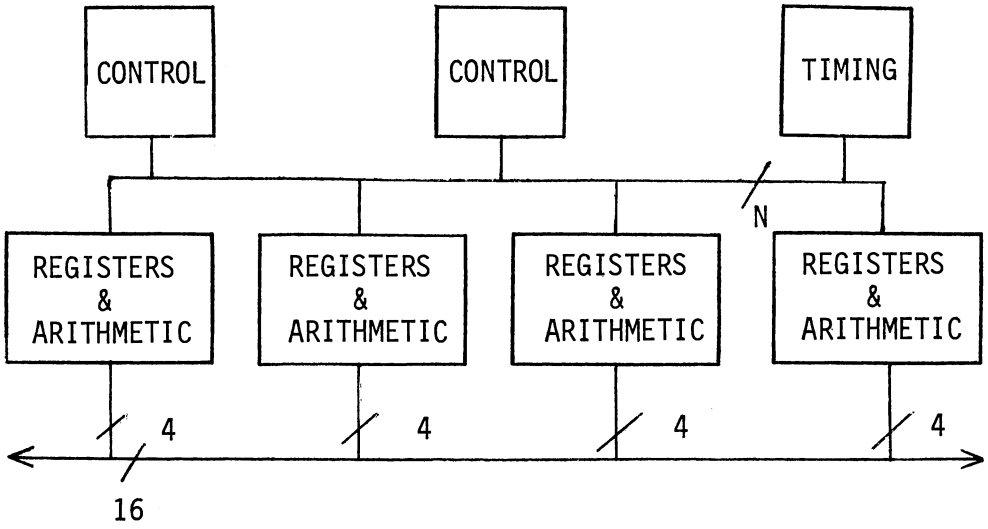


Figure 8. 16-Bit Microprocessor (Each Block Represents a Chip)

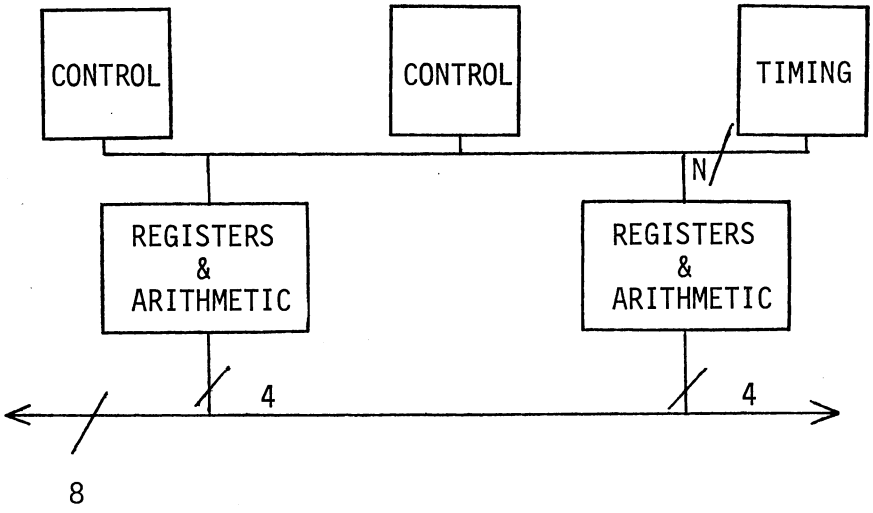


Figure 9. 8-Bit Microprocessor (Each Block Represents a Chip)

with minor modifications to the control circuits. These systems are implemented with 4-bit microprocessor slices and each block represents a chip. The diagrams show that the slice microprocessor can be expanded to different data path lengths but at the expense of more chips and complexity.

3.2.2 Monolithic Processor

The other category of microprocessors is where the design of chips has been optimized for a fixed data path of either 4, 8, 12 or 16 bits. These microprocessors usually combine both control and arithmetic functions into a single chip processor. Although these fixed data path processors have all the processor functions in one chip, they still may have peripheral chips such as read only memories, random access memories, interface adapters and communication interface adapters. On the other hand the single chip may be as complete as to have all control, arithmetic, memory and interface circuitry internal. Thus, in general this class of microprocessors will be called a monolithic processor although it may be part of a set of chips which together produce a working system.

These two categories, monolithic and slice, developed due to the different integrated circuit (IC) technologies. To understand the advantages and disadvantages between these two types of microprocessors, one must first understand the different IC technologies being used to manufacture microprocessors.

3.3 Integrated Circuit Technologies

At this point it is not necessary to discuss in detail the different

manufacturing techniques but instead to point out the distinguishing characteristics of the major technologies. MOS and bipolar are the two major groups under which the major technologies can be placed. MOS technology is based on the Metal-Oxide-Silicone-Field-Effect Transistor (MOSFET), while bipolar technology is based on the familiar epitaxial transistor.

3.3.1 MOS Technologies

The most mature MOS process is PMOS or p-channel MOS. Most first generation microprocessors were PMOS. These were monolithic processors with slow speed capabilities. The major advantage of PMOS is high packing density, which is very important for LSI microprocessor development. These devices also have relatively low power consumption but require multiple power sources. PMOS thresholds, the highest of all existing technologies, often require level translation from transistor-transistor logic (TTL) (bipolar) levels to guarantee that minimum thresholds are achieved (7).

N-channel MOS or NMOS has the potential for significantly higher speed than PMOS microprocessors. N-channel devices require a positive gate-to-source voltage and have a threshold of 1-volt or less. Thus, they are more easily driven by TTL drivers which are pulled-up by resistors (15). Also n-channel outputs are easily made TTL compatible and n-channel silicon-gate MOS can run on a single 5-V power supply if required. These advantages are at the cost of increased fabrication complexity but with high packing density. There are many present generation NMOS monolithic microprocessors (9).

One of the latest MOS technologies is CMOS or complementary MOS.

CMOS uses n-channel devices as drivers and p-channel devices for the load. In this configuration only one transistor is on in the quiescent state producing low standby power dissipation. The advantages of CMOS include high noise immunity, wide tolerance to power supply variation, low temperature sensitivity and low power dissipation (2). The CMOS process also requires isolation between n-channel and p-channel devices which again increases the process complexity and cost. The lower packing density, the major disadvantage, has for the most part kept CMOS out of the LSI microprocessor applications, but next generation CMOS microprocessors are being developed by RCA and Intersil (16). Finally, SOS or silicon-on-sapphire is again an improvement in speed but at the cost of process complexity and cost. An SOS process type microprocessor is being developed by Inselek (16).

In general, the present MOS technologies are relatively slow with high packing density useful for single chip processors. The second major technology is bipolar, which can only be utilized in the slice configuration.

3.3.2 Bipolar Technologies

The general characteristics of bipolar ICs are high speed, low packing density and higher cost. The higher cost results from the fact that the manufacture of bipolar circuits of all types involves five to seven masking steps to print the circuit patterns, as opposed to three for MOS (15). TTL is the most popular and most used bipolar logic family. The major advantages of TTL are the high speed, TTL compatibility, and a single +5-volt power supply. The major disadvantages are the high power consumption and the low packing density (7). The latest

improvement in TTL bipolar devices is the use of a Schottky diode clamp between the base and collector to keep the transistor from saturating. This has produced even better speed performance. Also, low power Schottky devices have been introduced which have about one-fifth the power consumption of regular TTL logic. Present generation slice microprocessors are generally low-power Schottky/TTL (16).

Next generation bipolar technologies include ECL and I^2L. ECL or emitter coupled logic has ultra high speed, more power consumption, temperature sensitivity, nonstandard power supply voltages and complex interconnection requirements (7). However, high-transient switching currents are avoided reducing peak current requirements over similar TTL chips and thereby making power distribution problems somewhat easier. Motorola is presently developing an ECL 4-bit slice microprocessor (34). I^2L or integrated injection logic is seen as the process with the potential of giving the best of bipolar and MOS with good speed and high packing density (10). I^2L attacks the isolation requirement of bipolar devices by careful partitioning and judicious removal of unnecessary resistors. Using this technique, gates can be fabricated which require no isolation within the gate structure (7). At present, I^2L is not as fast as Schottky TTL and does not have the density to produce a monolithic processor. Texas Instruments is presently developing a 4-bit, I^2L slice microprocessor (16).

3.3.3 Technology Comparison

At this point it is possible to compare the different technologies being used to manufacture microprocessors. Figure 10 shows how CMOS, I^2L and LS/TTL fit into the speed/power/complexity picture. CMOS

standard family logic dominates the low-power corner while low-power Schottky TTL takes over when speed requirement goes into the megahertz (MHz) region (2). The new I^2L is expected to be only applicable at the LSI level of complexity, fitting somewhere between CMOS and LS/TTL with respect to speed/power.

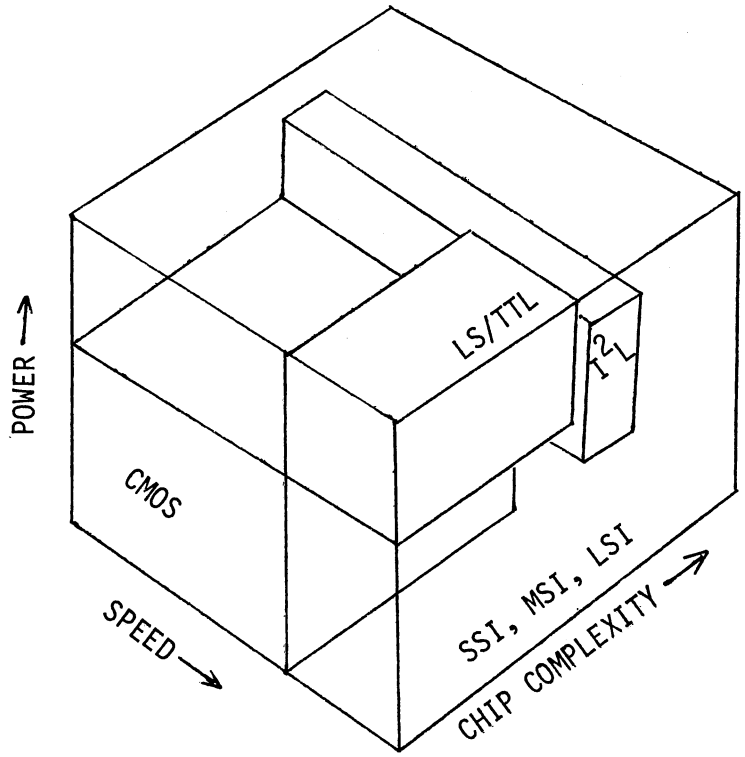


Figure 10. Speed/Power/Complexity Comparison

Source: (2), p. 26.

Finally, an overall technology comparison is made in Table III where each technology is rated from one to ten in each of the categories. Bipolar devices are seen as the best performance devices for

speed but have high power consumption and low packing density. The MOS devices have the opposite characteristics of low power consumption, low speed and high packing densities. This analysis has included the latest technologies being used in microprocessors but one must remember that final recommendations must be made using present generation processors or those in production with reasonable hardware and software documentation. Many problems can arise from trying to use a processor before final specifications are documented, documentation errors are eliminated and software development has been completed to some degree. Thus, final recommendations as to which microprocessor to use in the fire control system will also be based on the extent of development of the chip set.

TABLE III

TECHNOLOGY COMPARISON

Technology Characteristic	MOS			BIPOLAR		
	PMOS	NMOS	CMOS	TTL	ECL	I^2L
Speed	4	7	5	8	9	6
Reliability	6	9	8	5	4	5
Power Consumption	7	8	9	2	1	8
Density	8	9	5	6	2	8
Complexity	6	8	8	9	4	8
Process Maturity	9	7	6	9	8	6
Multiple Sourcing	9	8	2	8	2	1

Source: (7), pp. 64-65.

From the preceding discussion we can see how the two types of microprocessors emerged. The monolithic processor was produced using MOS, but in order to obtain the better performance of the TTL technology, designers had to fragment the microprocessor chip into a slice due to the lower packing density of TTL. The slice configuration not only enables the use of less dense technologies but it also has important system differences. The ability to use faster technologies and the increased fexlibility of the slice configuration has led to the incredibly fast development of bipolar slice microprocessors.

3.4 Monolithic and Slice Comparison

The slice microprocessor has introduced more versatility and greater speed than monolithic or fixed data path processors. The features which make it versatile are its expandable data path and microprogrammable capacity. The microprogram is a flexible firmware system which can be specifically designed to execute a special purpose macro-instruction set. A special purpose instruction set which is tailored to a specific application can be designed.

To better understand the microprogrammable system, a block outline of such a system is shown in Figure 11. Basically, the microprogram ROM and microprogram control unit replace the instruction decode logic shown in Figure 7. The macro-instruction is placed on the data bus from the memory. The micro-instruction address is determined by the macro-instruction and the previous micro-instruction. The microprogram then executes the macro-instruction. Conditional jump commands may be executed from the micro-instruction and feedback flag lines from the CP array. The macro-instruction execution ends with a micro-instruction

which returns microprogram address control to the macro-instruction bus which is the data bus from memory. During instruction execution the micro-instruction controls the central processing array.

Figure 11 describes the Intel 3000 series microprocessor slice which uses standard architecture except for the micro-instruction addressing scheme. The Intel set actually executes a jump the the next instruction where the more standard approach is to use a microprogram counter to sequentially execute the microprogram. In this manner, the macro-instruction or main program counter determines the beginning micro-instruction address which is then incremented for the next address of microprogram.

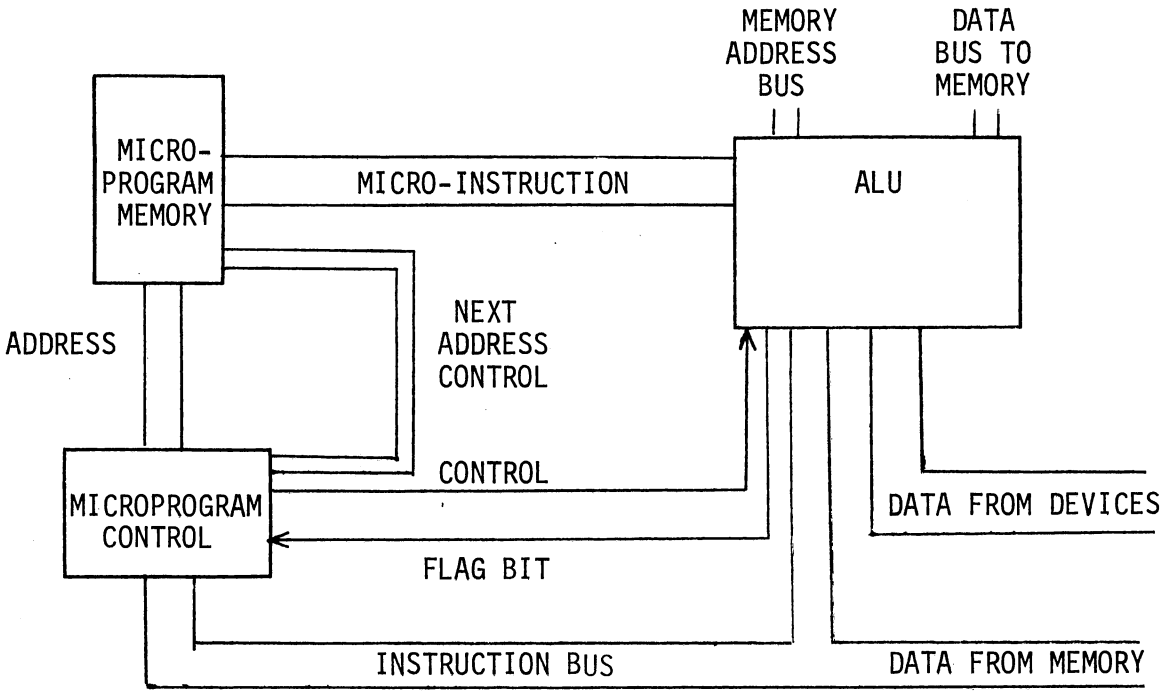


Figure 11. Basic Microprogrammable System

The microprogram is usually stored in a read only memory (ROM) or a programmable read only memory (PROM). The micro-instruction program is usually not changed after it has been designed to execute a given instruction set and thus is stored in a nonvolatile memory. ROMs and PROMs also have much better access times and data storage density than random access memories or read/write memories. Macro-instruction programs can be stored in either RAM or ROM depending on the system. Some systems use a loader program stored in a ROM to automatically load the operating macroprogram into RAM. This is useful when the system has several software options or functions it can perform without hardware modification.

It is not manditory for slice microprocessor systems to have a macro-instruction set. The entire software system may be programed on the microlevel but this adds more complexity to the software design. In this application the micro-instructions provide an even faster system.

The slice configuration has been shown to be the higher performance system but it has two important disadvantages. First, it requires a much more complex hardware system with a higher package count, and second, it requires much greater software development effort. The manufacturers are unable to provide the same software support, such as assemblers and simulators, when the instruction set is variable. Also, the programming on the microlevel is on a lower level and thus more difficult than with assembler language.

Assemblers for slice microprocessors are being developed with a great deal of flexibility designed into them to handle the variable micro-instruction sets (28). This type of support for slice micropro-

cessors is new and will not be available for present use except for only one manufacturer, Intel. Naturally, there are no assemblers available for the macro-instructions since they are user defined.

The major system characteristics for slice and monolithic microprocessors are summarized in Table IV. There are exceptions to the characteristics given in Table IV. There are combinations of these two major groups, such as monolithic processors which are microprogrammable and slice processors which use MOS technology, but presently bipolar technology is confined to slice configurations. This table is a reflection of the present microprocessor trends. Future development will be trying to produce faster systems with both flexibility and simplicity. These systems will probably use bipolar technology with improved speed and packing density.

TABLE IV

SUMMARY OF THE DIFFERENCES BETWEEN THE TWO TYPES OF MICROPROCESSORS

Monolithic Processors
a) Main processor functions in a single chip
b) Fixed instruction set
c) Better software support
d) MOS technology - slower cycle times
e) Simpler system design - smaller chip count
f) Fixed data path - requires use of software for expansion
Slice Microprocessors
a) Chip set centered around processor slice
b) Microprogrammable - variable macrolanguage
c) Less software support
d) Bipolar technology - fastest cycle times, more power
e) Complex system design - high chip count
f) More versatile - expandable data path

3.5 Previous Work in Evaluation of Microprocessors for Military Applications

A paper by Gregory Fox (6) considers the evaluation and comparison of the performance of commercial microprocessors and a discussion of potential military system applications. The paper begins by showing the exponential growth rate of available microprocessors and then points out the difficulty of selecting the best microprocessor for a given application. The difficulty arises when trying to evaluate scaled down, integrated versions of general purpose computer CPUs with documentation consisting of 20 to 30 pages of specifications. Microprocessors cannot be effectively evaluated on a data sheet basis.

Fox states that the performance of a microprocessor is a function of three factors: (1) cycle time, (2) number of cycles for each instruction and (3) the power of the instruction set. As a measure of these three factors, the author used throughput to compare different types of microprocessors. In order to obtain an estimate of a processor's throughput from the instructions' execution times, an instruction mix must be assumed. Table V shows the instruction mix assumed by the author. The instruction mix is a function of the microprocessor application and this mix is for an avionics fire control system. The mix is determined from the percentage use of a certain instruction and that instruction's execution time. It must be noted that this is not the same type of fire control system but is still an interesting evaluation.

Results from a comparison of an 8-bit, single chip, NMOS and a 16-bit, PMOS, slice microprocessor show that better throughput was obtained from the 16-bit slice. Although the 8-bit processor had a

faster cycle time, the use of double precision arithmetic reduced the throughput below that of a slower 16-bit machine designed with slice processors. One important advantage of the slice machine was having both multiply and divide instructions where the 8-bit processor did not. The author points out that the 16-bit processor's multiply requires fifty percent longer than the time indicated in the specification sheet. A further advantage of the 16-bit machine was the shift instruction useful for scaling incoming data. The 8-bit machine only had a rotate instruction which wraps the shifted bit around to the other end. Thus, processors with higher hardware bit accuracy which have the required instructions can overcome smaller but faster machines.

TABLE V

INSTRUCTION MIX FOR AVIONICS FIRE CONTROL

Instruction	Percent Utilization
Load	29.2
Store	18.1
Branch conditional	14.1
Branch unconditional	9.4
Add	8.8
Logical	5.4
Multiply	5.2
Shift	4.0
Subtract	2.5
Divide	1.3
Others	2.0

Source: (6), p.7.

The author cautions potential users about analyzing only instruction times or throughput. These studies are crude and only give rough ideas of a processor's usefulness for a particular application. Further study is required of memory addressing options, number and flexibility of the registers, and the variety of branch conditions available. These are factors which require either actual programming or a great deal of computer experience to evaluate.

Finally, the author does a comparison between different types of processors which is shown in Table VI. He shows that the bipolar slice machine has a considerably better throughput than any other processor configuration. The author also estimates the required throughput for a ground based fire control system as about 10 to 100 KOPS (thousand operations per second).

TABLE VI

AVAILABLE MICROPROCESSOR PERFORMANCE

Microprocessor	Word Length (bits)	Throughput* (KOPS)
8-bit NMOS	8	35
8-bit NMOS	16**	17
Byte Slice	16	42
Byte Slice Bipolar	16	400-500

* Using Instruction Mix in Table V
** Double Precision Arithmetic
Source: (6), p. 8.

3.6 Reduction of Microprocessor Spectrum for Analysis

The slice configuration processors give the best speed performance due to system and technology advantages, which is the major system requirement. The major disadvantages are higher power consumption, lack of software support, increased system complexity, and a large chip count. The only one of these disadvantages which will not affect the present problem significantly is the power consumption. However, a monolithic microprocessor would be considerably slower with a maximum of a 16-bit data path configuration. With a slice microprocessor it is conceivable to use eight 4-bit slices or sixteen 2-bit slices to form a 32-bit hardware data path system. This size system may be approaching a minicomputer in hardware and software design complexity. Thus, the best solution seems to be to use the faster slice configuration with at least a 32-bit data path.

An analysis of the available slice type microprocessors is now required. Because of the expected complexity of a slice configuration of 32 bits, an analysis of a 16-bit monolithic processor will also be done. The limited manpower and resources of this project may become the determining factor. So, one must determine if the slice configuration speed advantage can warrant the increased design effort, or if the monolithic microprocessor system is even a working alternative due to its lack of speed. Chapter IV will be a study of the available microprocessors including the newest of the slice microprocessors and one or two 16-bit monolithic processors. The emphasis is on the slice processors for they seem to be the best hope for a microprocessor system solution to the fire control problem.

3.7 Conclusion

In conclusion, we have introduced the basic concepts of microprocessors showing the effect integrated circuit technologies have had in producing two different types of microprocessors. These two types have been compared and a selection of an expandable data path slice microprocessor has been made based primarily on its speed advantages over monolithic processors. Chapter IV has been allocated to a complete study of the most recent slice microprocessors. Secondly, a monolithic 16-bit processor will be included for purposes of determining if a monolithic microprocessor is a viable alternative in case a large slice system demands excessive manpower.

