# AMI 7300 MICROPROCESSOR
# GENERAL AND FUNCTIONAL DESCRIPTION

American Micro-systems, Inc.
Microprocessor Group
3800 Homestead Road
Santa Clara, CA 95051

April 1973

## TABLE OF CONTENTS

Page

GENERAL FEATURES 1

TYPICAL APPLICATIONS 1

I. INTRODUCTION 1

II. GENERAL DESCRIPTION 2

A. DATA EXCHANGE 5
B. READ MEMORY/PERIPHERAL 7
C. WRITE MEMORY/PERIPHERAL 9
D. DIRECT MEMORY ACCESS 13
E. SYSTEM TIMING 13
F. INTERRUPT PROCESSING 16

III. FUNCTIONAL DESCRIPTION 17

A. REGISTERS AND ALU CHIP (RALU) 21

1. Data Exchange Drivers and Receivers 21
2. Arithmetic and Logic Unit (ALU) 21
3. Decimal Arithmetic 22
4. Signed Decimal Arithmetic 26
5. A and B Source Registers 26
6. Status Registers 28
7. Temporary Status Bits 28
8. Final Status Bits 29
9. General Registers 29
10. Stack and Stack Pointers 30
11. ALU Result Register (RR) 31

B. MICROINSTRUCTION ROM CHIP (MIR) 31

1. Macroinstruction Registef (IR) 32
2. Instruction Mapping Array 32
a. Mapping Array Design 32
b. Decoder Design 34
3. Microprogram Read-Only Memory (MIROM) 36
4. ROM Address Register (RAR) and ROM Return Address Stack (RRAS) 36
5. Starting Address ROM (SAR) and Microinterrupt Control Logic 37
6. Step Counter/Step Counter Stack 37

# LIST OF ILLUSTRATIONS

| Figure | Title | Page |
|---|---|---|
| 1 | 7300 Microprocessor | 3 |
| 2 | Read Input/Output | 8 |
| 3A | Write I/O - Full Cycle | 10 |
| 3B | Write I/O | 11 |
| 4 | Direct Memory Access | 14 |
| 5 | Processor Timing | 15 |
| 6 | Registers and ALU Chip | 18 |
| 7 | Microinstruction ROM Chip | 19 |
| 8 | 2-4-2-1 Decimal Arithmetic On One Digit | 24 |
| 9 | 2-4-2-1 Decimal Arithmetic on Two Digits | 25 |
| 10 | Algebraic Addition and Subtraction of Two Numbers with Sign Magnitude Representation | 27 |
| 11 | Mapping Array Circuit | 33 |
| 12 | Microinstruction Decoder | 35 |

# AMI 7300 MICROPROCESSOR

## GENERAL FEATURES

- Microprogrammable
- LSI Based 2-Chip CPU, Memory and Peripheral Functions
- 8-Bit Byte Oriented Processor
- Powerful Microprogram Instruction Set
- Easy ROM/RAM Expansion to 65K Bytes
- 4.0$\mu$ Microinstruction Execution Time
- 49 Registers
- Fully Parallel I/O Bus
- Multiprocessor Bus Capability
- TTL Logic Interface
- Interrupt Features
- +5, -12V Power Supplies
- 2Ø Non-Overlapping Clocks

## TYPICAL APPLICATIONS

- Cash Registers
- Terminals
- Test Systems
- Measurement Systems
- Process Control Systems
- Accounting Machines
- Calculators
- I/O Processors

## I. INTRODUCTION

The AMI 7300 CPU is a fully parallel, highly flexible bus-organized system. It features an 8-bit Data Exchange Bus concept whereby the I/O devices, memory modules and central processors communicate with each other asynchronously.

The CPU contains 49 registers, 48 of which may, under microprogram control, be utilized as one or two First-In Last-Out (push-down) stacks or as one or two files of general registers. The CPU incorporates many architectural and logic design features to increase speeds of execution and allow for a wide range of applications.