CHAPTER IV

FINAL SELECTION FROM A SURVEY OF AVAILABLE MICROPROCESSORS

4.1 Microprocessor Analysis

Analysis to determine the best microprocessor for a given application is very difficult unless the system is restricted by some special characteristics such as size or power consumption. The present major consideration is speed, the most difficult of all parameters to precisely measure for a microprocessor unless the exact application is programmed on each processor being considered through benchmark programs. This difficulty arises in trying to evaluate the many parameters which determine speed. A microprocessor's speed is a function of how it is used, software arrangement and chip architecture. One cannot determine the fastest microprocessor directly, but can, through the evaluation of its architecture, determine the processors with the most potential.

The factors which are going to be considered in the determination of a processor's potential include the number of registers or length of stack, the instruction set, cycle time, instruction average execution times, subroutine linkage facilities, external flags and general architectural characteristics. These factors will help determine if the processor can handle the required throughput. Other important factors include chip set complexity, system level of implementation and support software. These all determine manpower requirements. The system level

of implementation refers to whether one designs with large system blocks or with less functional but more flexible small system blocks. Also, the importance of software support cannot be over stressed. This type of software includes assemblers, editors, simulators, and loader programs. Assemblers are required in all phases of firmware development and simulators enable the parallel development of hardware and firmware. Support software when not supplied by the manufacturer becomes a necessary burden upon the designers during system implementation.

Extensive throughput analysis will not be done for the selection process of the slice microprocessor. This type of analysis requires the knowledge of each instruction's execution time and the instruction mix. To obtain instruction execution times would require the microprogramming of a predesigned macro-instruction set. The results derived giving throughput of each processor would be related to its cycle time and the previously discussed characteristics. Thus the conclusions reached from the throughput analysis would probably be the same as the more general analysis of the architecture. Other factors such as developmental manpower requirements are just as important and make the slightly more accurate throughput analysis unnecessary.

Also the selection analysis will not be a process of finding the processor which just meets the speed requirements. The firmware development will be simplified as the percent utilization of the processor's available speed is decreased. A processor with 50 percent utilization requires less programming effort and memory than one with 80-90 percent utilization. Slower processors will be considered only when other unique advantages are obtained from their use.

Previous analysis has led to the selection of a 32-bit bipolar

slice system. This decision is based on the improved speed of the 32-bit hardware versus a 16-bit double precision arithmetic system and the decreased cycle time of bipolar technology. The advantages of the monolithic systems, with better software support and smaller chip counts leading to less manpower requirements, have been discussed. The slower speed of these processors reduced the probability of their use, but the necessity of further study was shown. Thus this chapter will include a feasibility study of the use of a monolithic processor and a study of available slice microprocessors leading to the selection of the best slice processor.

Also, from the results of Chapter II, the extensive use of a multiply instruction was seen. Because of the long execution time of a firmware multiply instruction, the use of a hardware multiplication system has been deemed useful to reduce the system speed requirements. The first section of this chapter will deal with hardware multiplication so that the following analysis of microprocessors may use these results for execution time estimates.

4.2 Multiplication with Peripheral Hardware

There are many methods to achieve hardware multiplication which give different ratios of the hardware versus speed tradeoff. Systems which use readily available large scale integration (LSI) and medium scale integration (MSI) circuits to improve the existing processor's arithmetic logic unit (ALU) multiplication ability give the lower speed improvement with a small cost in hardware. At the other end of the spectrum is the specialized multiplication chip which requires little or no ALU functions to give very fast execution times but with increased

hardware cost. The alternative to hardware multiplication is the microprogram multiply which requires 170 microseconds (μs) for the IMP-16C and 30 μs for the Intel 3000 series for two 16-bit operands. These are the execution times which must be significantly reduced to justify the use of a hardware multiply.

Schmid (17) discusses hardware multiplication systems and in particular a system for the IMP-16C. His application requires no special purpose chips and improves the multiplication time to 23 μs with a 6.5 megahertz (MHz) clock operation. The analysis and application of his system is given in detail. The added circuit blocks, multiplication register and control signal generator, are described along with the software and interface problems. Although Schmid's system has a good balance between hardware and speed improvement, it is more complex and slower than specialized chip multiplication.

An example of specialized chip multiplication is the Advanced Micro Devices' (AMD) Am25S05 four-by-two two's complement multiplier. It is especially useful for the present application since it is expandable to any array size, available in the military temperature range and is very high speed. Flexibility in the hardware-speed tradeoff is also given by use of different chip configurations. Speed can be increased by use of look-ahead carry chips as shown in Table VII.

Other configurations which are not given in Table VII and which are time sequenced arrangements are discussed by Schmid (17) or by Ghest (8) in an AMD application note. Time sequenced arrangements are slower and more complicated but require less hardware.

The hardware increase is sizeable when compared to a microprocessor which may consist of 30-60 packages without the multiplication hardware.

Also the package count in the table does not include any holding registers that may be required or any clock control circuitry for stopping the clock during multiplication if the cycle time is less than the multiplication delay time. On the other hand, the speed of the slowest configuration is 0.315 μs, almost 100 times faster than the Intel microprogrammed soltuion which was for a 16-bit operand.

TABLE VII

Am25S05 CONFIGURATIONS

Configuration	Array Size (bits)	Total Time (μs) for Multiplication	Package Count 25S05	Package Count 54S18
1. Parallelogram carries stay in same row	32 x 32	.315	128	5
2. Parallelogram - carries from lower order multiplication skip to alternate rows where possible	32 x 32	.187	128	15
3. Split into two parts which are added with high-speed carry look-ahead adder	32 x 32	.152	128	32

Caution must be used when comparing the 0.315 μs time to other execution times. This time does not include the time required to input the operands and output the result. This time will vary according to

the microprocessor being used. This chip does produce a significant increase in speed without added system complexity and with little software requirements from the ALU other than set-up type instructions. The application of the Am25S05 is uncomplicated but the hardware increase is significant.

The parallelogram configuration, the first configuration of Table VII, is shown in Figure 12. This configuration requires the lowest number of additional chips of the non-sequenced configurations and is simpler to apply than the sequenced configurations. For these reasons it is probably the best choice of the multiplication hardware systems if the 128 additional required packages are acceptable.

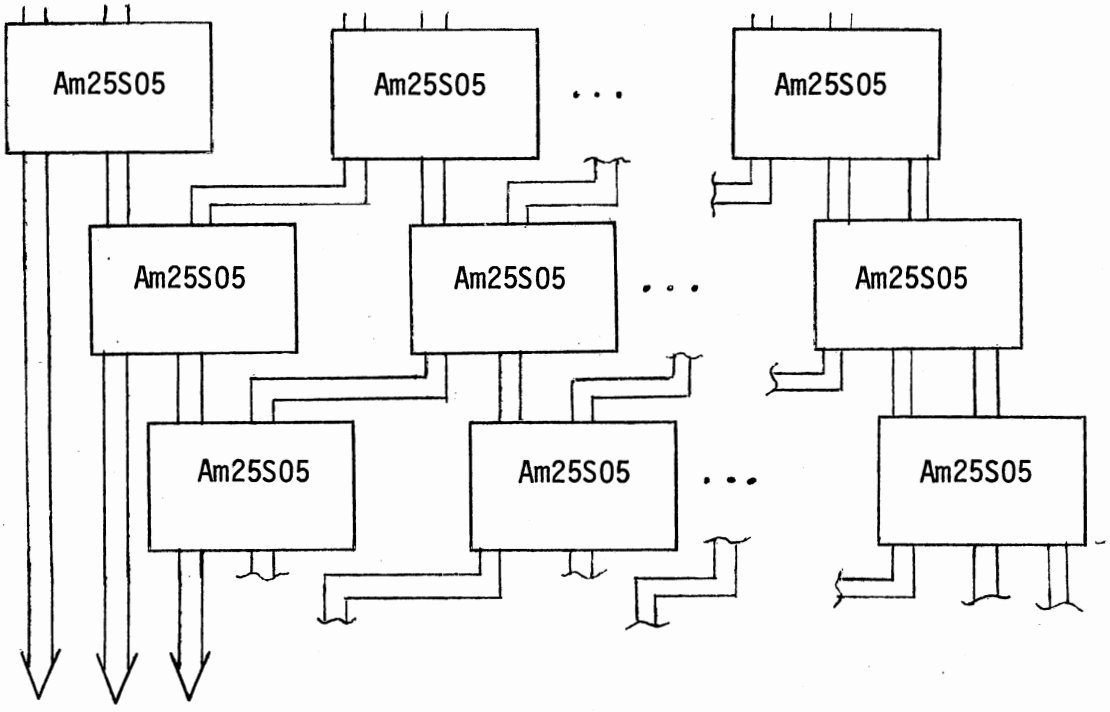


Figure 12. Parallelogram Configuration

4.3 Fixed Data Path Microprocessors

Three representatives of 16-bit fixed data path microprocessors are compared in Tables VIII, IX and X. Table VIII gives an architecture and general characteristics comparison while Tables IX and X give the hardware and software support comparisons, respectively. This is not a complete listing of all 16-bit fixed data path microprocessors but only a representation. The total number of 16-bit processors is not large since the greatest number of fixed data path microprocessors are of the 8-bit structure. The National Semiconductor's IMP-16 is actually designed around a 4-bit slice microprogrammable chip set. Offered as the IMP-16, it is a 16-bit fixed data path, fixed instruction set processor and is thus put into this category.

This comparison again shows the advantage of the lower development manpower requirements due to the hardware and software support. All three microprocessors have resident and cross assemblers, debug, diagnostic, and edit programs and are available on a predesigned card. This type of support is almost necessary for small quantity development programs.

The IMP-16 processor also shows what most experts believe is going to be the growing trend in processor development. This is the predesign of slice microprocessor boards with fixed instruction sets enabling the manufacturers to develop the required support products. As certain MOS chips become established, manufacturers could offer transistor-transistor logic (TTL) slice equivalent systems where improved speed is desired. Engineers could utilize previous experience with the well established chip and save the cost and time of a new learning process. Certain

TABLE VIII

16-BIT MICROPROCESSORS

Manufacturer Model	National Semiconductor IPC-16 (PACE)	General Instrument CP-16007	National Semiconductor IMP-16
General structure	16-bit CPU	16-bit CPU	4-bit slice
Technology	PMOS	NMOS	PMOS
No. of devices per CPU	1	1	
CPU size (pins)	40	40	24
Supply voltage	-12, 5	-3, 5, 12	-12, 5
CPU pwr. dissipation (mW)	700	750	
Data word size (bits)	8 or 16	16	16
Instruction word size (bits)	16	10, 20, 30	16
Directly addressable instruction words (no.)	65K	65K	65K
Clock frequency (Hz/phases required)	2 MHz/20	5 MHz/20	700 KHz/40
Register to register add time (μs/data word)	8	2.4	4.6
No. of registers			
Arithmetic	0	0	4
Index	0	0	0
General purpose	4	8	0
Return stack (no. x bits)	10 x 16	External RAM	16 x No. of bits
Interrupts	Standard	Standard	Standard
Type	Vectored, 6 Level	Vectored, multilevel	1/2/2 Level
Direct memory access	Optional	Standard	Optional
BCD arithmetic (hardware)	Standard	No	No
Microprogrammable	No	No	Yes
Extended temperature range available			$-55^{o}C$ to $85^{o}C$
Delivery start (qtr. year)	3 Qtr. 75	3 Qtr. 74	1 Qtr. 73

TABLE IX

HARDWARE SUPPORT

Manufacturer Model	National Semiconductor IPC-16 (PACE)	General Instrument CP-1600	National Semiconductor IMP-16
Processor cards (CPU system on a card)	Yes	Yes	Yes
Prototyping system (hardware and software development system)	Yes	Yes	Yes
In-system emulator (tests system in place)	No	No	No

TABLE X

SOFTWARE SUPPORT

Manufacturer Model	National Semiconductor IPC-16 (PACE)	General Instrument CP-1600	National Semiconductor IMP-16
Resident assembler	Yes	Yes	Yes
Cross assembler	FORTRAN IV	FORTRAN IV	FORTRAN IV
Simulator	No	FORTRAN IV	No
High level language	SM/PL	No	SM/PL
Programs (firmware)			
Debug	Yes	Yes	Yes
Diagnostic	Yes	Yes	Yes
Edit	Yes	Yes	Yes

options can be made available such as extended instruction sets through additional boards or chips. Even special purpose microprogrammed instructions would be available by working with the manufacturer. These are precisely the advantages which prompted the selection of the IMP-16 for the feasibility study of the use of a 16-bit fixed data path microprocessor in the fire control system. The IMP-16 has an optional extended instruction set which includes multiply, divide, and double precision add and subtract. A total of 43 basic instructions are offered with an optional 17 instructions. See Appendix A. The IMP-16 is also a well established system initially being offered in the first quarter of 1973.

4.3.1 Feasibility of Using the IMP-16C

The IMP-16C is a microprocessor card assembly with the functional blocks as shown in Figure 13. The card assembly is a self-contained 16-bit parallel processor with 256 16-bit words of read/write memory. The off-card memory may be expanded in increments of 4,096 16-bit words to provide a total memory of 65,536 words. Two new versions of the IMP-16C are being offered, the IMP-16C/200 and the IMP-16C/300. The new versions have one extra control read only memory (CROM) for purposes of expanding the basic instruction set. The IMP-16C/200 leaves the socket empty so the user may microprogram desired instructions while the IMP-16C/300 comes with a pre-programed CROM for the extended instruction set. The IMP-16P is also available which is a complete microcomputer.

The architecture of the register and arithmetic logic unit (RALU) is straightforward as shown in Figure 14. There are four working reg-

isters (AC), AC1, AC2, AC3), a status flag register, a program counter (PC), memory data register (MDR), a memory address register (MAR) and a 16-word register stack for subroutine and interrupt return addresses. Two of the accumulator registers (AC2 and AC3) are used as index registers. The status flag register is updated by hardware and includes a link bit, overflow bit, carry bit and 13 general-purpose flag bits. The PC, MDR and MAR serve hardware functions and are not directly available to the programmer.

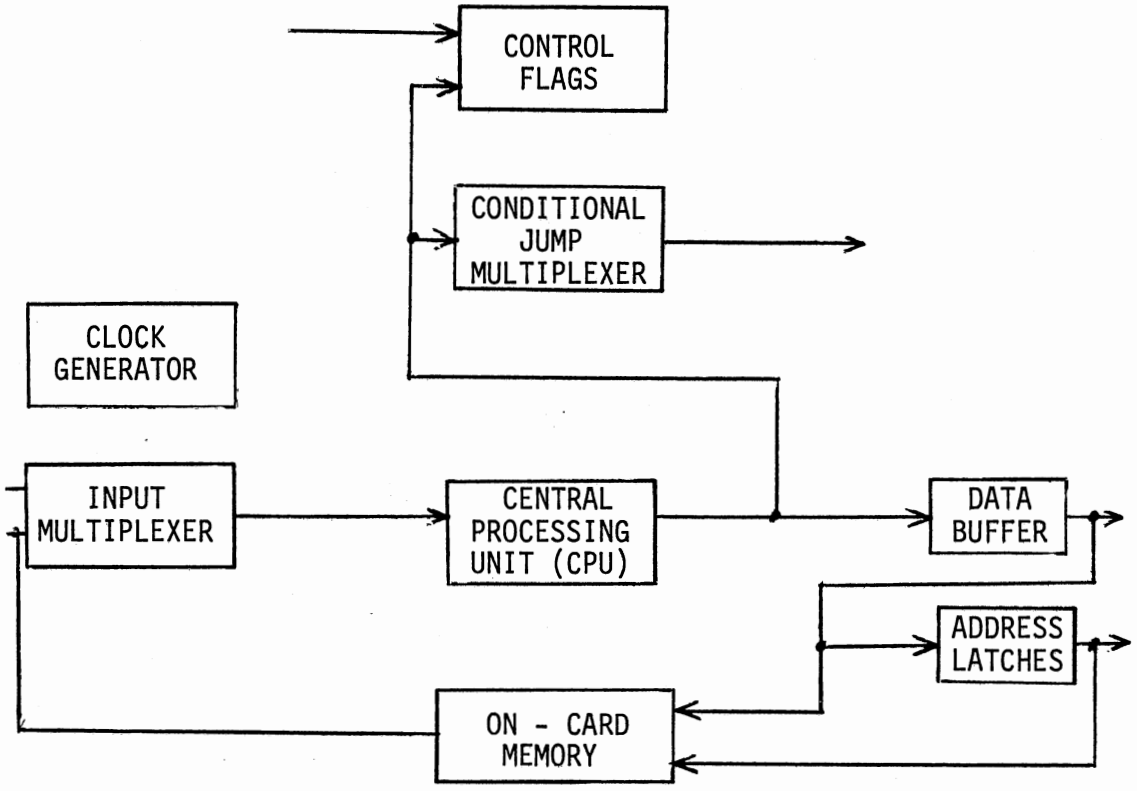


Figure 13. IMP-16C Card Assembly Block Diagram

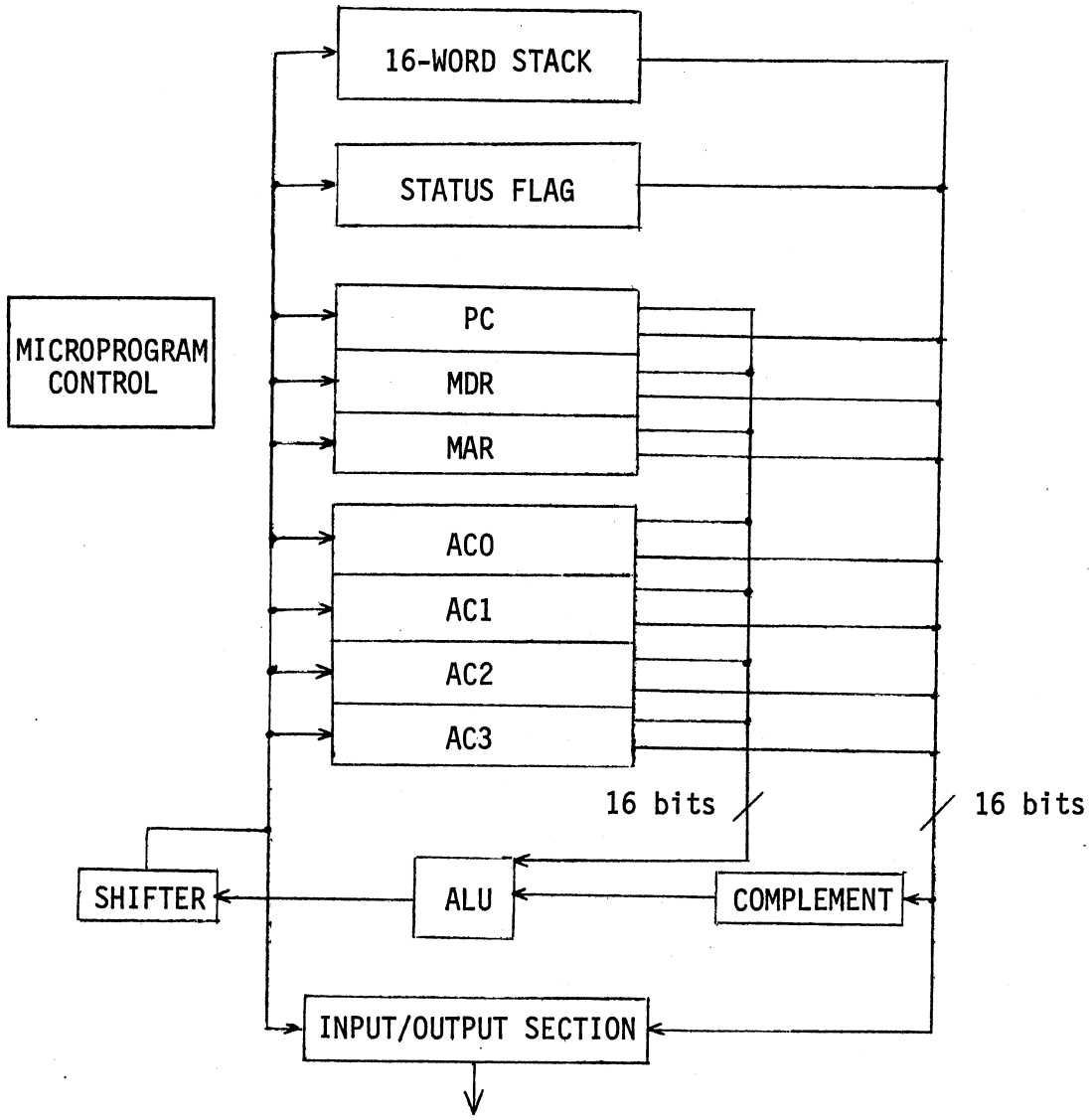


Figure 14. RALU Block Diagram

There are three addressing modes: direct, relative and index. One may address direct to only the base page (0 to 255). To address above 255 one uses either relative, which adds or subtracts from the PC, or index, which adds the contents of the desired index register to the PC. The above system is implemented with a 2-bit mode field and a displace-

ment field.

The basic and extended instruction sets are shown in Appendix A. The execution times are calculated for purposes of throughput analysis. These calculations are based on a 1.4 μs micro-instruction cycle time and a 1.75 μs read/write memory cycle time. The execution time is thus calculated using

$$t_{ex} = (1.4)x(\text{Number of execution cycles}) + (1.75 - 1.4)x(\text{Number of read/write cycles})\ \mu s \quad (1)$$

or

$$t_{ex} = (1.4)x(\text{Number of execution cycles}) + 0.35\ x\ (\text{Number of read/write cycles})\ \mu s \quad (2)$$

Using the instruction mix derived in Chapter II, the critical worst case execution time is calculated below for two cases. These calculations are a very rough first approximation effort.

Case 1 Firmware Multiply Instruction

16-bit multiply: (172 μs) x 1100 multiplies/cycle	=	189	ms*	(3)
Double precision add-subtract: (18.2 μs) x 900 instr./cycle	=	16	ms	(4)
Firmware 16-bit DIV: (223.65 μs) x 14 instr./cycle	=	3	ms	(5)
Trigonometric instr. (Cordic): (87 μs) x 7 instr./cycle		.609	ms	(6)
Memory access instr.: (8 μs) x 4000 instr./cycle	=	32	ms	(7)
Total		241	ms	

* ms = millisecond

Case 2 Hardware 32-bit Multiply

32-bit multiply: (.315 μs) x 1100 instr./cycle	=	.346	ms	(8)
Double precision add-subtract: (18.2 μs) x 900 instr./cycle	=	16	ms	(9)
Firmware 16-bit DIV: (223.65 μs) x 14 instr./cycle	=	3	ms	(10)
Trigonometric instr.: (87 μs) x 7 instr./cycle	=	.609	ms	(11)
Memory access instr.: (8 μs) x 4000 instr./cycle	=	32	ms	(12)
Total		51.96	ms	

These calculations show that without any modification the IMP-16 is too slow. With the addition of a hardware multiply system it may be able to work at the 0.1 second data rate. The 50 percent margin is not excessive with these rough calculations. System time requirements and memory requirements usually become accurate only with extensive experience with the microprocessor. If the advantages of the IMP-16 warrant its use even at this speed a further, more detailed time calculation would be required using the instruction execution times in Appendix A.

This IMP-16 description and analysis was included mainly to give insight into the manpower requirement advantage and speed limitation of the fixed data path systems. If manpower and project schedule requirements restrict the use of the obviously capable slice microprocessors then one must explore the newer, faster systems. The IMP-16 had the smallest manpower requirements while the newer fixed data path systems will be somewhere between the IMP-16 and slice systems. The slice application is a very large jump in manpower from any fixed data path system.

4.4 Variable Data Path Microprocessors

The following sections give brief discussions of five slice configuration microprocessors which represent the state of the art in microprocessors. The discussions will be limited to mainly architecture until the comparison section which will include a hardware comparison table. Also the instruction sets or control words are shown in tabular form for each processor in Appendix B.

Each of these microprocessors has the speed for this application. The selection factor is thus more related to the ease of application or

manpower requirements than speed.

A similar calculation is done below for slice microprocessors as was done for the IMP-16. Using the slowest times of the 5-slice systems, the critical times are calculated as before.

Firmware mult. 32-bit: 80 μs x 1100 instr.	=	88	ms	(13)
Add-sub. instr. 32-bit: 1 μs x 900 instr.	=	0.9	ms	(14)
Firmware div. 32-bit: 80 μs x 14 instr.	=	1.12	ms	(15)
Trig. functions (Cordic): 87 μs x 7 instr.	=	0.61	ms	(16)
Mem. access instr.: 1 μs x 4000 instr.	=	4.0	ms	(17)
Total		94.6	ms	
Hardware mult. 32-bit: 1 μs x 1100 instr.	=	1.1	ms	(18)
Add-sub. instr. 32-bit: 1 μs x 900 instr.	=	0.9	ms	(19)
Firmware div. 32-bit: 80 μs x 14 instr.	=	1.12	ms	(20)
Trig. functions (Cordic): 87 μs x 7 instr.	=	0.61	ms	(21)
Mem. access instr.: 1 μs x 4000 instr.	=	4.0	ms	(22)
New Total		7.0	ms	

These calculations show that the firmware multiplication instruction requires 88 percent of the 100 millisecond data rate alone. Yet the total time requirements of the hardware multiplication system is seven percent of the data rate. The importance of the hardware multiplication system is that the 50 millisecond data rate may be obtained with ease and with room for expansion due to either experimental firmware or double precision arithmetic.

4.4.1 Intel 3000 Series Microprocessor

The Intel 3000 series microprocessor set is a family of Schottky bipolar LSI circuits which include the 3001 microprogram control unit (MCU), 3002 central processing element (CPE), 3003 look-ahead carry generator, 3212 multi-mode latch buffer, 3214 priority interrupt control unit, 3226 inverting bi-directional bus driver and the 3301, 3304, 3601,

3604 memories. A block diagram of a typical system is shown in Figure 15.

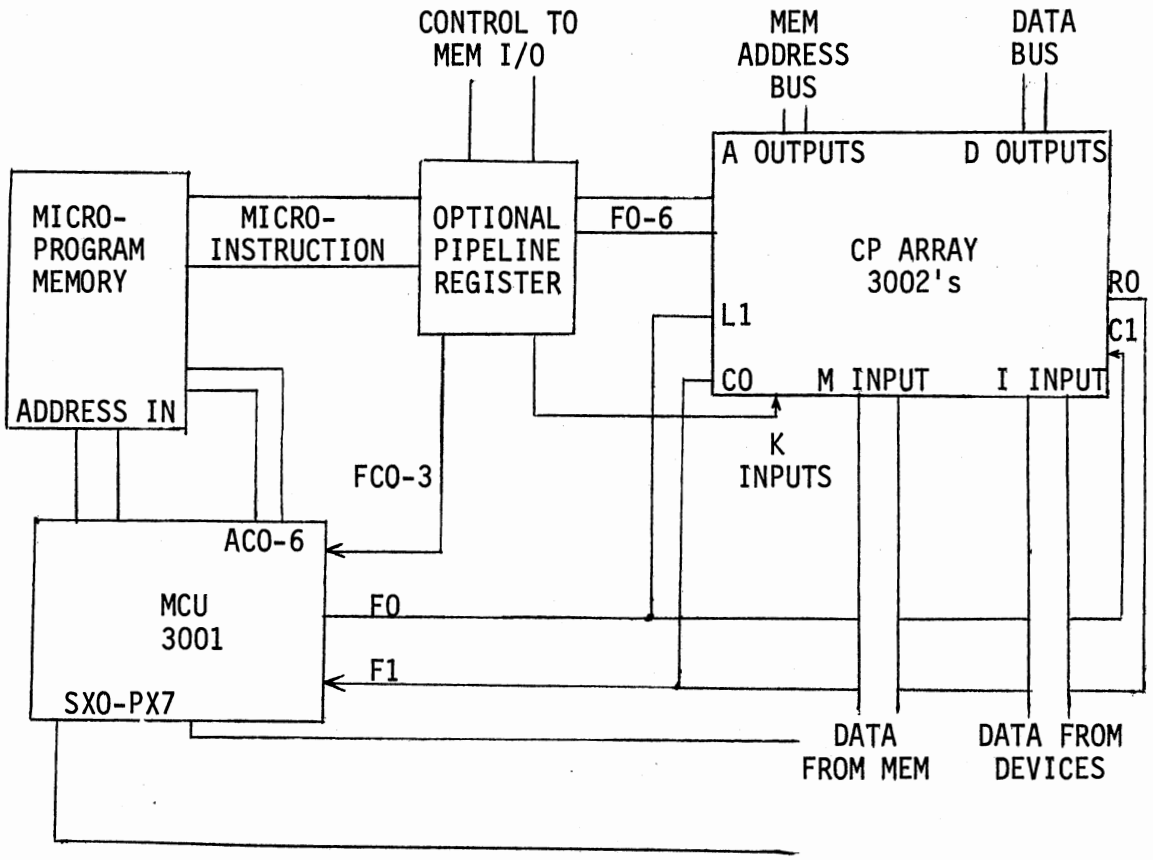


Figure 15. Block Diagram of a Typical System

One of the most important and unusual system blocks is the MCU which has the functions of address control. The unusual aspect of the MCU is the addressing scheme in which each read only memory (ROM) location is represented by a row and column address. Furthermore, the instruction fetch is accomplished through the execution of micro-instruction

jump commands such as jump in current row or jump in current column (see Appendix B). The jump command is encoded in each micro-instruction in a special jump field. This is in contrast to the often used sequential address selection using a program counter. The 3001 addressing scheme allows a jump unconditionally in one operation anywhere in the present row or column. It is not possible however, to jump anywhere in the address matrix. For a given location in the matrix, there is a fixed subset of microprogram addresses that may be selected as the next address.

This present address dependency leads to some microprogramming difficulty. The recommended procedure for assigning memory locations to each instruction is as follows:

1. First write the microprogram without regard to the assignment. For conditional jumps, use the basic conditional jumps provided by the MCU (JFL, JCF, JZF, JPR, JLL, JRL, JPX), noting the number of possible destinations for the conditional jump chosen. However, when a sequence of instructions is to be executed unconditionally, do not indicate what jump codes will be used to advance to the next state unless the JCE enable feature is required. Similarly for unconditional jumps use the non-committal code JMP rather than selecting a JCC, JZR, or JCR.

2. Prepare a state sequence flowchart for the program. According to the programmer's preference, this may be done before, during or after the actual writing of the code. Label the conditional jump points on the flowchart.

3. Using the flowchart as a guide, perform the assignment. In general, conditional jumps should be assigned first, with clusters of conditional jumps assigned before isolated jumps. Leave long chains of unconditional sequences for last. The process of assignment can be assisted by using a diagram of the control memory showing the 32 rows and 16 columns. As each state is assigned, the control memory diagram is marked to show occupancy of that word and the flowchart marked to show the assignment of the state. With the assignment complete, the numbers are copied from the flowchart (29).

One can see that the complicated addressing scheme of the 3001 can be a disadvantage; yet Intel is the only manufacturer of slice micro-

processors which has developed a micro-assembler. This micro-assembler (XMAS) is written in FORTRAN IV and is designed to assemble microcode to produce a ROM programming file. Other user aids such as a micro-program memory map and a cross-reference dictionary are also provided but actual assignment of microprogram memory addresses to micro-instructions is left to the user. XMAS is flexible and extensible in both micro-instruction length and microprogram memory address space. Also fields may be added to the micro-instruction word length to as large as 64 bits. Thus, one of the advantages of the Intel set is the micro-assembler.

The basic micro-instruction is made up of three fields which are seven bits of micro-instruction sequence control to the MCU, four bits of carry control to the MCU, and seven bits of function selection to the CPE array. This basic micro-instruction of 18 bits is then extended to control special purpose functions such as interrupt control, I/O controls, CPE clock inhibit and generation of constants to be issued to the CPE array. The CPE clock inhibit field (one bit) is very useful because it allows non-destructive testing of CPE registers via the MCU carry logic. The carry logic in the MCU responds just as if the micro-instruction were executed, but the fact that the CPE clock was inhibited leaves the CPE unaltered.

Another important system block which requires detailed discussion is the 3002 CPE which is a 2-bit slice register and arithmetic logic unit. It's capabilities include two's complement arithmetic, logical AND, OR, NOT and exclusive OR, incrementing and decrementing, shifting left or right, bit testing and zero detection, carry look-ahead generation and multiple data and address buses. The 3002 has 11 scratchpad

registers designated R_0 through R_g and T, and one accumulator (AC). The block diagram is shown in Figure 16 and the CPE microfunctions are given in Appendix B.

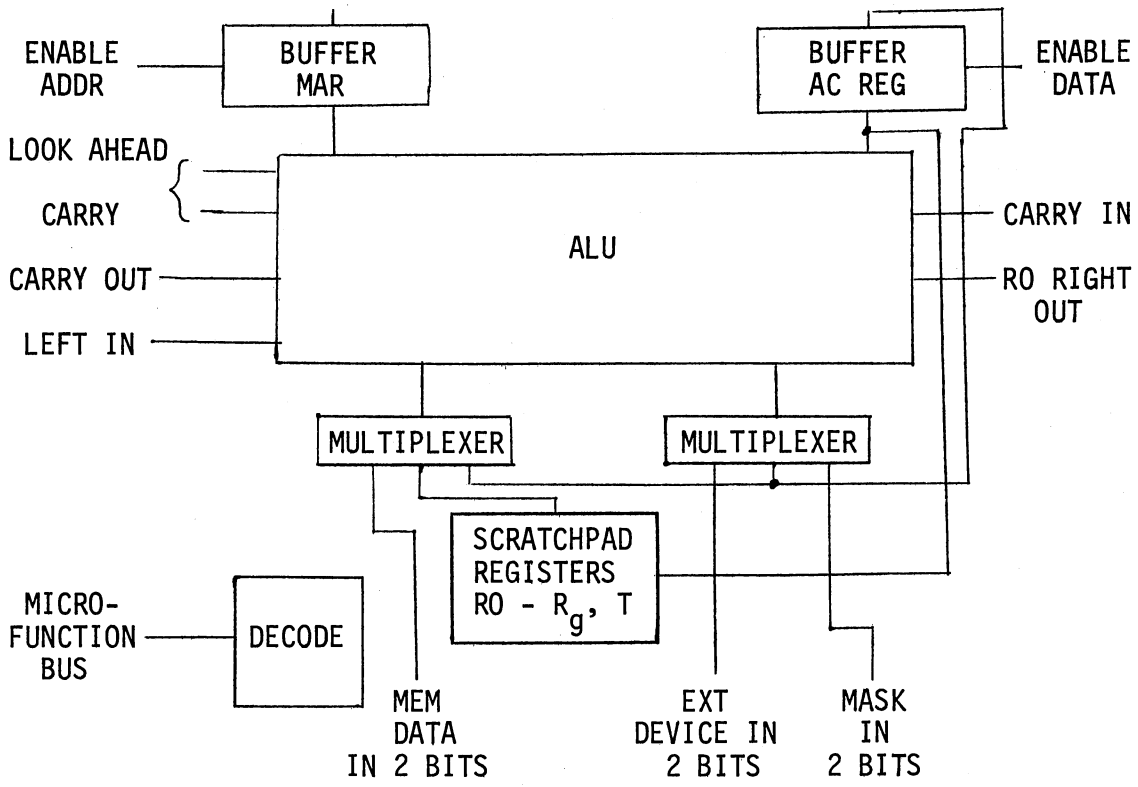


Figure 16. Intel 3002 Block Diagram

4.4.2 9400 Series Macrologic

The Fairchild Schottky TTL 9400 series macrologic is designed at a lower level, meaning more system blocks are required to produce a working processor. The system blocks include 9401 cyclic redundancy check (CRC), generator/checker, 9403 serial/parallel FIFO, 9404 data path

switch (DPS), 9405 arithmetic logic register (ALRS), 9406 p-stack, 9407 data access register (DAR), and 9410 16 x 4 clocked random access memory (RAM). The basic processor functions are broken into four chips, the 9404 DPS, 9405 ALRS, 9406 p-stach and the 9407 DAR.

The 9405 ALRS consists of a 4-bit ALU, an 8-word by 4-bit RAM with output latches, an instruction decode network, control logic, and a 4-bit output register. Data for ALU functions are provided from the RAM and from the input data lines ($\overline{D}_0$ - $\overline{D}_3$). The instructions are shown in Appendix B. R_x is the RAM location.

The 9406 program stack consists of an input multiplexer, a 16 x 4 RAM with output latches addressed by the stack pointer (SP), an incrementer, control logic, and output buffers. The 9406 implements four instructions as determined by inputs I_0 and I_1. (See Appendix B.) Thus the main function of this device is control of subroutine execution. This device is also expandable as a 4-bit slice.

The 9407 DAR performs memory address arithmetic for RAM resident stack applications. It contains three 4-bit registers intended for program counter (R_0), stack pointer (R_1), and operand address (R_2). The 9407 implements the 16 instructions shown in Appendix B.

The overall microprocessor functions are broken into smaller functions producing more system flexibility but also increasing complexity. Each member is also a 4-bit slice. The ALRS has only eight registers and the ALU inputs are fixed to the data input bus and one internal register. The ALRS seems to be the weakest link in the system. The 9406 program stack and the 9407 DAR on the other hand are strong points. These two chips may be useful when combined with other manufacturers' systems which have more flexible ALU and register systems but lack

program control logic. Overall the macrologic does not provide the system simplicity we desire.

4.4.3 Am2900 Series

The Advanced Micro Devices (AMD) Am2900 series microprocessor circuits are low-power Schottky devices with a total of 14 different chips in the family. The major blocks in the system are the Am2901, 4-bit bipolar microprocessor slice and the Am2909 microprogram sequencer. The Am2901 is almost a complete system including register and data control, internal register matrix and ALU. A block diagram of the Am2901 is shown in Figure 17. The impressive architectural features include 16 general purpose registers, one Q-register, two separate shifting networks, and internal ALU input selection.

This ALU input selector can select eight different input combinations between the Q-register, direct input and two of the 16 internal registers. The two internal registers selected from the 16 total are addressed from the A and B address buses which may be used to select the same register at the same time. An output multiplexer is also included for direct access to the register file or ALU output.

Hardware advantages include a faster cycle time than the 3000 series, less power consumption and fewer required chips due to the 4-bit slice architecture. Also the total architecture requires only two different major chips other than I/O and holding register requirements.

The micro-instruction word is nine bits, three bits for ALU source operand selection, three bits for ALU function selection and three bits for ALU destination control. These fields and their associated functions are shown in Appendix B.

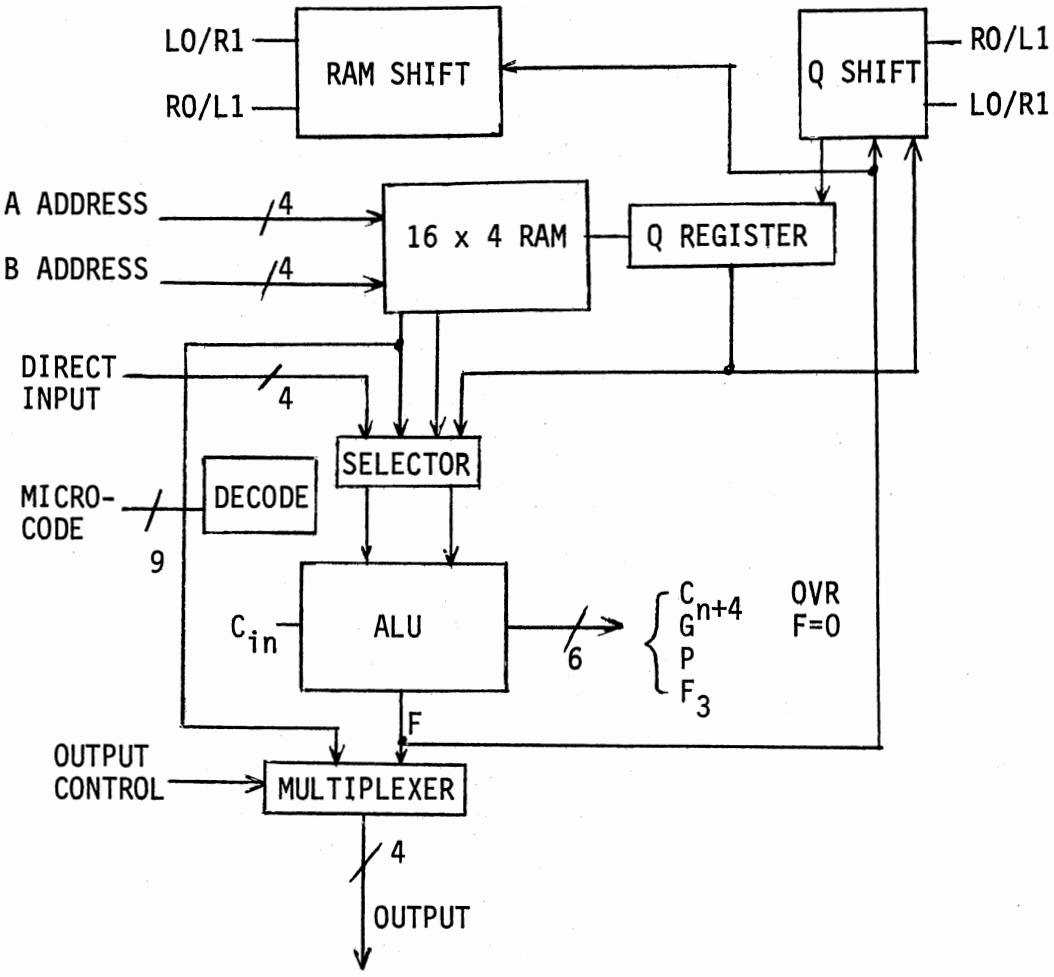


Figure 17. Am2901 Block Diagram

When discussing the Am2901 one must also mention the Monolithic Memories Inc. (MMI) 6701 which is architecturally identical to the Am2901. The major difference is the Am2901 is twice as fast and has some architecture improvements. Also the MMI 6701 does not have any peripheral chips such as the Am2909 microprogram sequencer at present. For these reasons the MMI 6701 was not included in the microprocessor selection process.

The second important system block, the Am2909 microprogram se-

quencer, is shown in Figure 18. This chip includes a register stack for subroutine return address storage, program register and incrementer, instruction register and associated multiplex and control circuits for branch and subroutine control. This microprogram control chip is totally different from the Intel 3001. The major differences are the program addressing schemes and the Intel's lack of subroutine execution hardware. The Intel chip would require an external RAM with either an up-down counter for stack pointer or additional bits in the microinstruction for control. The 3001 has the advantage of internal control circuits for condition jump execution.

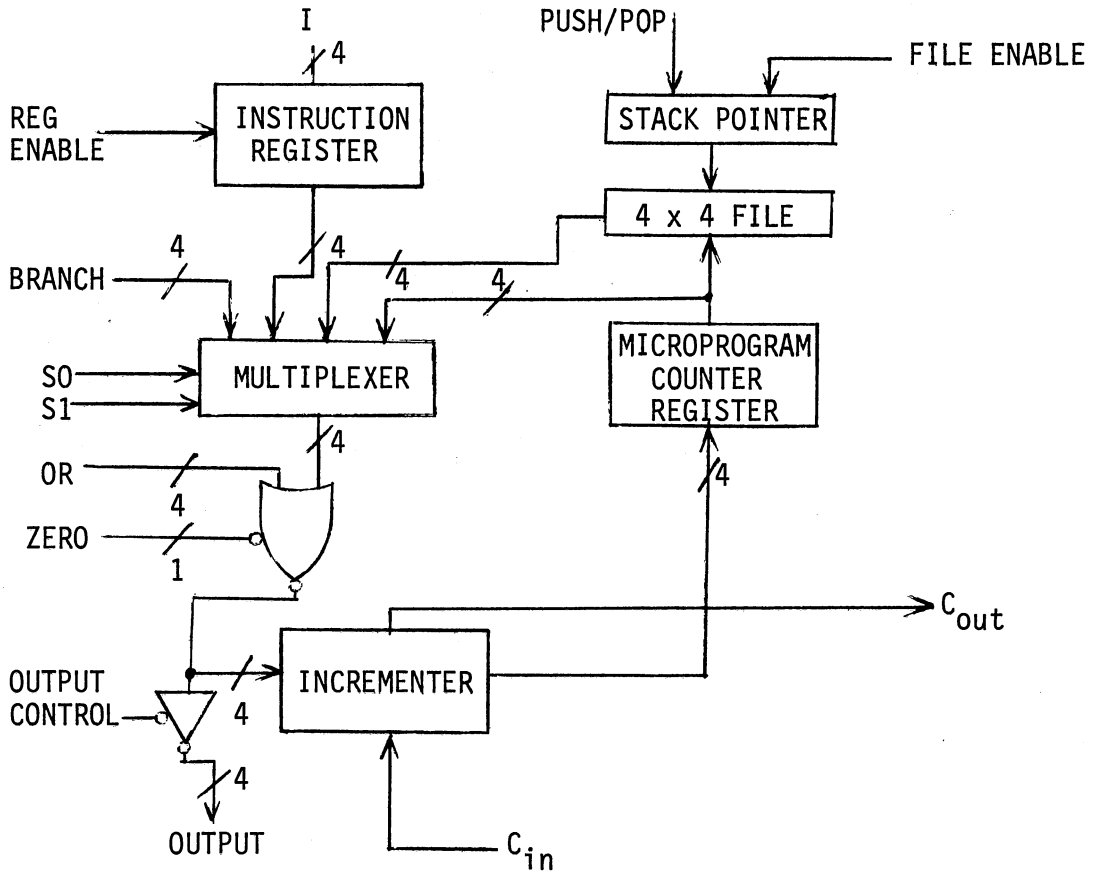


Figure 18. Am2909 Block Diagram

Finally, a typical central processor unit using the Am2900 series microprocessor circuits is shown in Figure 19. This shows the other family members in a typical system arrangement.