## II. GENERAL DESCRIPTION

The 8-bit 7300 processor consists of two P-channel, Silicon Gate, Ion Implanted ($I^2$) MOS LSI devices:

Registers and ALU Chip (RALU)
Microinstruction ROM Chip (MIR)

The 7300 performs all of the functions commonly found in most minicomputers; the primary difference is the speed of execution, which is approximately one-sixth that of minicomputers. However, in many cases the 7300 may be faster. For instance, the stacks allow for fast subroutine calls executed at microinstruction speeds rather than memory cycle speeds. In the 7300, the system memory that contains macroinstructions and data may be core, semiconductor RAM, ROM or any combination operating at various rates. The basic machine architecture and partitioning are shown in Figure 1.

A typical byte-oriented application might use the complement of registers in the following manner:

| Number of 8-Bit Registers | Function | Number of Bits per Register |
|---|---|---|
| 1 | Accumulator 1 | 8 |
| 1 | Accumulator 2 | 8 |
| 2 | Index Register 1 | 16 |
| 2 | Index Register 2 | 16 |
| 2 | Index Register 3 | 16 |
| 2 | Index Register 4 | 16 |
| 2 | Program Counter | 16 |
| 1 | Temporary Register 1 | 8 |
| 1 | Temporary Register 2 | 8 |
| 1 | Temporary Register 3 | 8 |
| 1 | Temporary Register 4 | 8 |
| 1 | Status Register | 8 |
| 32 | Push-down Stack | 8 |