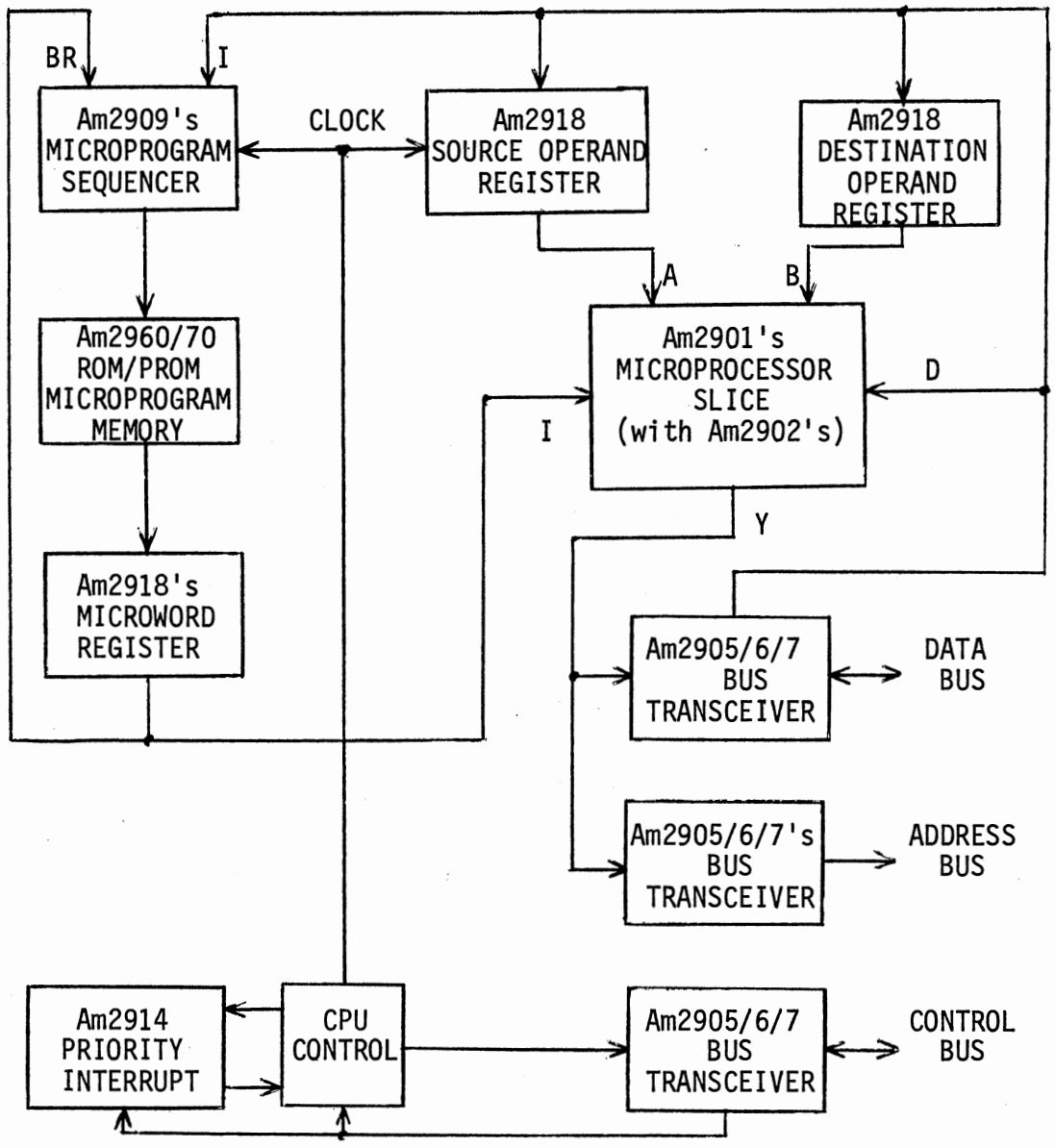


Figure 19. Typical System Using 2900 Family Elements

4.4.4 M10800 Microprocessor System

Motorola's M10800 microprocessor family is the fastest system to be discussed, being an emitter coupled logic (ECL) chip set, and it also has a well thought-out architecture. The family members include the MC10800 4-bit processor slice, MC10801 control function, MC10802 timing function, MC10803 slice/memory interface and the MC10804 slice look-ahead carry. The architecture is designed at about the same level as the 9400 series but with distinct differences. The 10800 family is also the most versatile system (see Figure 20). The basic functions are included in four chips requiring external multiplexers and a RAM for ALU registers. The basic functions of each member are described in the comparison table, Table XI.

Very little application literature is available, since official announcement of the M10800 microprocessor family won't be until 1976. The only additional information given other than a general description is the ALU function set shown in Appendix B. This shows the very complete function set of the ALU. The distinguishing features which make the system versatile are the ability to use external RAM for the ALU working registers and the use of external multiplexers. Detailed analysis is impossible without better data, but the system blocks are conveniently segmented. At this point, however, this level of performance is not necessary and would require more hardware and development manpower.

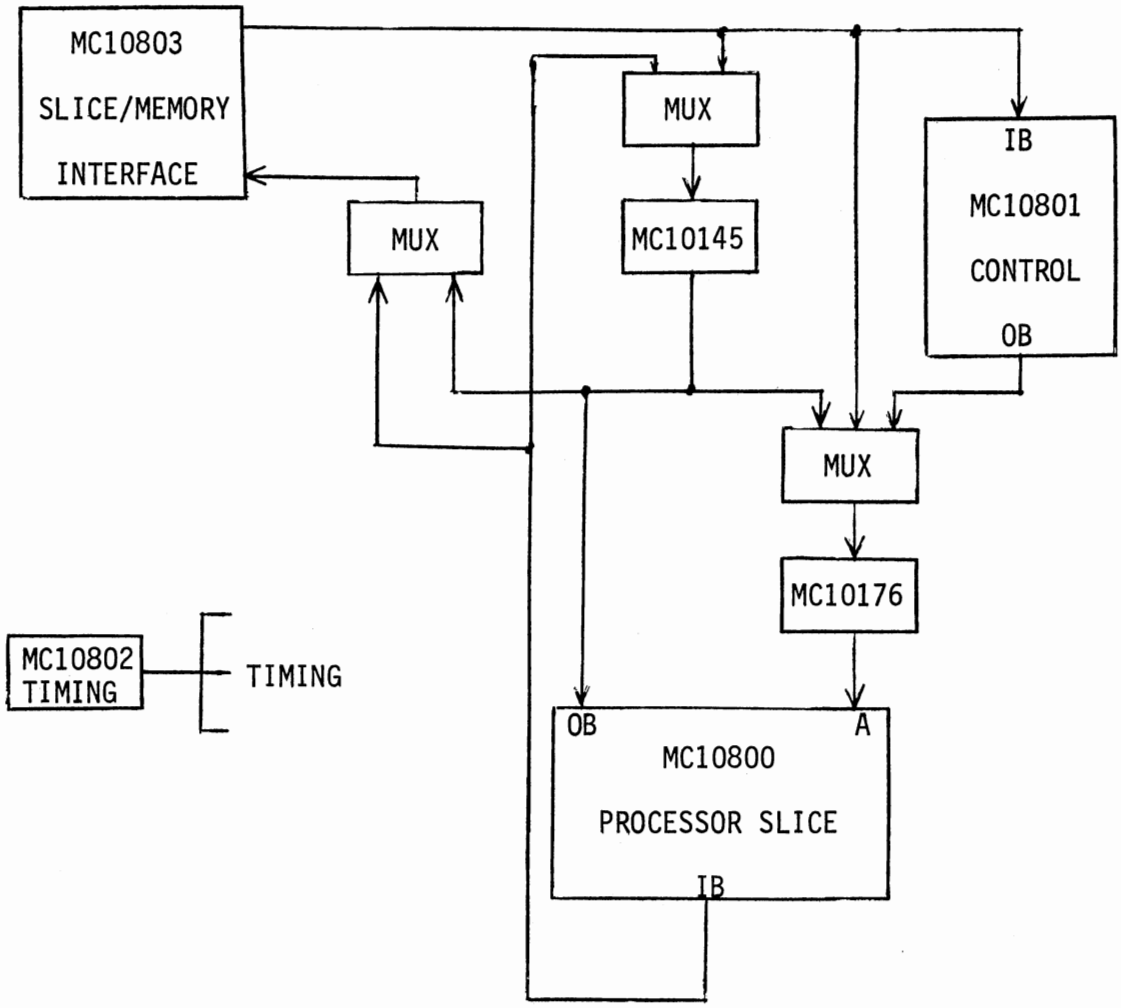


Figure 20. M10800 Family - Pipeline Processor Example

4.4.5 Comparison

The Fairchild 9400 series and the Motorola 10800 series are almost in a different class from the Intel and AMD processor families. The 9400 and 10800 series both break up the processor into smaller system blocks requiring more packages to build a basic processor. Of these two, the MC10800 series seems to have two major advantages. The first is the increased speed provided by the use of ECL technology and the

second is a better architectural arrangement. The MC10800 series has a more powerful ALU and the registers are added externally. This means one can design the system to have the number of registers required by the application. Neither manufacturer is planning to supply software support.

The major disadvantage of both processors is the increased development manpower and external hardware required. Also the MC10800 has a disadvantage of working with a new technology. The 9400 series has no real system advantages and is slower than the Am2900 series. The MC10800 system is impressive but is not scheduled to be released until some time in 1976.

The choice for the fire control processor lies between the Intel 3000 series and the Am2900 series. The Am2900 series has a faster cycle time, a simpler architecture, a more complete chip set, more powerful ALU functions and more registers. The 3000 series has a complicated addressing scheme which reduces the bit size of the micro-instruction but increases the software development time. It's greatest advantage is an existent cross assembler for the ALU chip. This assembler is only a microcode assembler and a second macro-assembler is still needed.

Further examination of the instruction sets also shows the Am2900 to be stronger. This type of comparison is difficult because each manufacturer presents the instruction set in a different form. The Intel 3002 instruction set is shown with a mnemonic for each function because Intel has an assembler for their instruction set. The Am2901 shows the instructions as a function of the bit code and it is broken into three groups. The 3002 appears to have more functions but actually it has fewer, and has less destination and source control. The most

obvious difference is the absence of subtraction in the 3002 functions where the 2901 includes both R-S and S-R subtraction. Also according to the Advanced Micro Devices' engineers, the 2901 has 203 source operand combinations to the ALU, while the 3002 has 24 (23). Also the Am2901 has simultaneous shift and arithmetic operations, while the 3002 does not. Furthermore, the Am2901 has a more flexible register addressing mode where any of the 16 registers may be a source or destination of an ALU operation. The 3002 requires the use of the T or AC registers as the destination register. Also the 3002 is very difficult to subroutine on the microlevel requiring external hardware.

The above advantages have prompted the selection of the Am2900 series for application for the fire control system. The 3000 series seems to be suited mainly for controller type applications and not for general processing functions which are the primary needs of the present system. Thus Chapter V will begin the hardware application of the Am2900. The four microprocessor families discussed are summarized in Table XI. Also a comparison of the four families is shown in Table XII.

TABLE XI

MICROPROCESSOR FUNCTIONAL SUMMARY

Device	Description
Intel 3001 microprogram control unit	Maintains microprogram address register, selects next micro-instruction based on flags, instruction and present address. Completely different concept of address control from program counter approach.

TABLE XI (Continued)

Device	Description
Intel 3002 central processing element (2-bit slice)	Operations include ALU functions, data path manipulation, and register control. Three input and two output buses are available and 11 general purpose registers with one full function accumulator.
Fairchild 9404 data path switch (4-bit slice)	Only executes the functions of data path manip_ ulation which include dual 4-input multiplexer, a true/complement one/zero generator, and a shift left/shift right array. A 5-bit instruction word selects function.
Fairchild 9405 arithmetic logic register stack (4-bit slice)	Performs ALU functions and register control. Eight internal registers are provided. Register selection requires three bits and the instruction word is three bits.
Fairchild 9406 program stack (4-bit slice)	Provides return address storage for nested subroutines. Executes four instructions: Return, Branch, Call, and Fetch. Program stack is 16 x 4 bits.
Fairchild 9407 data access register (4-bit slice)	Contains three 4-bit registers intended for program counter, stack pointer and operand address. Implements 16 instructions for address control.
Am2901 microprocessor slice (4-bit slice)	Performs ALU, register control and data path manipulation between registers and ALU. Instruction word is nine bits, and two 4-bit words select register address. 16 general purpose registers and one Q-register are provided.
Am2909 microprogram sequencer (4-bit slice)	Performs address control with an instruction register, stack pointer, 4-word stack, and microprogram counter register. Branch, program counter increment and subroutine function are all controlled by the 2909.
MC10800 microprocessor slice (4-bit slice) (ECL)	Performs ALU functions with only data latches and an accumulator as internal registers. Very powerful ALU instruction set with 16-bit instructions. A register file must be externally provided using a RAM.

TABLE XI (Continued)

Device	Description
MC10801 control function (4-bit slice) (ECL)	Performs microprogram address control including status, branching and interrupt operations.
MC10802 timing function (ECL)	Ties other system blocks together by providing various clock phases and control.
MC10803 slice/ memory interface circuit	Performs main program address control and allows for more complex addressing techniques.

TABLE XII

MICROPROCESSOR COMPARISON

Manufacture Model	Intel 3002	Fairchild 9405	Am 2901	MC 10800
Structure	2-bit slice	4-bit slice	4-bit slice	4-bit slice
Technology	TTL	TTL	TTL	ECL
Supply voltage (V)	5	5	5	-5.2, -2.0
Pwr. diss. (mW)	800 x 2	470	925	1300
Other main system parts	3001	9404, 9406, 9407	2909	MC10801 10802, 10803
Cycle time (ns)	150		125	55
Clock frequency	6 MHz	10 MHz	10 MHz	
Phases required	Single	Single	Single	Double
No. of registers	11	8	17	External RAM
Return stack size	None	16 (9406)	4x4 (2909)	
BCD arithmetic	No	No	No	Yes
Microprogrammable	Yes	Yes	Yes	Yes
Software support	Yes	No	No	No
Simultaneous shift and arithmetic	No	Yes	Yes	
Number of microcode control inputs	7	9404-5 9405-3 9406-2 9407-4	9	16
Temp. range	-55°C to 125°C	-55°C to 125°C	-55°C to 125°C	0°C to 75°C
Availability	1972	1975	1975	1976
PINS	28 x 2	24 each	40	48

CHAPTER V

APPLICATION OF THE Am2900 SERIES

5.1 Introduction

The selection process has been completed and the next objective is to define the development time and required resources. The hardware and preliminary micro-firmware specification will be investigated and project flowcharts outlined. The project flowcharts show the development phases and required parallel development of firmware and hardware. Chapter VI will be devoted to the project flowcharts and overall foreseeable problems. At this point preliminary hardware design will begin for the purposes of showing system complexity and to obtain an approximate package count. The hardware design will also produce a preliminary microcode definition. Table XIII defines the system elements which will frequently be referenced by their associated abbreviation.

5.2 System Specification and Design

At this point one must make primary system decisions. The address bus size, data size, macro-instruction length, micro-instruction word and many other factors must be defined. The design goals are to outline a system which has a general and flexible microcode function base with expandable ROM and RAM capabilities. Possible system debug problems must also be anticipated during design.

TABLE XIII

SYSTEM ELEMENTS

Name (Abbr.)	Definition
Program counter (PC)	One of 16 internal registers of 2901
Instruction register (IR)	16-bit macro-instruction register
A address register (A)	3-bit section of IR for A address
B address register (B)	4-bit section of IR for B address
Memory address reg. (MA)	16-bit hardware register
Memory data register (MD)	32-bit hardware register
Memory data reg. 1 (MD1)	First 16 bits of MD
Memory data reg. 2 (MD2)	Last 16 bits of MD
Microprogram counter (MPC)	Register internal to 2909
Micro-instruction reg. (MIR)	Pipeline register
Read only memory (ROM)	Microprogram memory
Random access memory (RAM)	Working memory
Either ROM or RAM (MEM)	Total memory system for macro system

The starting point is the basic block outline shown in Figure 21. The bus system must be defined from data and instruction length requirements. The micro-instruction word will consist of several functional fields which will be defined as required during hardware design. The macrocode will only consist of three fields. One field is required for the ROM starting address and two fields to specify the CPU register addresses. The macrocode word is not decoded by hardware but will include a field which is the ROM address of the microcode program which executes the macro-instruction. The macrocode word will be limited to 16 bits which means nine bits will be ROM address and seven bits will specify the CPU register addresses. The Am2901 requires four bits for each A and B register address selection. Because there are only seven

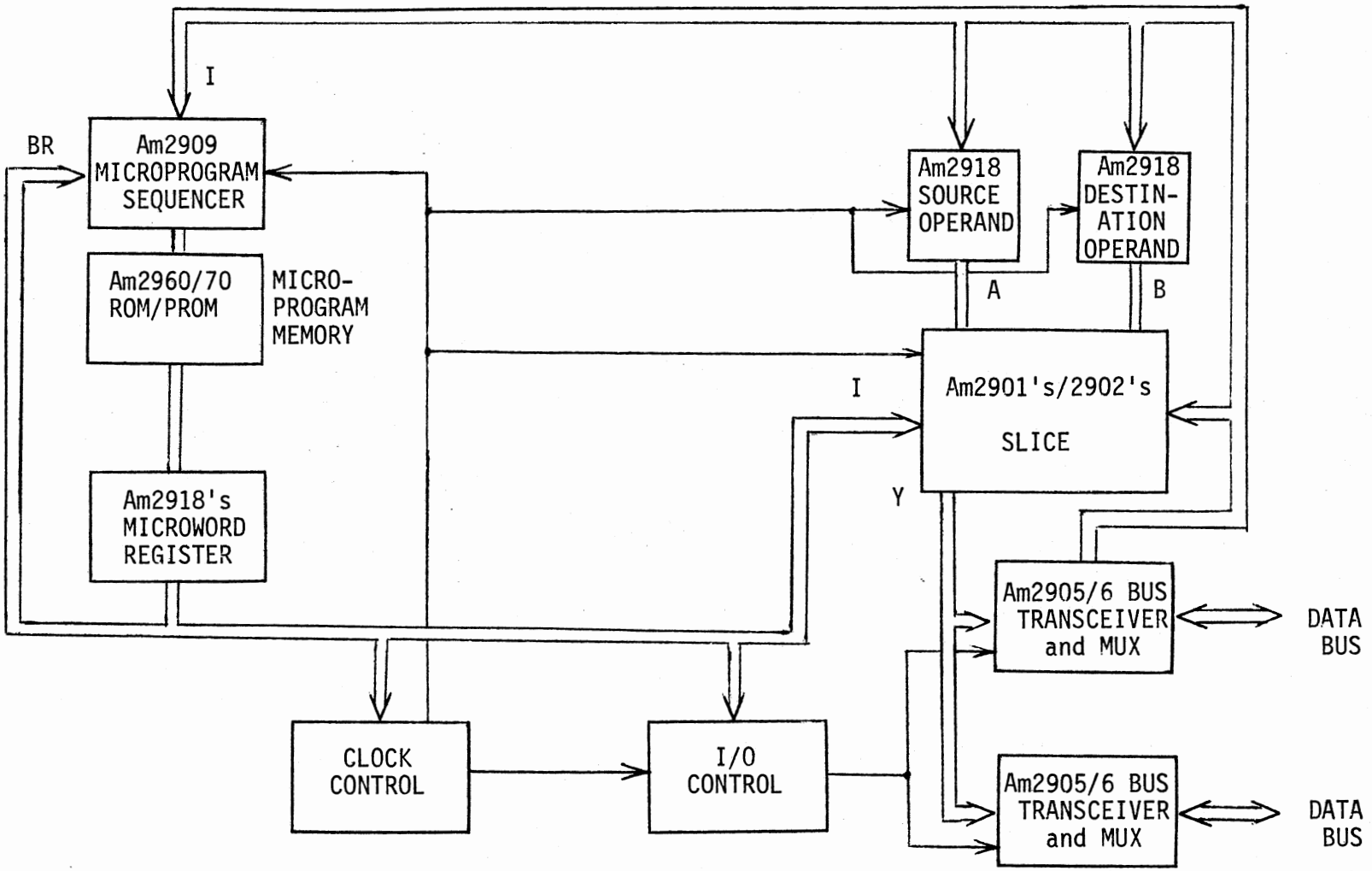


Figure 21. Basic Block Outline

bits, complete access to all 16 registers through address B is possible but only eight through address A. The Am2901 allows one to address through A and B buses, one of 16 internal registers as input to the ALU (see Figure 17). The ALU results are written back into the register selected by address B. The macrocode word is shown below.

ROM STARTING ADDRESS	ADDRESS A	ADDRESS B
nine bits	three bits	four bits

(maximum of 512 micros)

The above macro-instruction is the first and primary instruction but it may be followed by two more instruction words. The second word will be either a 16-bit address for memory reference instructions or part of a 32-bit constant. A third byte will be the second half of the 32-bit constant. Thus a maximum of three macrocode words may be accessed during an instruction execution. The ability to read a 32-bit constant from macrocode program memory allows the programmer to load internal registers with either a decrementing constant for loop control or a masking constant.

5.2.1 Memory System

The external memory system will consist of RAM working memory and ROM macroprogram memory. This is not to be confused with the micro-instruction memory which will also be in ROM but considered internal memory. The address bus structure is shown in Figure 22. The actual addressing requires 14 bits for the ROM of which eight are shared with the RAM memory. The ROM system is divided into 1K sections. Ten bits

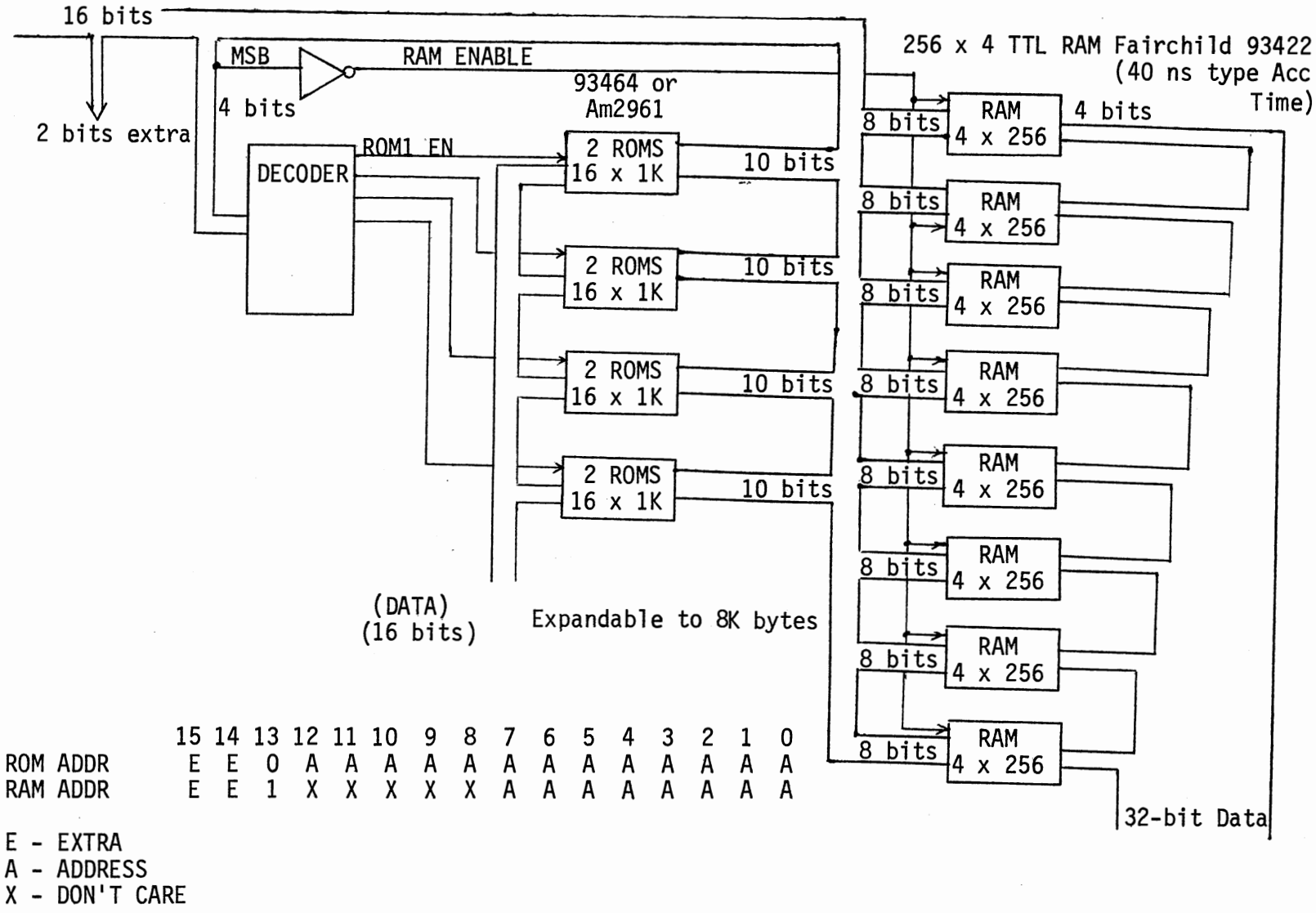


Figure 22. Memory System

are used to address each of the 1K ROMs and four bits are used to select a particular chip. This arrangement allows preliminary work to start with a rough estimate of ROM size and add new ROM when necessary in 1K increments up to 8K bytes. The RAM system will provide 256 bytes of 32-bit data which can also be increased to 1K if necessary. The addressing scheme is also shown in the figure. Address 0 to 1FFF (hexadecimal) will select ROM while address 2000 to 3FFF will select RAM. Two extra bits are left for I/O control of radar data.

The I/O bus control system is shown in Figure 23. Am2905s (or 2906s if parity is required) will be used for bus drivers and control. The 2905 block diagram is shown in Figure 24. This system enables four types of data transfer:

1. 32 bits of data from RAM to ALU
2. 16 bits of instruction from ROM to instruction register
3. Two bytes of 16 bits of instruction from ROM to ALU
4. 16 bits of address from ROM to memory address register (MA)

Also because each 2905 has internal data latches they will also function as the memory data register (MD). The 2905 which controls the memory address bus will be used as the memory address register (MA). The 2905 block diagram shows the control lines available which include an input select line, buffer clock line, an output enable, an input enable and a load buffer line. The output buffer is clocked by an edge and the input buffer is loaded by a level. Each of these lines must be controlled as a function of the microword to perform the desired output states. The design of this circuit will be discussed next.

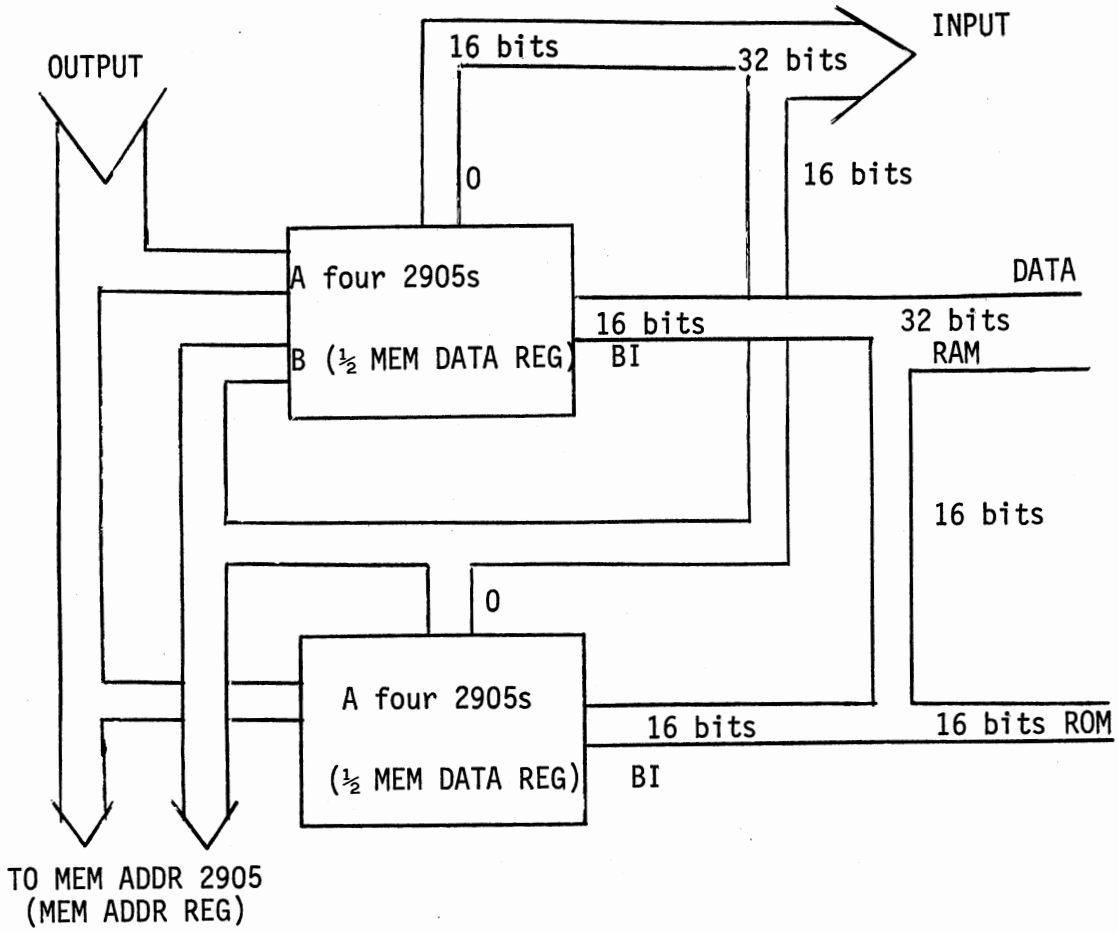


Figure 23. I/O Bi-Directional Bus System

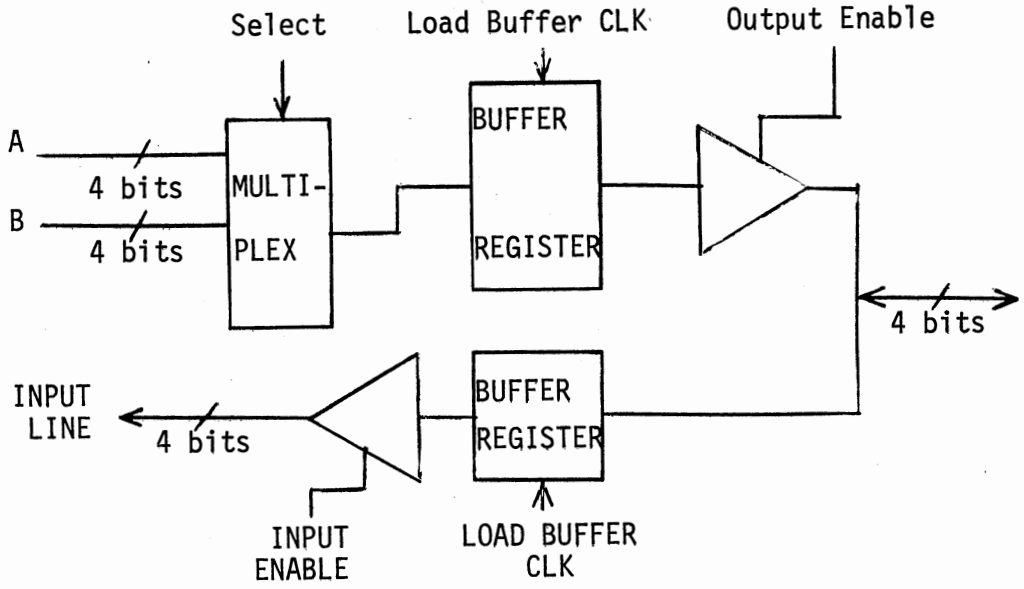


Figure 24. 2905 Block Diagram

To specify the I/O control functions one must first define the types of I/O operations to be performed and relate them back to the system clock. Certain types of instructions and their associated required I/O functions are shown below.

TABLE XIV

I/O OPERATIONS

Instruction	I/O Function	I/O Path
All instructions	Load MA Fetch instruction	(ALU) PC to MA MEM to MD2 to IR
Mem reference	Data address fetch	MEM to MD2 to MA
Load immediate	Fetch two word constant	MEM to MD2 to MD 1 MEM to MD2
Mem reference	Fetch 32-bit RAM data Output result	MEM to MD ALU to MD

The table shows that several I/O operations may be required by a single instruction. The problem becomes how to divide the I/O functions. For example, one may make a single transfer, memory to MD2, one cycle and the transfer, MD2 to IR, the next cycle or more operations may be performed in a single cycle. The logical breaks for I/O operations are determined by when data are available and the time delays within the system. The above is given where each I/O path shown is

one cycle of microcode execution time. These functions give the best use of the available I/O capabilities of the 2905 and only require a single phase clock. Further combining of the operations into more powerful functions would require multiphase clocks and a slower cycle time. Multiphase clocks were avoided by using the 2905's multiplexer capabilities to shorten I/O paths.

Hardware for the control of each I/O operation as a function of the system clock and the microcode word must now be designed. The hardware design is shown in Figures 25 and 26. The design is basically centered around the use of a 4 to 16 line decoder chip. The design utilizes the tradeoffs between the microcode bit assignments and the decoder output. The seven required states could have been defined using only three microcode bits but a fourth was added so large numbered input gates could be eliminated in the decode hardware. Also this system is easily expandable with only seven of the available 16 lines used. Finally, the system debug problem was simplified by the use of the decoder. Each I/O state is now defined by a single lead.

The system timing had to be specified during the I/O control design for each microcode execution cycle. A 5-megahertz (MHz) system clock or 200 nanoseconds (ns) cycle time was chosen based on the set-up and delay times within the system. The Am2900 family chips are new and not well specified as to timing. The only hard specification is 135 ns delay from the A or B address inputs of the Am2901 to the ALU output Y. This delay is a maximum for the Am2901DM or military temperature range device. The cycle time must be greater than 135 ns + delay of input registers. Most of the registers are clocked on the raising edge of the system clock cycle. The timing is shown in Figure 27.

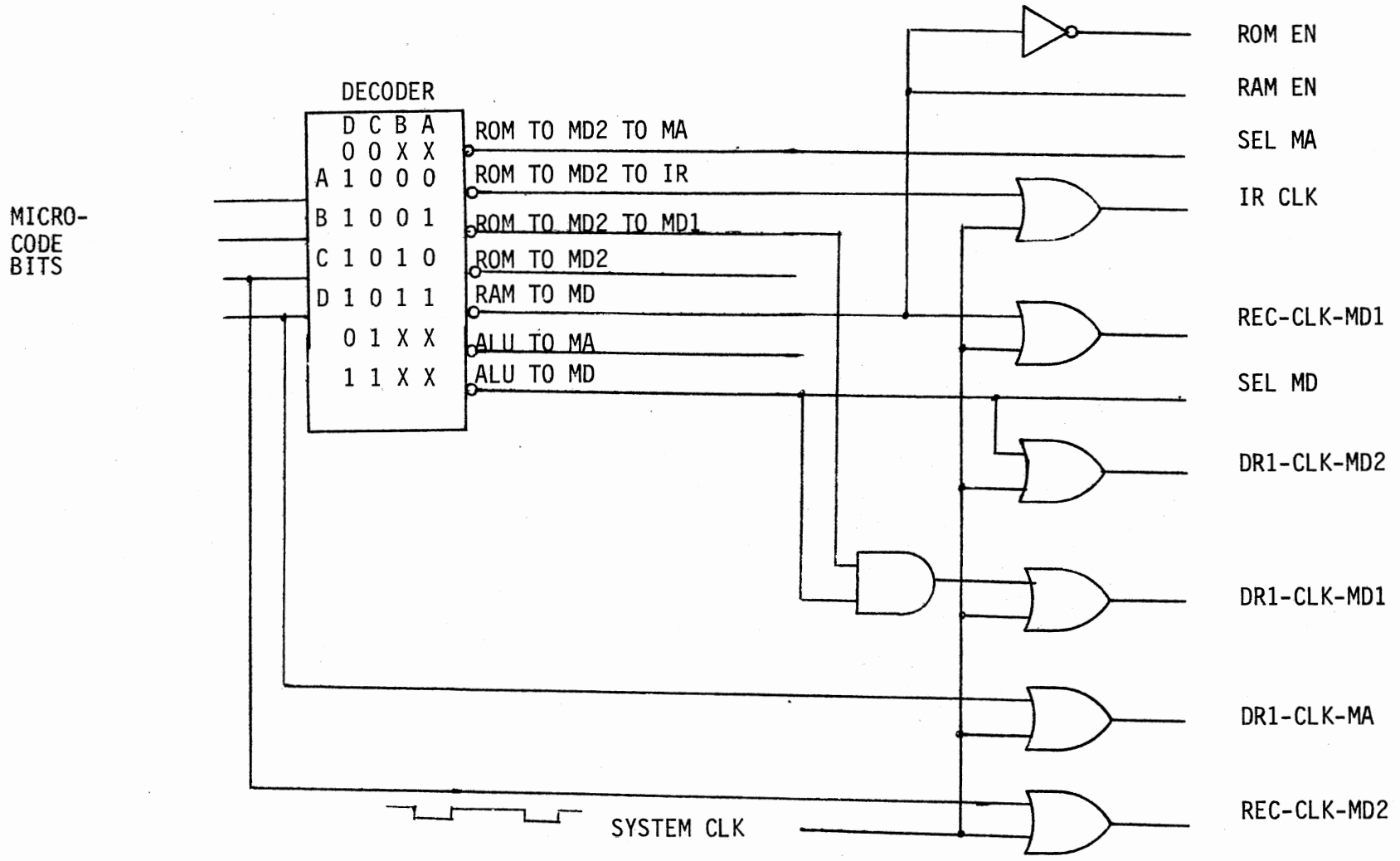


Figure 25. Input/Output Control Circuit

This figure shows a simplified timing scheme where each register load operation is shown. The timing was not derived in detail and is not meant as a final design. The basic considerations were that the 200 ns cycle time be greater than the 135 ns ALU delay plus worst case delay of the other internal registers. The worst case delay of the registers was not available but it is felt that 200 ns cycle time is more than adequate. The other consideration is that the negative edge to positive edge time of the system clock be greater than 30 ns, an Am2901 requirement. This delay must also be great enough to allow MD2 to be loaded at the negative edge and MD1 to be clocked at the positive edge, thus allowing a single cycle transfer of data from the ROM to MD1, IR, or MA.

Once a general timing scheme was derived, the I/O control circuit was completed by producing the required control signals from the DECODER and the system clock. The two lines RAM EN and ROM EN are the only lines not going to the Am2905 I/O chips. These lines are required to control the tri-state bus which the ROM and RAM memories share. The lower order 16 bits of RAM data bus are shared with the ROM data bus.

A separate microcode bit will be used for RAM read/write control as shown in Figure 26. It also controls the I/O direction on the RAM data bus.

5.2.2 Branch Control

The Am2901 provides three bits of status information which can be used for conditional branch operations. Figure 28 shows a circuit which provides the indicated operations based on three microcode bits. If more branch operations are required, a larger multiplexer could be

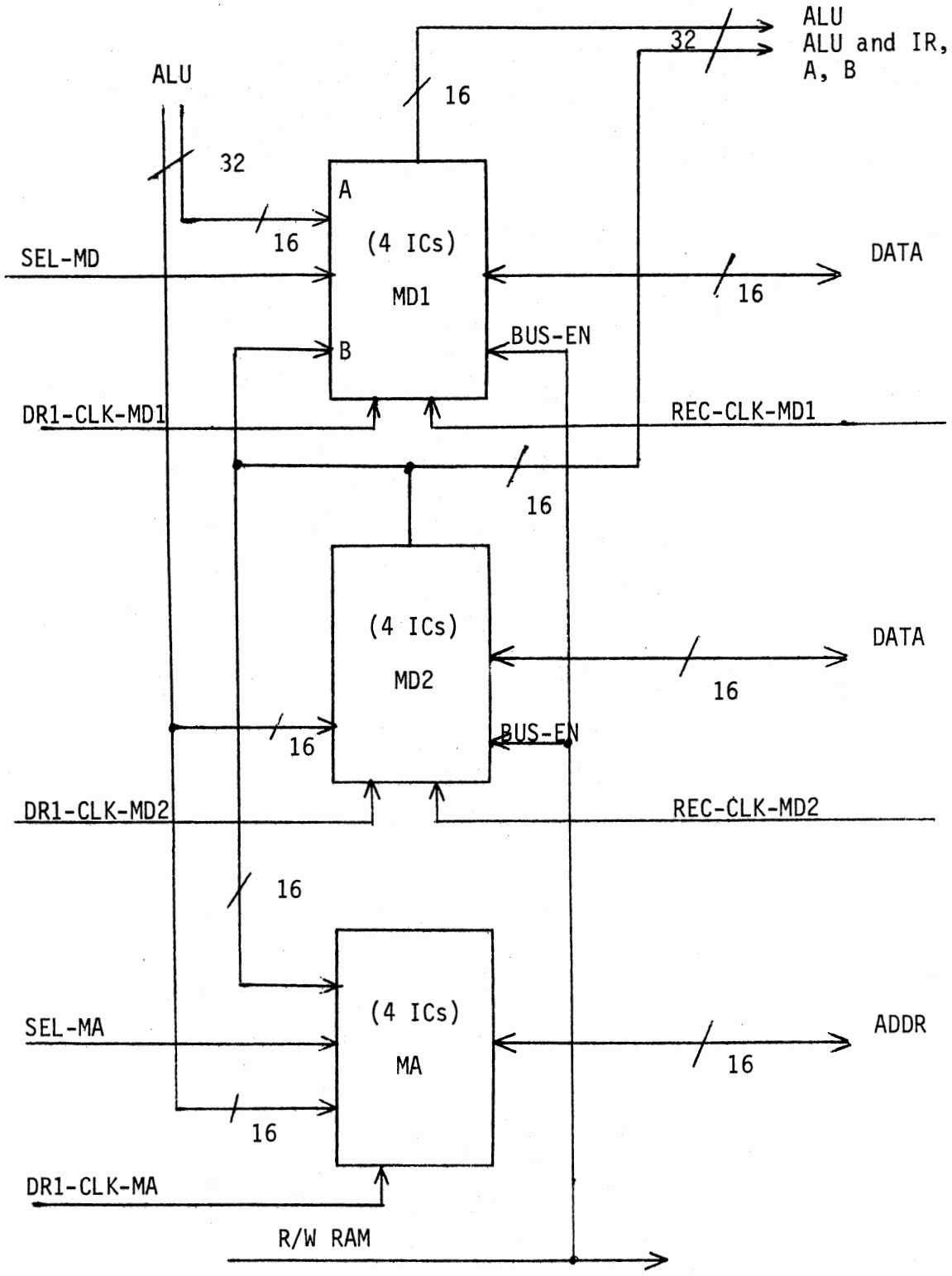


Figure 26. Input/Output Bus Circuit

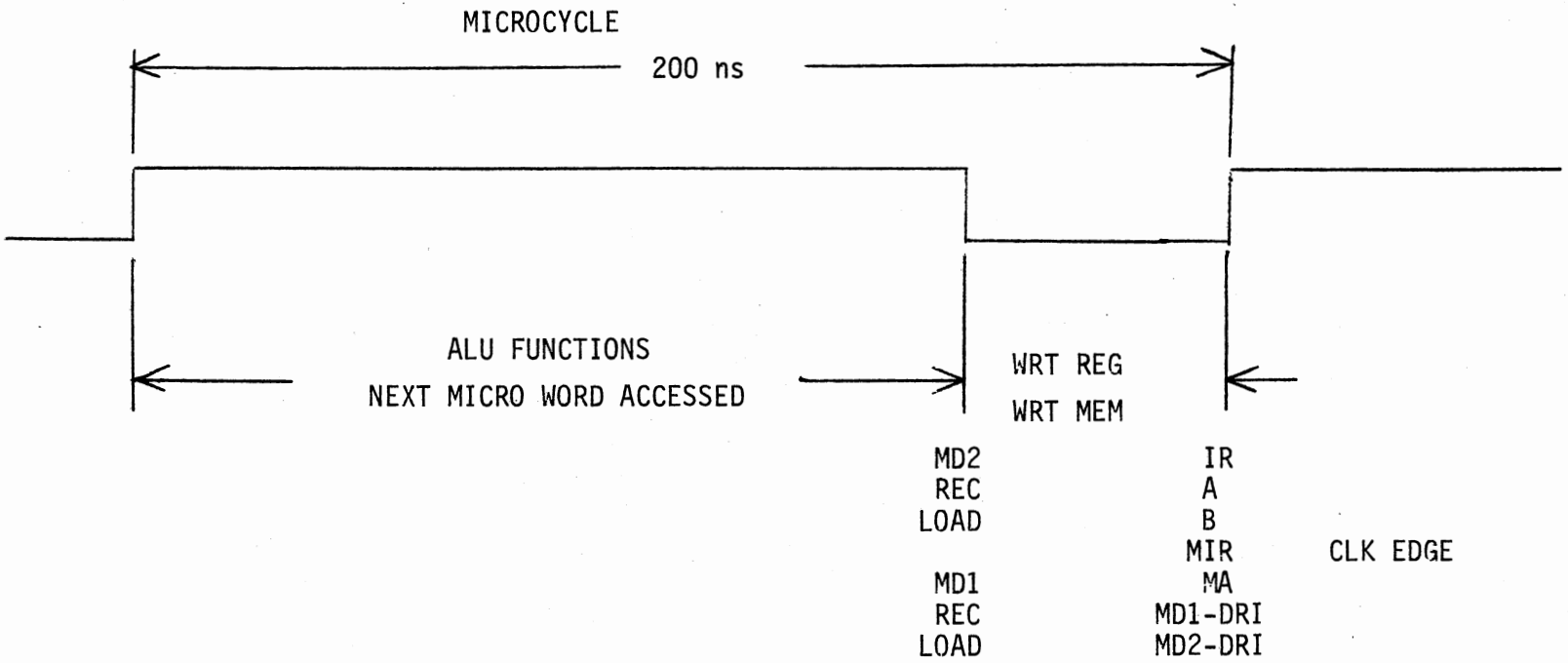
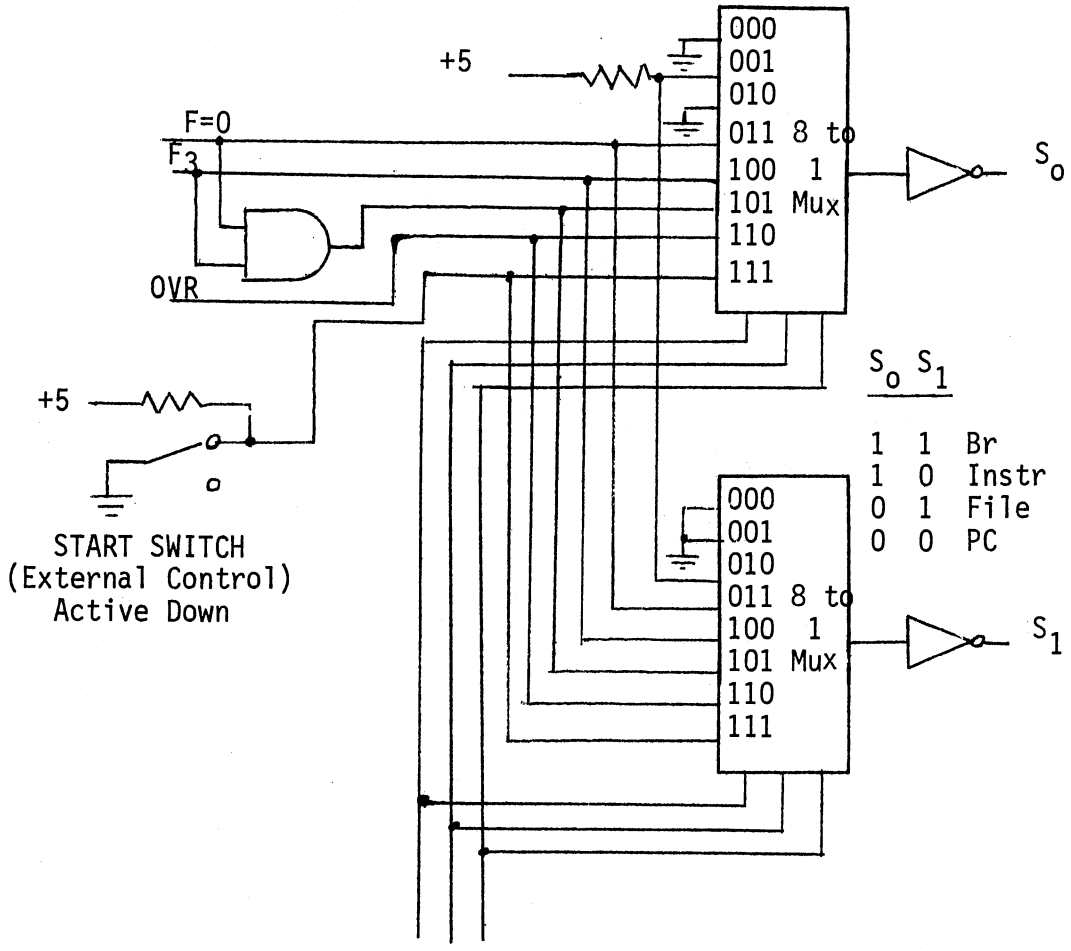


Figure 27. Microcycle Timing



0 0 0 Program Counter
0 0 1 Instruction Register
0 1 0 File
0 1 1 Br if F=0 otherwise PC
1 0 0 Br if F_3 otherwise PC
1 0 1 Br if F=0 and F_3 otherwise PC
1 1 0 Br if OVR otherwise PC
1 1 1 Br if START otherwise PC
Br unconditional after START = 1

Figure 28. Branch Control Circuit

used at the expense of an additional microcode bit. The outputs, s_0, s_1, feed into the Am2909 microprogram control chip. These two bits are multiplex select lines internal to the Am2909 which select one of either the micro-instruction register, the microprogram counter, the internal register file or an external input. The external input is connected to the micro-instruction register branch field so that one may select the branch address from the microword. The selected lines are then multiplexed into the microcode memory address lines.

Five branch conditions are provided including a branch on start bit which can be used for external control. Once the switch is activated it becomes an unconditional branch. Thus, this branch condition provides a firmware start condition which, once active, can be used as an unconditional jump instruction.

5.2.3 Firmware Register Control

At least some of the 16 registers within the 2901 will be used as special purpose registers such as a program counter, stack pointer or status register. Firmware must be able to access these special purpose registers while maintaining the macrocode's ability to specify registers. The circuit shown in Figure 29 provides firmware control of three special purpose registers while the macrocode still specifies an external address. The fourth input shall be the macrocode's specified address. Thus firmware may select one of three special registers or the macrocode specified register. This system requires two multiplexer packages and two microcode control bits.

A second method is to use a 2-input 4-bit multiplexer to select either the macrocode address or a microcode address. One microcode

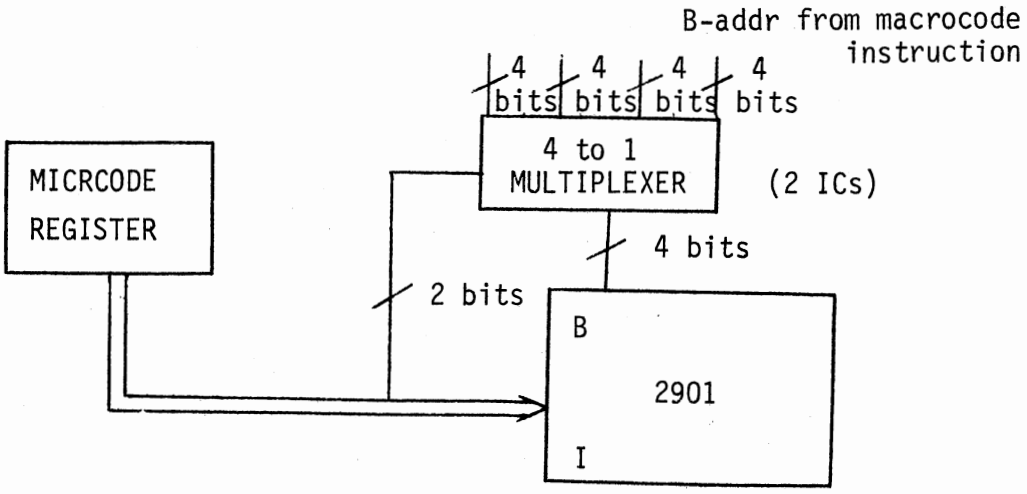


Figure 29. Firmware Control Using 2 bits of Microcode

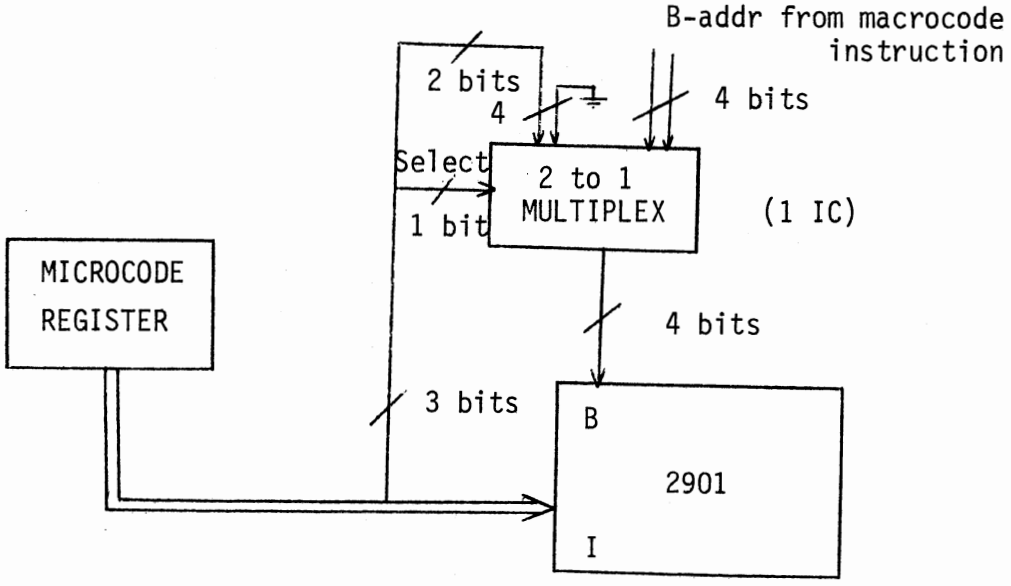


Figure 30. Firmware Control Using 3 bits of Microcode

bit is required for selection and up to four additional bits, depending on the number of firmware controlled registers required. If only four special purpose registers were required, three microcode bits would be necessary. This system requires only one multiplexer package and is shown in Figure 30.

If a status register is actually going to be implemented it would require further hardware to input status information into the register. Input multiplexing and probably one bit microcode control would be required.

5.2.4 Pipeline Register Operation

The pipeline register or micro-instruction register (MIR) is used so that the next micro-instruction can be fetched during the execution of the present instruction. Table XVI shows the pipeline operation during normal program execution. Explanation of the symbols used is given in Table XV.

TABLE XV

DESCRIPTIVE SYMBOLS

$\leftarrow$	load into
+1	increment
α^n/reg	left-most n bits of register
ω^n/reg	right-most n bits of register
( )	as a function of
-1	decrement

TABLE XVI

FIRMWARE EXECUTION WITH PIPELINE REGISTER

Address	Am2909 Functions	Am2901 Functions
0	MPC ← MPC + 1, MIR ← ROM (MPC)	PC ← PC + 1, MA ← ω^{16}/PC
1	MPC ← MPC + 1, MIR ← ROM (MPC)	IR ← MEM (MA)
2	MPC ← α^{9}/IR + 1, MIR ← ROM (α^{9}/IR)	PC ← PC + 1, MA ← ω^{16}/PC
	Instruction Execution	
	If two word instruction: MPC ← MPC + 1, MIR ← ROM (MPC)	ALU or MA ← MEM (MA)
	If one word instruction: MPC ← MPC + 1, MIR ← ROM (MPC)	PC ← PC - 1
	Execution of Instruction	
	End: MPC ← 1, MIR ← ROM (0) or MIR ← ROM (File)	(Last function required)

Lines 0, 1, and 2 are the instruction fetch cycle in microcode. The Am2909 functions are those actions taken by the micro-controller, and the Am2901 functions are the macro-instruction execution control. Microprogram control is not transferred to the macro-instruction register until line 2, for this is the point where the first macro-instruction is available. At this point the PC is loaded into MA so that if the instruction is multiword a continuous fetch can take place. If the next instruction is a single word, the PC is decremented to its original value because the memory access is not done. Lines 0, 1, 2 and the last line are common to all instructions. Line 2 is a jump to the particular instruction's execution code based on the contents of IR. The last line shows the transfer of control either back to the next macrocode fetch or a subroutine return to a microcode execution point.

Conditional branch instructions require special clock control because the next micro-instruction address is a function of the present execution results. On a conditional branch instruction the next microcode fetch is halted until the present execution is complete. Then the ALU clock must be stopped while the next microcode word is fetched. Thus one cycle time is lost during a conditional branch instruction execution. Table XVII shows the functional timing and Figure 31 shows the clock timing. Finally, Figure 32 shows the required hardware to control the system clocks during this process.

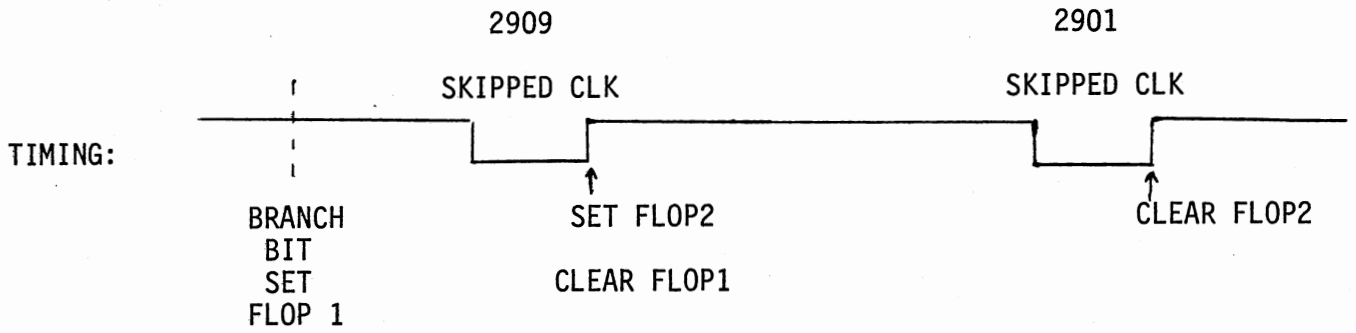


Figure 31. Clock Timing During Conditional Branch

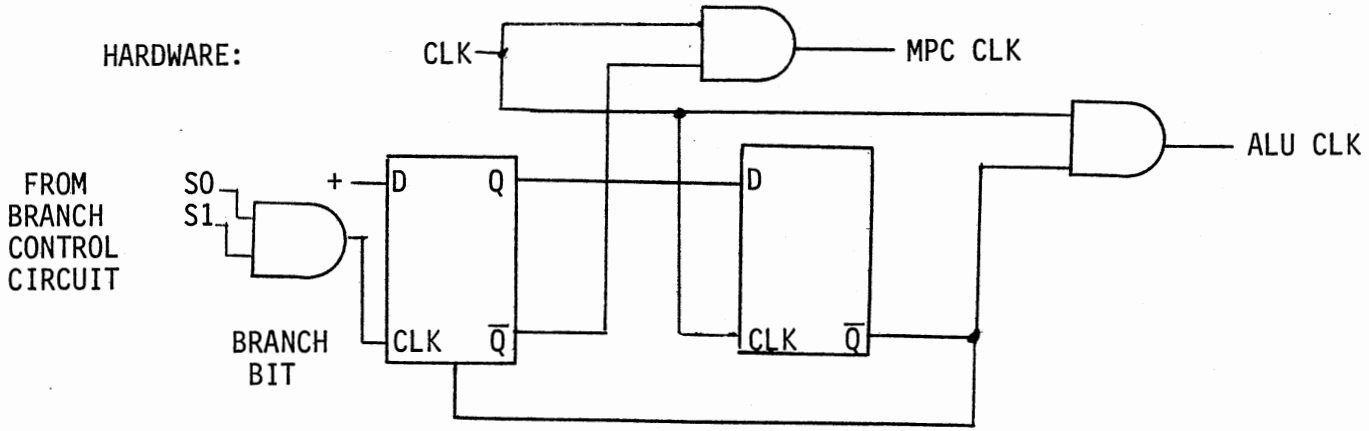


Figure 32. Hardware for Clock Control for Conditional Branch

TABLE XVII

EXECUTION OF CONDITIONAL BRANCH INSTRUCTIONS

Am2909 Functions	Am2901 Functions
No new MPR load, stop MPRCLK Branch on results (same μcode) Continue	Function loose 1 cycle time Stop ALUCLK Function

5.2.5 Shift and Rotate Functions

Shift and rotate hardware are discussed in Am2901 application literature. These functions are discussed as part of the multiplication scheme used by the Am2901. The present application requirements are for a zero shift function for data scaling but if the Am2901 multiplication scheme is used, two additional shift functions are required. The external circuit consists of two multiplexers which allow a right or left shift of one of four types. The four types are zero, one, rotate and arithmetic. The first three are the familiar shift functions offered by most processors, while the arithmetic shift is used specifically during a multiply routine execution. On a right arithmetic shift a zero is loaded into the Q-register least significant bit (LSB), and the Q-register most significant bit (MSB) is loaded into a selected ALU register's LSB.

The shift control circuit is shown in Figure 33. This circuit requires two microcode bits for shift type selection. The direction of the shift is determined by I_7 of the Am2901 instruction code.

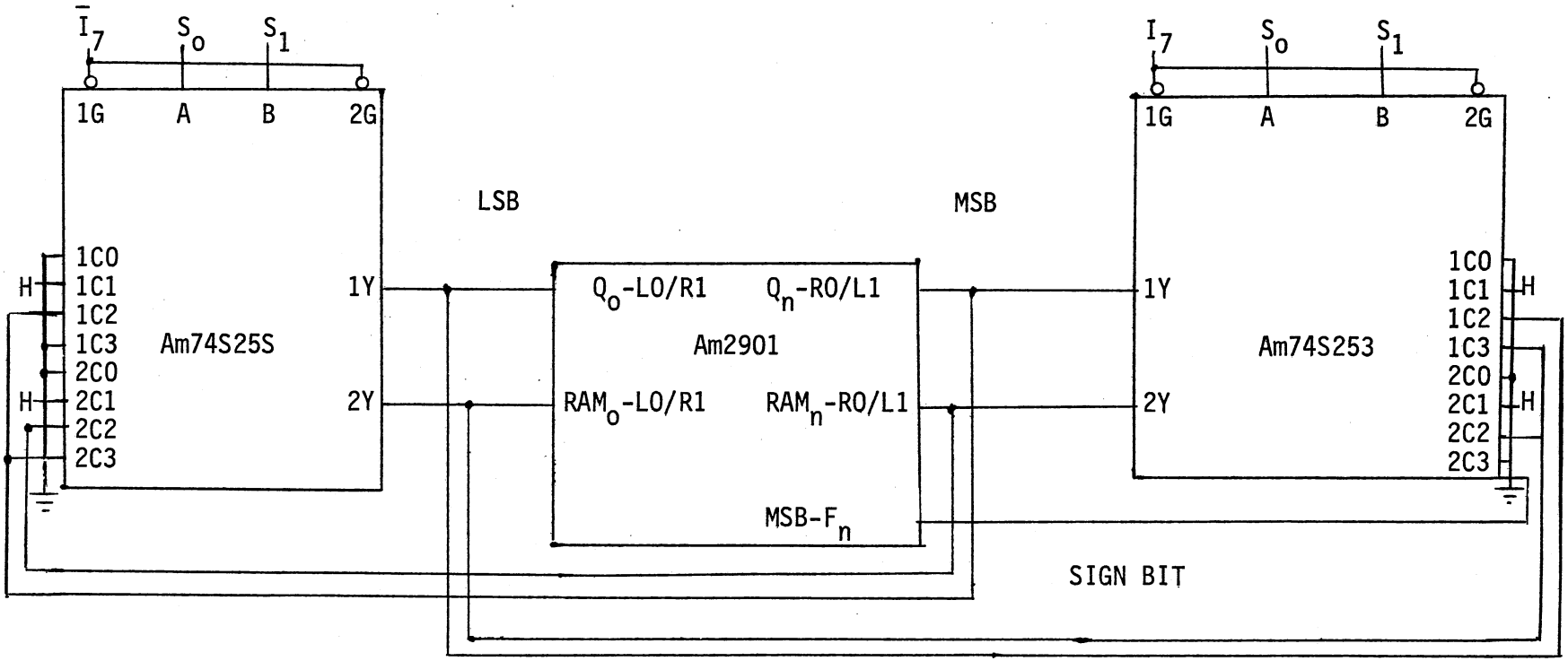


Figure 33. Shift Control Circuit

The microcode bits I_6, I_7 and I_8 are the ALU destination control. These also control the shift operation within the Am2901. Table XVIII shows the bit codes required for each combination of type and direction of a shift operation. The Am74S253s are dual 4-input multiplexers with tri-state outputs.

TABLE XVIII

SHIFT CONTROL CODE

Code			Direction	Type
I_7	S_1	S_o		
0	0	0	Right	Zero
0	0	0	Right	One
0	1	0	Right	Rotate
0	1	1	Right	Arithmetic
1	0	0	Left	Zero
1	0	1	Left	One
1	1	0	Left	Rotate
1	1	1	Left	Arithmetic

5.2.6 Multiplication Hardware

It has been determined previously that some type of hardware multiplication scheme is necessary to increase the throughput of the system. The Am2901 already has time sequenced type hardware multiplication capabilities without external hardware. The basic technique used is the "add and shift" algorithm. The multiplier and multiplicand are in R_o and R_1, respectively, and the result is placed in R_2 and R_3. The

exact firmware and hardware details are given in the Am2901 applications literature. The multiplication execution requires 37 cycle times for a 32-bit multiplier. Thus 7.4 μs are required for one multiply.

The other scheme mentioned was the use of Am25S05 multiplier chips for a completely hardware multiply. This requires 128 additionsl chips and six cycle times for execution. Two cycles are required for instruction fetch, one loads the multiplier register, one outputs the multiplicand to the Y outputs of the ALU, one is a no-op cycle for time delay and the last cycle time loads the result into an ALU register. Execution requires 1.2 μs. Figure 34 shows a block diagram of the required system changes for this arrangement. The number of Am25S05s required can be reduced from 128 to 78 if a truncated result is acceptable. This is not to be confused with a rounded-off result which would also be 32 bits. Rounded results require 128 chip arrangement. Truncation also has no speed advantage.

One very important detail has been ignored during the multiplication hardware discussion. This is the method in which data is to be represented. Both of these schemes will require some type of scaling depending on the data representation. Many ways of data scaling exist. The data can be represented as a fixed integer, as a scaled integer or as a scaled fraction. The data representation choice must be made by looking at the data form available from the radar and the operations to be performed on the data. These aspects of the design are not specified so the multiplication hardware cannot be discussed in detail.

The time sequenced method is fast enough if the scaling requirements are not extensive. The scaling must not require more time than the multiply operation. These scaling requirements can also be handled

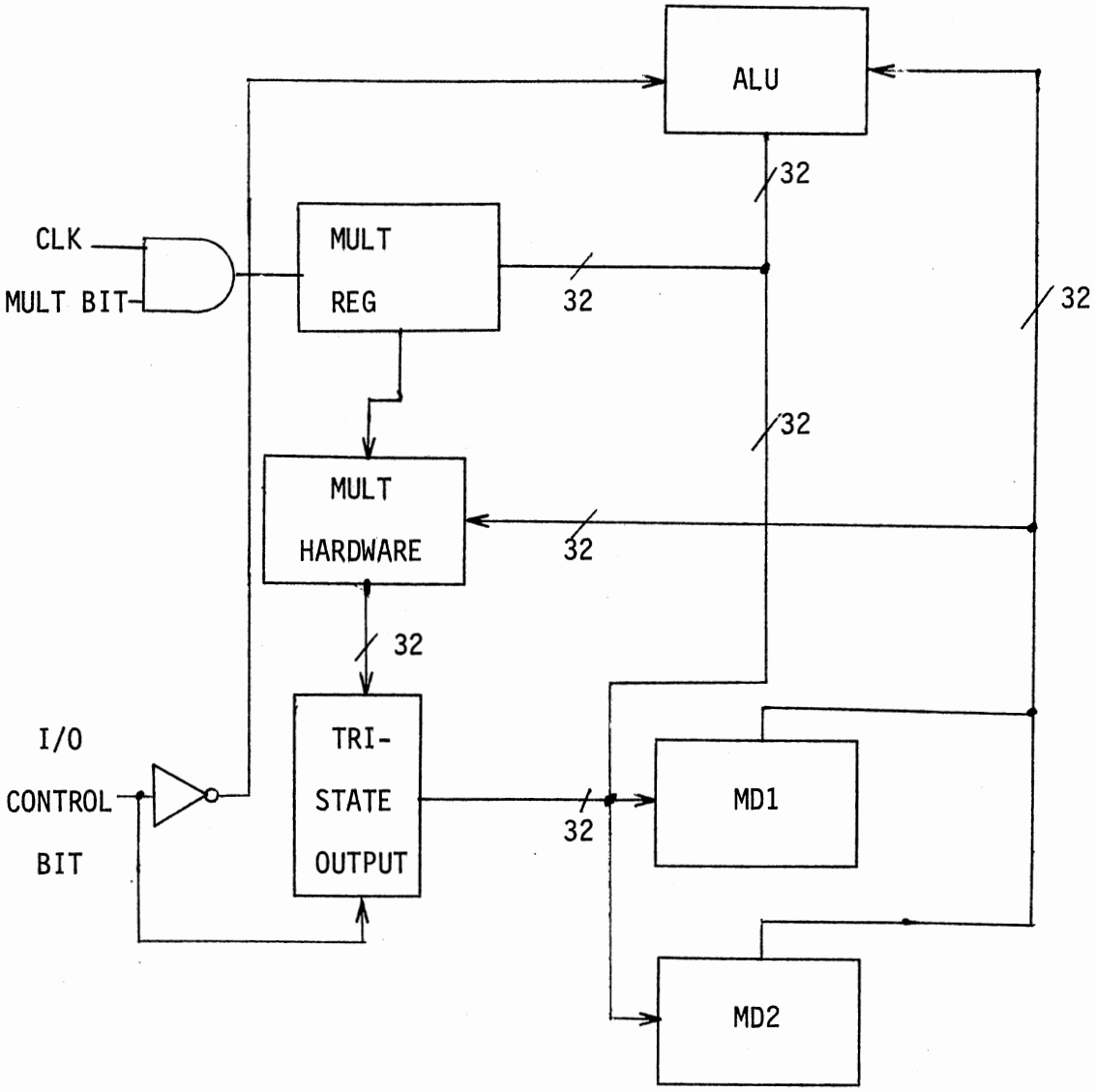


Figure 34. Multiplication Hardware Modification

by hardware if necessary. The other multiplication scheme is approximately six times faster but still requires scaling. Another advantage of the Am25S05s is that they can also be used as a hardware squareroot generator and reciprocal evaluation. The required additional hardware would have to be determined.

Both schemes give correct results for two's complement form multiplier and multiplicand. The question is whether the data must be scaled and whether it will be in a fractional or integer form.

Finally, a similar scheme for a time sequenced hardware divide is also applicable to the Am2901. The method is actually described in Monolithic Memories 6701 application literature. A non-restoring two's complement divide is produced by successively subtracting the divisor from the dividend and the result shifted left until the remainder changes sign. At this point the divisor has been subtracted one too many times and is added back until the original sign is restored. Thus the divide requires approximately the same execution time as a multiply.

The remaining required functions should be firmware implemented, devoting most of the hardware resources to fast multiply. Algorithms for the execution of each function should be investigated and determined so that exact multiply speed requirements can be determined.

5.2.7 Macrocode and Microcode Summary

The macrocode words are summarized in Figure 35. A maximum of three instruction fetches may be required with a minimum of at least the first macrocode.

The microcode fields are summarized in Figure 36. The hardware has been specified to the point where an approximate microword may be

Macrocode

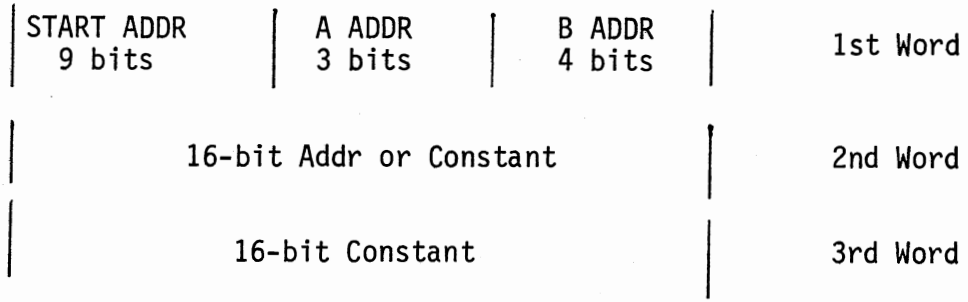


Figure 35. Macrocode Summary

Microcode

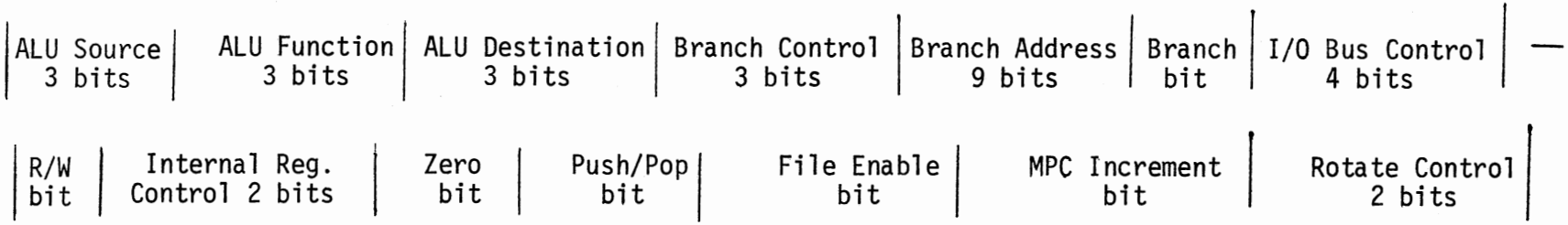


Figure 36. Microcode Summary

defined. The microcode word consists of 14 fields or 35 bits. All fields have been previously discussed during hardware design except the branch control fields which include branch bit, zero bit, push/pop bit, file enable bit and the MPC increment bit. The MPC increment bit is the C_{in} input to the Am2909 microprogram sequencer. It simply controls the incrementer so that when high the microprogram address is incremented before being stored in the MPC register. The remaining bits control subroutine call and subroutine return execution.

The basic hardware system has been discussed except for subroutine control. More hardware may be required if micro-subroutine calls are placed under both macrocode and microcode control. This means a subroutine call to the microcode which executed a multiply could be processed without returning control at the end to the instruction fetch firmware. At present, when the execution of a macro-instruction is complete, the last micro-instruction jumps to next-instruction-fetch routine. The double control would allow either a jump to the fetch routine or a subroutine return if the code was accessed by a subroutine call instruction. This would require either setting a flip-flop to remember each subroutine call or just using a subroutine call for the execution of each macro-instruction. The latter method would slow the macro-instruction execution by adding two more cycles to each instruction.

This type of control would be very helpful in firmware programming the trigonometric and other functions requiring existing firmware. The trigonometric execution firmware would be able to use the multiply firmware without going to the macrocode level. The hardware control of this type of subroutine function would simply modify the jump to the instruction fetch routine based on whether the subroutine call

flip-flop had been set previously. The two bits push/pop bit and file enable are necessary since they are used by the Am2909 during subroutine function execution. The push/pop bit is the push/pop input of the Am2909 which controls the increment or decrement of the stack pointer. File enable bit is the read/write control of the 2909's file stack.

The two bits, branch and zero, are included for hardware control and may or may not be necessary depending on the actual hardware. The branch bit may be necessary in setting the subroutine jump flip-flop when the special subroutine jump is necessary. The zero bit was included since it conveniently allows a jump to address zero over any other control of the Am2909. Thus the special subroutine function has been anticipated in the micro-instruction word definition but the actual implementation was not completed. The final micro-instruction will probably differ but it will be within a maximum of 36 bits. The other fields can be decreased if it is desired to lower the instruction size to 32 bits. However, this would increase the decode hardware.

5.3 Approximate Instruction Execution Times and Package Count

Two additional results may be obtained from the hardware outline, instruction execution times and an approximate package count. This preliminary system outline will enable the calculation of execution times for basic instruction types based on the cycle time, required memory accesses and the macro-instruction system definition. Table XIX shows the execution time of various types of instructions. Each instruction is separated into several execution states to show how the cycles are broken-up between different functions.

TABLE XIX

INSTRUCTION EXECUTION TIME SUMMARY

Instruction Type	Execution States	# Cycles	Execution Time
Memory reference	Instruction fetch	2	1.2 μs
	Data address fetch	2	
	Fetch data (RAM)	1	
	Function	1	
Load Immediate	Instruction fetch	2	1.2 μs
	1st half of constant	2	
	2nd half of constant	2	
Load (RAM)	Instruction fetch	2	1 μs
	Address fetch	2	
	Load	1	
Output (address in MEM)	Instruction fetch	2	1 μs
	Address fetch	2	
	Output to RAM	1	
Output (address in register) Note 1	Instruction fetch	2	1 μs
	Decrement PC	1	
	Load MA	1	
	Output to RAM	1	
Register to register Note 1	Instruction fetch	2	0.8 μs
	Decrement PC	1	
	Function	1	

Note 1: All single word instructions lose one cycle time. The PC is actually decremented during the output to RAM cycle.

Single word instructions lose one cycle time due to the pipeline register architecture. It takes three cycle times before control is transferred to the macro-instruction ROM address field. Thus the micro-instruction will increment PC and load MA as if a second macro-instruction is going to be fetched. If a second macro-instruction is not fetched, then the PC must be decremented before the next instruction fetch. This can be done during any microcode output function when the ALU is not being used. Also it is possible to actually gain cycles by using the ALU during those memory output cycles. If the last function of a particular instruction is a memory write, then the PC can be incremented and loaded into the MA register for the next instruction at the same time.

An approximate package count is calculated in Table XX. This calculation does not take into account two important problems. First ROM memory cannot be used in the prototype system. RAM can be substituted so that the system still sees ROM memory, but external hardware will be required to load the RAM with the program. Even after initial debug, PROMs will probably be used rather than ROMs. The second problem is that more control hardware and a control panel will be necessary for debug. The ability to stop on a given address and to single cycle through program execution is necessary. There are two levels of firmware which means two levels of control are required. Even a second processor will be required to run assemblies, load memory, simulate the input interface, etc. The system designed to this point gives only an idea as to how the working or debugged system hardware will look. The slice interconnections were never drawn because of an effort not to duplicate material presented in the application literature. Furthermore,

the system presented was only segments of the design without a complete circuit being presented. Also the power supply requirements have not been estimated. Thus, the hardware outline presented is not complete in every detail.

TABLE XX

PACKAGE COUNT

Function	IC	Number Used
Multiplier	Am25S05	0 or 131
Latches - IR	Am2918	2
MIR	Am2918	9
Microprogram sequencer	Am2909	3
MEM - μROM	256-Word by 4-bit	8
ROM	1024-Word by 8-bit	8 (initial 4K
RAM	256-Word by 4-bit	8 of ROM)
Slice-	Am2901	8
	Am2902	3
Bus - MD	Am2905	8
MA	Am2905	4
Control	Misc.	12
Shift	74S253	2
	Total	75 or 206

At this point, the complete fire control data processing system, especially the data smoothing algorithm, should be determined. After

software simulation of the system one must begin defining required macro-instructions for adopting the software to the microprocessor system and determine the microfunctions required to implement the macro-instructions. The system presented has a general but powerful microfunction base believed adequate to implement the required macrocodes. Only refinement and adjustment is now required as firmware development brings up more specialized required functions. A great deal of parallel development is required now for firmware and hardware.

Chapter VI involves project development charts showing how to proceed with the project and final remarks. Approximate execution times for the more complex functions such as SIN, COS, etc. can now be calculated. Using the algorithm discussed in Chapter II the SIN function, for example, may be implemented as a macrolevel subroutine requiring approximately eight multiplies, one subroutine call, one subroutine return, six load immediates, three loads, four exchanges (load contents of one register into a second), three adds and one subtract. The execution time is approximately 30 μs for the hardware multiply system and 80 μs for the semi-hardware multiply. These times could be improved if the functions were executed on the microlevel but this would require more hardware and microcode bits for expanded micro-control for entering the required constants. These are the constants used in the execution of the rational polynomial approximations. The macrolevel execution simply uses the load immediate instruction to load 32-bit constants.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

6.1 Summary and Conclusions

The feasibility study is now completed by the selection and application of the Am2900 microprocessor. The study was begun by outlining some basic system requirements for the selection process. The spectrum of microprocessors was studied and shortened to those which met the processing requirements. At this point specific manufacturers' microprocessors were examined for final selection. After the selection was made, a preliminary system design was outlined for purposes of instruction execution time calculations and hardware complexity determination. This simplified system design shows that it is feasible to implement the controller system using microprocessors given that the resources are available.

This study shows the hardware implementation to be feasible but the firmware requirements are exceptional. The very complex completely discrete hardware implementation has been simplified by the use of a microprocessor but the complexity has been shifted to the firmware development. The system approach for this study was to implement all functions if possible in firmware. This led to the required use of a slice microprocessor. A second approach is to implement to a greater degree more functions in hardware thus reducing the microprocessor requirements. For example, the extended Kalman filter could be implemented in discrete

hardware so that the firmware requirements would be reduced to decision making and calculation of new firing parameters. This would allow the use of the 16-bit, fixed-instruction set type of microprocessors reducing the firmware development significantly. Not only are the firmware functions reduced, but the microprogramming requirement of a slice microprocessor is eliminated. It would also reduce the effectiveness of the microprocessor in providing the full advantages of the microprocessor's use, such as the flexibility of firmware and the reduced cost of the hardware implementation.

The present system outlined using the Am2900 microprocessor has the advantage of maximum firmware use for flexibility and hardware reduction. The major disadvantage is the slice microprocessor requirement of microprogramming a macro-instruction set. This is also an advantage in some respects, since it allows the implementation of special purpose instructions but at the cost of the increased firmware development.

6.2 Project Continuance

If the system approach taken by this study is accepted, then the next step is to completely define the firmware functions. The extended Kalman filter should be implemented into the existing simulation programs. It is imperative that a working simulation program is available so that firmware development will be mostly transforming the simulation program into the microprocessor's instruction set.

The present state of development is shown in Chapter V. The hardware system has been outlined to the point to at least define the basic microcode functions. At this point one may begin a microcode assembler, finish the hardware design and begin definition of the required macro-

code functions. The design effort will require a parallel structure as shown by the program development chart in Figure 37. The parallel development is required due to the feedback of each section into the other sections. As macrocode is developed, new microcode functions will be required which then lead to new hardware.

The development chart of Figure 37 shows the complexity of the program and the manpower requirements. There are three distinct development paths: hardware, microlevel and macrolevel development. The majority of the effort will be concentrated in firmware development. This means software oriented and hardware oriented people are necessary. The project can be segmented into various development phases with one person responsible for each phase. This can only be done to a small degree, since communication problems arise and many problems surface during final system debug when the pieces must be put together.

The total manhours required are impossible to determine but this will be a significant project requiring good organization. The manpower requirements were never estimated but the study was presented in such a manner as to give the reader a basic understanding of what will be required to complete the project. Thus, the project may begin where this study leaves off. Continuation has been outlined and present status given. Different approaches to the system have also been presented. The system outlined in this study must now be evaluated. If accepted, the project continuance will be a process of hardware modification and refinement and firmware development.

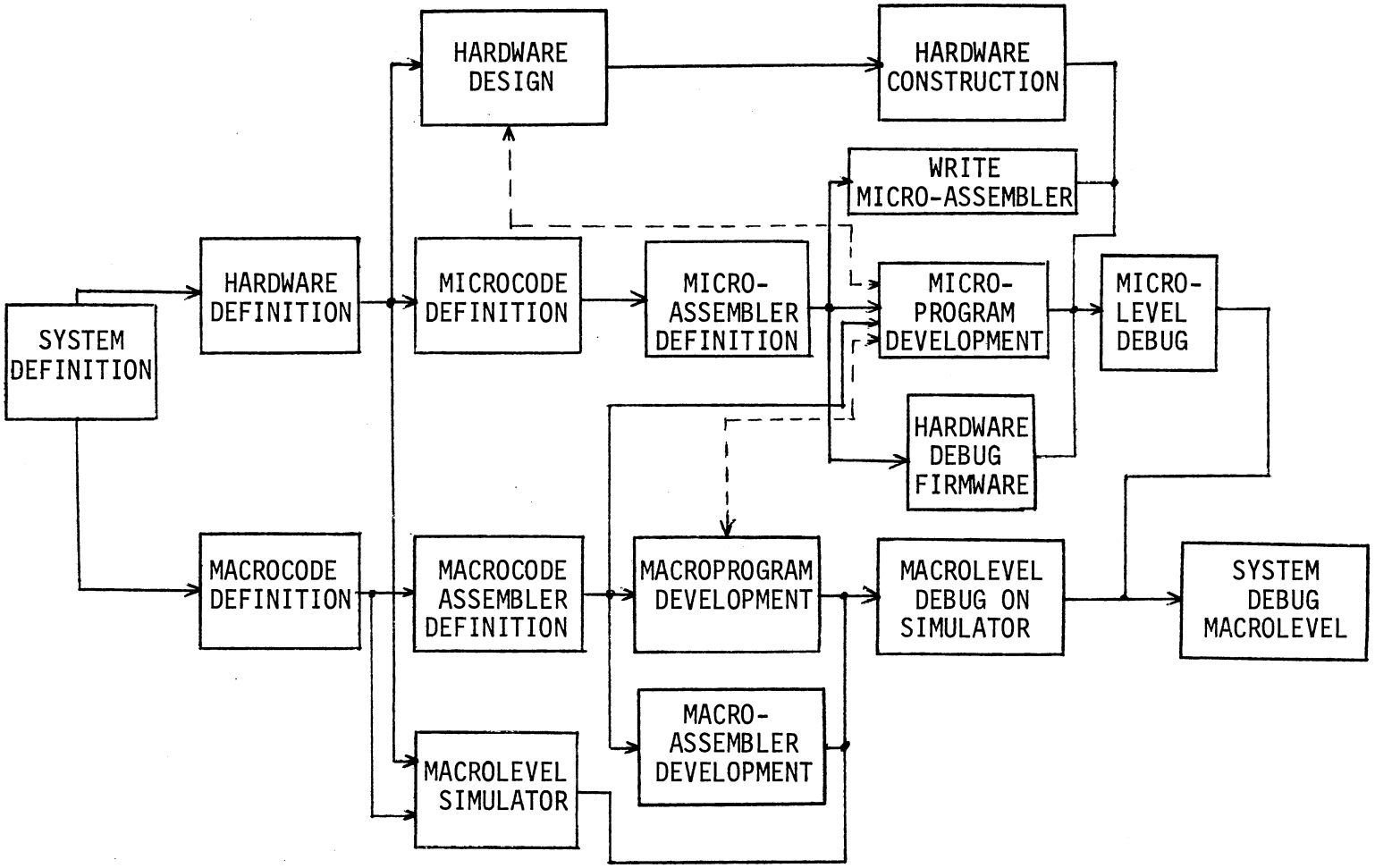


Figure 37. Program Development Chart

BIBLIOGRAPHY

Articles

(1) Cronenwett, W. T. and J. H. Christensen. "Digital Applications and Microprocessors." IEEE Fall Technical Seminar, Oklahoma City, December, 1974.

(2) Cushman, Robert H. "CMOS-Yesterday's Orphan Has Greatly Prospered." EDN, Vol. No. 20 (August, 1975), pp. 22-31.

(3) Davidow, William H. "General-Purpose Microcontrollers Part I: Economic Considerations." Computer Design, Vol. No. 11 (July, 1972), pp. 75-79.

(4) Davidow, William H. "General-Purpose Microcontrollers Part II: Design and Applications." Computer Design, Vol. No. 11 (August, 1972), pp. 69-75.

(5) Faggin, F. and M. E. Hoff. "Standard Parts and Custom Design Merge in Four-Chip Processor Kit." Electronics, Vol. No. 45 (April, 1972), pp. 112-116.

(6) Fox, Gregory. "Evaluation of Microprocessor Performance for Military Systems Application." IEEE Transactions on Circuits and Systems, Vol. No. CAS-22 (April, 1975), pp. 6-8.

(7) Frankenberg, Robert J. "Designer's Guide to: Semiconductor Memories - Part II." EDN, Vol. No. 20 (August, 1975), pp. 58-65.

(8) Ghest, R. C. "A 2's Complement Digital Multiplier for the Am2505." Sunnyvale, California: Advanced Micro Devices, 1971.

(9) Holt, Raymond and Manuel R. Lemas. "Current Microcomputer Architecture." Computer Design, Vol. No. 13 (February, 1974), pp. 65-73.

(10) Horton, Richard L., Jesse Englade and Gerald McGee. "I^2L Takes Bipolar Integration a Significant Step Forward." Electronics, Vol. No. 48 (January, 1975), pp. 83-90.

(11) Jaeger, Robert. "Microprogramming: A General Design Tool." Computer Design, Vol. No. 13 (August, 1974), pp. 150-157.

(12) Lewin, Morton H. "Integrated Microprocessors." IEEE Transactions

on Circuits and Systems, Vol. No. CAS-22 (July, 1975), pp. 577-585.

(13) "New Products - Four-Bit Bipolar/LSI Processor Slice Cuts Microcycle Time to 100 ns." Electronic Design, Vol. 15 (July, 1975), pp. 77-78.

(14) Perlowski, A. A. "Potential User's Microprocessor Checklist." Plymouth, Minnesota: Microprocessors Evaluation Laboratory Solid State Electronics Center, June, 1974.

(15) Riley, Wallace B. "Special Report: Semiconductor Memories are Taking Over Data-Storage Applications." Electronics, Vol. No. 46 (August, 1973), p. 75.

(16) Schmid, Hermann. "Monolithic Processors." Computer Design, Vol. No. 13 (October, 1974), pp. 87-95.

(17) Schmid, Hermann. "Speed Microcomputer Multiplication With A CPU Complementary Circuit or Peripheral Multiplier. Here are Typical Examples That Use Available ICs." Electronic Design, Vol. 9 (April, 1975), pp. 44-51.

(18) Schultz, Gaymond, Raymond Holt and Harold McFarland, Jr. "A Guide to Using LSI Microprocessors." Computer, Vol. No. 6 (June, 1973), pp. 13-19.

(19) Tarui, Tadaaki, Keiji Namimoto and Yukiharu Takahashi. "Twelve-Bit Microprocessor Nears Minicomputer's Performance Level." Electronics, Vol. No. 47 (March, 1974), pp. 111-116.

(20) Torrerro, Edward A. "Focus on Microprocessors." Electronic Design, Vol. 18 (September, 1974), pp. 52-68.

(21) Weisbecker, Joe. "A Simplified Microcomputer Architecture." Computer, Vol. No. 7 (March, 1974), pp. 41-48.

(22) Weissberger, Alan J. "Microprocessors Expand Industry Applications of Data Acquisition." Electronics, Vol. No. 47 (September, 1974), p.107.

Specification Literature

(23) Advanced Micro Devices Incorporated. A Preview of the Am2900 Family. Marketing Bulletin. Sunnyvale, California, 1975.

(24) Advanced Micro Devices Incorporated. Am2900 Bipolar Microprocessor Family. Data Catalog. Sunnyvale, California, 1975.

(25) Fairchild Corporation. 9400 Series Macrologic Composite Data Sheet. Preliminary Technical Information. Mountain View, California, 1975.

(26) Intel Corporation. Disk Controller Design Uses New Bipolar Microcomputer LSI Components. Applications Note. Santa Clara, California, 1975.

(27) Intel Corporation. Intel Microcomputers. Data Catalog. Santa Clara, California, 1974.

(28) Intel Corporation. Preliminary Specification, Intel Bipolar Microcomputer Set Cross Micro-assembler, XMAS. Santa Clara, California, 1975.

(29) Intel Corporation. Using the Intel Bipolar Microcomputer Set. Santa Clara, California, 1975.

(30) Intel Corporation. 3000 Bipolar System Development Kit. Data Sheet. Santa Clara, California, 1974.

(31) Intel Corporation. Intel 3001 Microprogram Control Unit. Data Sheet. Santa Clara, California, 1975.

(32) Intel Corporation. Intel 3002 Central Processing Element. Data Sheet. Santa Clara, California, 1975.

(33) Motorola Semiconductor Products Incorporated. A New Concept in Processor ICs. Phoenix, Arizona, 1975.

(34) Motorola Semiconductor Products Incorporated. General Information M10800 Bipolar Microprocessor Program. Tentative Data Sheet. Phoenix, Arizona, 1975.

(35) National Semiconductor Corporation. IMP-16 Programming and Assembler Manual. Santa Clara, California, 1974.

(36) National Semiconductor Corporation. IMP-16C Product Description. Santa Clara, California, 1973.

(37) Norman, Robert. Independent Study Project. Stillwater, Oklahoma: Oklahoma State University, 1975.

(38) School of Engineering, Final Report, Real Time Registry and Control Feasibility Study. Stillwater, Oklahoma: Oklahoma State University, 1975.

(39) Texas Instruments Corporation. "Microprocessor Introduction Seminar." Oklahoma City, Oklahoma, July, 1975.

(40) Texas Instruments Corporation. SBP0400 Four-Bit Parallel Binary Processor Element. Tentative Data Sheet. Dallas, Texas, 1975.

(41) Texas Instruments Corporation. TMS1000 Series MOS/LSI One-Chip Microcomputers. Data Sheet. Dallas, Texas, 1975.

APPENDIX A

IMP-16 INSTRUCTION SET

TABLE XXI

MEMORY REFERENCE INSTRUCTIONS

Instruction	Execution Cycles	Memory cycles Read	Write	Execution Time (μs)
Load	5	2		7.7
Load indirect	5	3		8.05
Store	6	1	1	9.1
Store indirect	8	2	1	12.25
Add	5	2		7.7
Subtract	5	2		7.7
Jump	3	1		4.55
Jump indirect	5	2		7.7
Jump to subroutine	4	1		5.95
Jump to subroutine indirect	6	2		9.1
Increment and skip if zero	7,8 if skip	2	1	12.25
Decrement and skip if zero	8,9 if skip	2	1	13.65
Skip if AND is zero	6,7 if skip	2		10.5
Skip if greater Like signs:	8,9 if skip	2		13.3
Unlike signs:	9,10 if skip	2		14.7
Skip if not equal	6	2		9.1
AND	5	2		7.7
OR	5	2		7.7

TABLE XXII

REGISTER REFERENCE INSTRUCTIONS

Instruction	Execution Cycles	Memory Cycles Read	Memory Cycles Write	Execution Time (μs)
Push on to stack reg	3	1		4.55
Pull from stack	3	1		4.55
Add immediate, skip if zero	4,5 if skip	1		7.35
Load immediate	3	1		4.55
Complement and add immediate	3	1		4.55
Register copy	6	1		8.75
Exchange register and top of stack	5	1		7.35
Exchange registers	8	1		11.55
Register AND	6	1		8.75
Register exclusive OR	6	1		8.75
Register add	3	1		4.55
Shift left	4 + 3K	1		
Shift right	4 + 3K	1		
Rotate left	4 + 3K	1		
Rotate right	4 + 3K	1		

Note: "K" equals the number of bits shifted

TABLE XXIII

INPUT/OUTPUT, FLAG, AND HALT INSTRUCTIONS

Instruction	Execution Cycles	Memory Cycles Read	Memory Cycles Write	Execution Time (μs)
Set flag	4	1		5.95
Pulse flag	4	1		5.95
Push flags on stack	4	1		5.95
Pull flags from stack	5	1		7.35
Register in	7	1		10.15
Register out	7	1		10.15
Halt	-	1		1.75

TABLE XXIV

TRANSFER OF CONTROL INSTRUCTIONS

Instruction	Execution Cycles	Memory Cycles Read	Memory Cycles Write	Execution Time (μs)
Branch-on condition	4,5 if branch	1		7.35
Return from subroutine	4	1		5.95
Return from interrupt	5	1		7.35
Jump to subroutine implied	4	1		5.95

TABLE XXV

EXTENDED INSTRUCTION SET

Instruction	Execution Cycles	Memory Cycles Read	Memory Cycles Write	Execution Time (μs)
Multiply	106 to 122	3		171.85
Divide	125 to 159	3		223.65
Double precision add	12	4		18.2
Double precision subtract	12	4		18.2
Load byte	20 (left)	4		29.4
	12 (right)	4		18.2
Store byte	24 (left)	4	1	35.35
	17 (right)	4	1	25.55
Set status flag	17 to 36	1		50.75
Clear status flag	17 to 36	1		50.75
Skip if status flag true	19 to 39	1		54.95
Set bit	15 to 34	1		47.95
Clear bit	15 to 34	1		47.95
Complement bit	15 to 34	1		47.95
Skip if bit true	19 to 39	1		54.95
Interrupt scan	9 to 80	1		112.35
Jump indirect to level zero interrupt	7	2		10.5
Jump through pointer	7	3		10.85
Jump to subroutine through pointer	8	3		12.25

APPENDIX B

MICROPROGRAMED MICROPROCESSOR INSTRUCTION SETS

TABLE XXVI

INTEL 3001

Mnemonic	Function
JCC	Jump in current column
JZR	Jump to zero row
JCR	Jump in current row
JCE	Jump in current column/row group and enable PR-latch outputs
JFL	Jump/test F-latch
JCF	Jump/test C-flag
JZF	Jump/test Z-flag
JPR	Jump/test PR-latch
JLL	Jump/test leftmost PR-latch bits
JRL	Jump/test rightmost PR-latch bits
JPX	Jump/test PX-bus and load PR-latch
SCZ	Set C-flag and Z-flag to F1
STZ	Set Z-flag to F1
STC	Set C-flag to F1
HCZ	Hold C-flag and Z-flag
FF0	Force F0 to 0
FFC	Force F0 to C
FFZ	Force F0 to Z
FF1	Force F0 to 1

TABLE XXVII

INTEL 3002

Mnemonic	Microfunction
ILR	$R_n + C1 \rightarrow R_n$, AC
ACM	$M + C1 \rightarrow AT$
SRA	$AT_L \rightarrow RO \quad AT_H \rightarrow AT_L \quad L1 \rightarrow AT_H$
LMI	$R_n \rightarrow MAR \quad R_n + C1 \rightarrow R_n$
LMM	$M \rightarrow MAR \quad M + C1 \rightarrow AT$
CIA	$\overline{AT} + C1 \rightarrow AT$
CSR	$C1 - 1 \rightarrow R_n$
CSA	$C1 - 1 \rightarrow AT$
INR	$R_n + C1 \rightarrow R_n$
INA	$AT + C1 \rightarrow AT$
CLR	$C1 \rightarrow CO \quad 0 \rightarrow R_n$
CLA	$C1 \rightarrow CO \quad 0 \rightarrow AT$
NOP	$C1 \rightarrow CO \quad R_n \rightarrow R_n$
LMF	$C1 \rightarrow CO \quad M \rightarrow AT$
CMR	$C1 \rightarrow CO \quad \overline{R_n} \rightarrow R_n$
LCM	$C1 \rightarrow CO \quad \overline{M} \rightarrow AT$
CMA	$C1 \rightarrow CO \quad \overline{AT} \rightarrow AT$
ALR	$AC + R_n + C1 \rightarrow R_n$, AC
AMA	$M + AC + C1 \rightarrow AT$
DSM	$11 \rightarrow MAR \quad R_n - 1 + C1 \rightarrow R_n$
LDM	$11 \rightarrow MAR \quad M - 1 + C1 \rightarrow AT$
DCA	$AT - 1 + C1 \rightarrow AT$
SDR	$AC - 1 + C1 \rightarrow R_n$
SDA	$AC - 1 + C1 \rightarrow AT$
LDI	$I - 1 + C1 \rightarrow AT$
ADR	$AC + R_n + C1 \rightarrow R_n$
AIA	$I + AT + C1 \rightarrow AT$
ANR	$C1 \vee (R_n \wedge AC) \rightarrow CO \quad R_n \wedge AC \rightarrow R_n$
ANM	$C1 \vee (M \wedge AC) \rightarrow CO \quad M \wedge AC \rightarrow AT$
ANI	$C1 \vee (AT \wedge I) \rightarrow CO \quad AT \wedge I \rightarrow AT$

TABLE XXVII (Continued)

Mnemonic	Microfunction
TZR	$C1 \; V \; R_n \rightarrow CO \quad R_n \rightarrow R_n$
LTM	$C1 \; V \; M \rightarrow CO \quad M \rightarrow AT$
TZA	$C1 \; V \; AT \rightarrow CO \quad AT \rightarrow AT$
ORR	$C1 \; V \; AC \rightarrow CO \quad R_n \; V \; AC \rightarrow R_n$
ORM	$C1 \; V \; AC \rightarrow CO \quad M \; V \; AC \rightarrow AT$
ORI	$C1 \; V \; 1 \rightarrow CO \quad 1 \; V \; AT \rightarrow AT$
XNR	$C1 \; V \; (R_n \; AC) \rightarrow CO \quad R_n \; \overline{\oplus} \; AC \rightarrow R_n$
XNM	$C1 \; V \; (M \; AC) \rightarrow CO \quad M \; \overline{\oplus} \; AC \rightarrow AT$
XNI	$C1 \; V \; (AT \; 1) \rightarrow CO \quad 1 \; \overline{\oplus} \; AT \rightarrow AT$

TABLE XXVIII

EXPLANATION OF SYMBOLS

Symbol	Meaning
Cl	Carry input
CO	Carry output
I, K, M	Data on I, K, M buses, respectively
R_n	Contents of register n including T and AC
AC	Contents of accumulator
AT	Contents of AC or T, as specified
MAR	Contents of memory address register
+	Two's complement addition
-	Two's complement subtraction
$\wedge$	Logical AND
V	Logical OR
$\overline{\theta}$	Exclusive NOR
$\rightarrow$	Deposit into
LI, RO	Data on left input and right output
L, H	As subscripts, designate low and high order bit, respectively

TABLE XXIX

INSTRUCTION SET FOR THE 9404

Instruction Inputs I_4 I_3 I_2 I_1 I_0	$\overline{LO}$	$\overline{O}_3$ $\overline{O}_2$ $\overline{O}_1$ $\overline{O}_0$	$\overline{RO}$	Function
L L L L L		L L L L		Byte Mask
L L L L H		H H H H		Byte Mask
L L L H L		L L L H		Minus "2" in 2's Complement
L L L H H		L L L L		Minus "1" in 2's Complement
L L H L L		$\overline{D}_3$ $\overline{D}_2$ $\overline{D}_1$ $\overline{D}_0$		Byte mask D-bus
L L H L H		H H H H		Byte mask D-bus
L L H H L		$\overline{D}_3$ $\overline{D}_2$ $\overline{D}_1$ $\overline{D}_0$		Byte mask D-bus
L L H H H		L L L L		Byte mask D-bus
L H L L L		L H H H		Negative byte sign mask
L H L L H		H H H H		Positive byte sign mask
L H L H L		$\overline{K}_3$ $\overline{K}_2$ $\overline{K}_1$ $\overline{K}_0$		Byte mask K-bus
L H L H H		L L L L		Byte mask K-bus
L H H L L		$\overline{D}_3$ $\overline{D}_2$ $\overline{D}_1$ $\overline{D}_0$		Load byte
L H H L H		$\overline{K}_3$ $\overline{K}_2$ $\overline{K}_1$ $\overline{K}_0$		Load byte
L H H H L		H H H L		Plus "1"
L H H H H		H H H H		Zero
H L L L L	$\overline{RI}$	$\overline{RI}$ $\overline{RI}$ $\overline{RI}$ $\overline{RI}$		K-bus sign extend
H L L L H	$\overline{K}_3$	$\overline{K}_3$ $\overline{K}_2$ $\overline{K}_1$ $\overline{K}_0$		K-bus extend
H L L H L	$\overline{RI}$	$\overline{RI}$ $\overline{RI}$ $\overline{RI}$ $\overline{RI}$		D-bus sign extend
H L L H H	$\overline{D}_3$	$\overline{D}_3$ $\overline{D}_2$ $\overline{D}_1$ $\overline{D}_0$		D-bus sign extend
H L H L L	$\overline{D}_3$	$\overline{D}_2$ $\overline{D}_1$ $\overline{D}_0$ $\overline{RI}$		D-bus shift left
H L H L H	$\overline{K}_3$	$\overline{K}_2$ $\overline{K}_1$ $\overline{K}_0$ $\overline{RI}$		K-bus shift left
H L H H L		$\overline{LI}$ $\overline{D}_3$ $\overline{D}_2$ $\overline{D}_1$	$\overline{D}_0$	D-bus shift right
H L H H H		$\overline{D}_3$ $\overline{D}_3$ $\overline{D}_2$ $\overline{D}_1$	$\overline{D}_0$	D-bus shift right arithmetic
H H L L L		$\overline{LI}$ $\overline{K}_3$ $\overline{K}_2$ $\overline{K}_1$	$\overline{K}_0$	K-bus shift right
H H L L H		$\overline{K}_3$ $\overline{K}_3$ $\overline{K}_2$ $\overline{K}_1$	$\overline{K}_0$	K-bus shift right arithmetic

TABLE XXIX (Continued)

Instruction Inputs I_4	I_3	I_2	I_1	I_0	$\overline{LO}$	$\overline{O}_3$	$\overline{O}_2$	$\overline{O}_1$	$\overline{O}_0$	$\overline{RO}$	Function
H	H	L	H	L		$\overline{K}_3$	$\overline{K}_2$	$\overline{K}_1$	$\overline{K}_0$		Byte mask K-bus
H	H	L	H	H		H	H	H	H		Byte mask K-bus
H	H	H	L	L		D_3	D_2	D_1	D_0		Complement D-bus
H	H	H	L	H		K_3	K_2	K_1	K_0		Complement K-bus
H	H	H	H	L							Unassigned
H	H	H	H	H							Unassigned

H = High level
L = Low level

TABLE XXX

I-FIELD ASSIGNMENT

I_2	I_1	I_0	Internal Operation
L	L	L	Rx plus D-bus plus 1 → Rx
L	L	H	Rx plus D-bus → Rx
L	H	L	Rx D-bus → Rx (logic AND)
L	H	H	D-bus → Rx
H	L	L	Rx → D-bus
H	L	H	Rx + D-bus → Rx (logic OR)
H	H	L	Rx ⊕ D-bus → Rx (logic X-OR)
H	H	H	$\overline{\text{D-bus}}$ → Rx

NOTES:

1. Rx is the RAM location addressed by A_0 - A_2.
2. The result of any operation is always loaded into the output register.

H = Logic high level
L = Logic low level

TABLE XXXI

INSTRUCTION SET FOR THE 9406

I_1	I_0	Instruction	Internal Operation	X-bus	O-bus (with $\overline{EO}_0$ low)
L	L	Return (pop)	Decrement stack pointer	Disabled	Depending on the relative timing of $\overline{EX}$ and CP, the outputs will reflect the current program count or the new value while CP is low. When CP goes high again, the output will reflect the new value.
L	H	Branch (load PC)	Load D-bus into current program counter location	Disabled	Current program counter until CP goes high again, then updated with newly entered PC value.
H	L	Call (push)	Increment stack pointer and load D-bus into new program counter location	Disabled	Depending on the relative timing of $\overline{EX}$ and CP, the outputs will reflect the current program count or the previous contents of the incremented SP location. When CP goes high again, the outputs will reflect the newly entered PC value.
H	H	Fetch (Increment PC)	Increment current program counter if $\overline{CI}$ is low	Current program counter while both CP and EX are low, disabled while CP or $\overline{EX}$ is high	Current program counter until CP goes high again, then updated with incremented PC value.

H = High level; L = Low level

TABLE XXXII

INSTRUCTION SET FOR THE 9407

Instruction I_3 I_2 I_1 I_0	Combinatorial Function Available on the X-bus	Sequential Function Occurring on the Next Rising CP Edge
L L L L L L L H	R_0 R_0 plus D plus CI	R_0 plus D plus CI → R_0 and O-register
L L H L L L H H	R_0 R_0 plus D plus CI	R_0 plus D plus CI → R_1 and O-register
L H L L L H L H	R_0 R_0 plus D plus CI	R_0 plus D plus CI → R_2 and O-register
L H H L L H H H	R_1 R_1 plus D plus CI	R_1 plus D plus CI → R_1 and O-register
H L L L H L L H	R_2 D plus CI	D plus CI → R_2 and O-register
H L H L H L H H	R_0 D plus CI	D plus CI → R_0 and O-register
H H L L H H L H	R_2 R_2 plus D plus CI	R_2 plus D plus CI → R_2 and O-register
H H H L H H H H	R_1 D plus CI	D plus CI → R_1 and O-register

TABLE XXXIII

2901 MICROCODE

Microcode I_2 I_1 I_0 (Octal)	ALU Source Operands R	S
0	A	Q
1	A	B
2	0	Q
3	0	B
4	0	A
5	D	A
6	D	Q
7	D	0

Microcode I_5 I_4 I_3 (Octal)	ALU Function	Symbol
0	R plus S	R+S
1	S minus R	S-R
2	R minus S	R-S
3	R or SR V S	
4	$\overline{R}$ and S	$\overline{R} \wedge S$
5	$\overline{R}$ and S	$\overline{R} \wedge S$
6	R EX - OR S	$R \not\vee S$
7	R EX - NOR s	$\overline{R \not\vee S}$

Microcode I_8 I_7 I_6 (Octal)	RAM Function Shift	Load	Q-Reg Function Shift	Load	Y Output
0	-	-	None	ALU	F
1	-	-	-	-	F
2	None	ALU	-	-	A
3	None	ALU	-	-	F
4	Left	ALU	Left	Q-reg	F
5	Left	ALU	-	-	F
6	Right	ALU	Right	Q-reg	F
7	Right	ALU	-	-	F

TABLE XXXIV

MC10800 ALU FUNCTION SET

Logic	Binary	BCD
F = logic 0	F = A plus 0	F = A plus 0
F = A	F = A minus 0	F = A minus 0
F = $\overline{0}$	F = 0 minus A	F = 0 minus A
F = $\overline{A}$	F = A	F = A
F = $\overline{0}$	F = $\overline{0}$	F = 0
F = A + $\overline{0}$	F = $\overline{A}$	F = 9's comp. A
F = $\overline{A}$ + $\overline{0}$	F = $\overline{0}$	F = 9's comp. 0
F = $\overline{A}$ + 0	F = A minus 1	F = A plus 2
F = A · $\overline{0}$	F = 0 minus 1	F = 0 plus 2
F = $\overline{A}$ · $\overline{0}$	F = A minus 2	F = A plus A
F = $\overline{A}$ · 0	F = 0 minus 2	F = 0 plus 0
F = A ⊕ $\overline{0}$	F = A plus 2	F = ACC plus A
F = $\overline{A}$ ⊕ $\overline{0}$	F = 0 plus 2	F = ACC plus 0
F = $\overline{A \cdot 0}$	F = A plus A	F = ACC minus A
F = $\overline{A + 0}$	F = 0 plus 0	F = ACC minus 0
F = logic $\overline{1}$	F = ACC plus A	F = ACC plus A · 0
F = ACC · $\overline{A}$	F = ACC minus A	F = ACC minus A · 0
F = ACC · $\overline{0}$	F = ACC plus 0	F = ACC plus A + 0
F = $\overline{ACC}$ + A	F = ACC minus 0	F = ACC minus A + 0
F = $\overline{ACC}$ + 0	F = ACC plus A · 0	
F = ACC ⊕ A	F = ACC minus A · 0	
F = ACC ⊕ $\overline{A}$	F = ACC plus A + 0	
F = ACC ⊕ 0	F = ACC minus A + 0	
F = ACC ⊕ $\overline{0}$		
F = ACC ⊕ $\overline{A \cdot 0}$		
F = ACC ⊕ $\overline{A \cdot 0}$		
F = ACC ⊕ $\overline{A + 0}$		
F = ACC ⊕ $\overline{A + 0}$		

VITA

David Ernest West

Candidate for the Degree of

Master of Science

Thesis: MICROPROCESSOR FEASIBILITY STUDY AND PRELIMINARY DESIGN FOR AN ARTILLERY FIRE CONTROL APPLICATION

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in San Antonio, Texas, January 13, 1952, the son of Mr. and Mrs. Leonard L. West.

Education: Graduated from Northwest Classen High School, Oklahoma City, Oklahoma, in May, 1970; received the Bachelor of Science in Electrical Engineering degree from the University of Oklahoma, Norman, Oklahoma, in May, 1974; completed requirements for the Master of Science degree at Oklahoma State University, Stillwater, Oklahoma, in May, 1976.

Professional Experience: Member of Advanced Engineering program of Honeywell Information Systems, Oklahoma City, Oklahoma, from June 1974 to July 1975; Digital Systems Engineer, Magnetic Peripherals Incorporated, Floppy Disk Controller Development Section, Oklahoma City, Oklahoma, from July 1975.