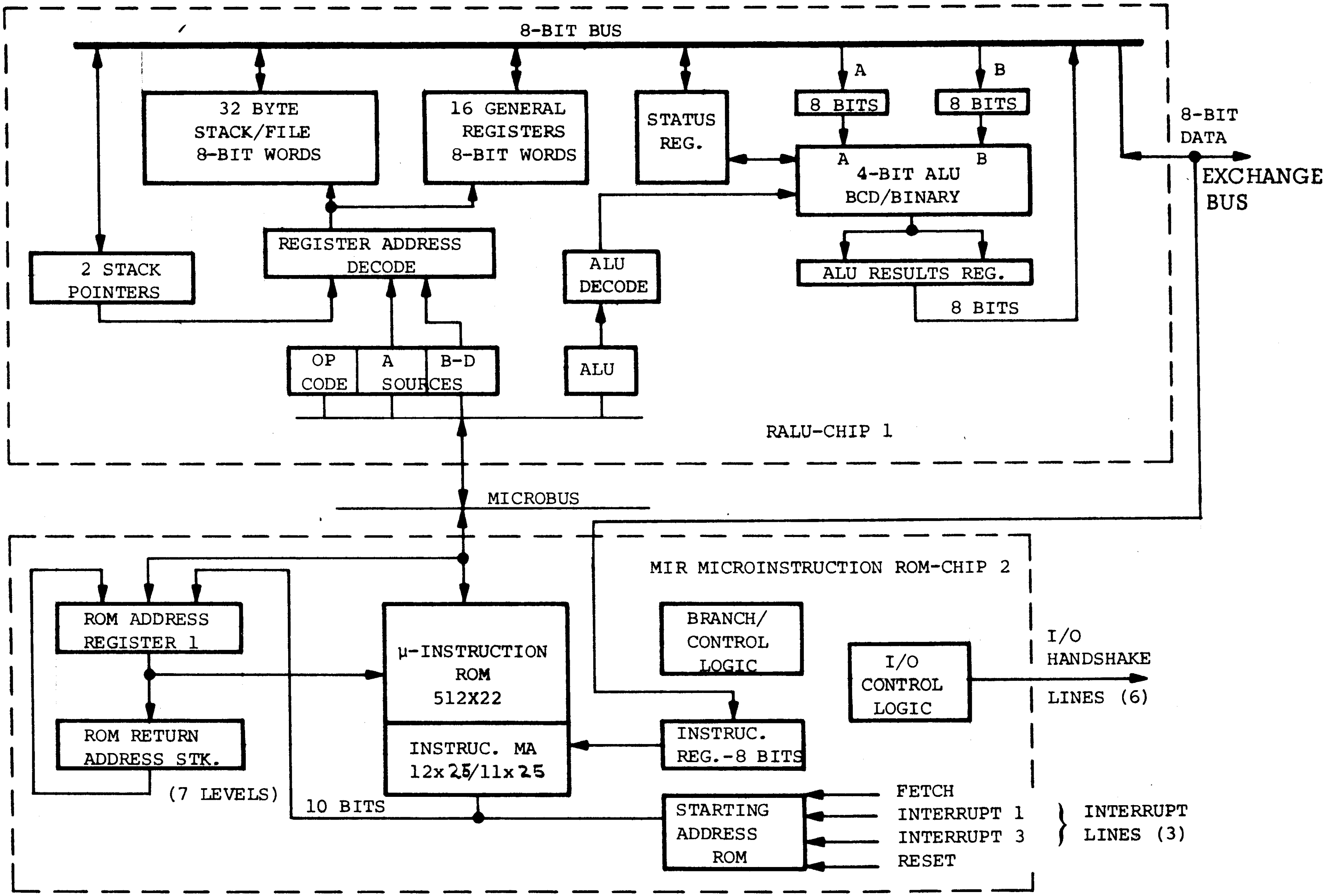


FIGURE 1 - 7300 MICROPROCESSOR

The 32 register push-down stack may be utilized as a set of general registers. A foreground-background system could be implemented by utilizing 32 registers as two groups of 16 general purpose registers.

The D Bus provides the sources and destinations for the two operands. The ALU chip operates autonomously because it includes all microinstruction decoding logic along with the appropriate timing signals. A 3-bit status register incorporates the following functions:

| Bit | Flag |
|---|---|
| 2 | Carry Flag |
| 0 | Zero Condition Flag |
| 1 | Minus Condition Flag |

The A and B Source Registers, which may also act as temporary storage registers, are connected directly to the ALU. The 4-bit ALU performs its 8-bit operations in five time slots; while the ALU is operating on one set of operands, those for the next operation are being accessed from the general file registers or stack. After the next operands have been accessed the last result is written into the register as specified by microprogram control.

The A and B Source Registers interface with the external world through the Data Exchange Multiplexers. When addressing external memory, the contents of the program counter are placed on the D Bus and clocked out to the Data Exchange Bus. When data is to be read from external memory the Data Exchange will drive the incoming data onto the D Bus. Arithmetic operations can be performed immediately if desired. Macroinstructions are clocked off the Data Exchange into the Instruction Register for decoding in the Instruction Mapping Array (IMA) on the MIR chip.

The Control section of the processor is implemented by microprogramming techniques. The microprogrammed control program is contained in the MIR which has a maximum capacity of 512 words by 22 bits. The instruction decoding function is performed by the IMA by relating the instruction to a starting point in the MIR. The large MIR and IMA make it possible to implement virtually any minicomputer instruction set. MIR chips may be paralleled to expand control store and IMA.

## A. DATA EXCHANGE

The 7300 Data Exchange (DE) is an 8-bit byte parallel I/O structure used to interconnect Memory, Processors, Control Panel and I/O Controllers. Most systems will require only one processor; however, the DE Bus and its handshake lines are designed so that more than one processor may be attached to the same bus simultaneously to construct simple multiprocessor systems. Peripheral controllers may communicate with the processor or directly with memory. It is also possible for peripheral processors to communicate with each other. The handshake lines are operated semi-asynchronously so that memory and peripherals of varying speeds and responses may be freely mixed on the same bus structure.

The form of communication is the same for every device on the DE. The processor uses the same set of signals to communicate with memory as with peripheral devices. Peripheral devices also use the same set of signals when communicating with the processor, memory or other peripheral devices. Peripheral device registers may be manipulated as flexibly as core memory by the processors.

The total Bus Addressing space is 65, 384 bytes. Although not mandatory, it is usually desirable to let the MSB of the 16-bit memory address differentiate between memory addresses and peripheral addresses.

At any point in time only one device can be in control of the DE. 16-bit addresses are communicated as two each 1-byte transfers. Data is sent or received on a byte transfer basis.

The Processor will drive the Data Exchange directly in small systems (such as with 3 chip calculators) and may be connected to Driver/Receiver pairs for larger systems. In larger systems the Data Exchange Drivers on the RALU chip will not have adequate Drive capability to Drive a number of TTL loads presented by memory controllers and peripheral controllers. The Data Exchange lines are defined below:

SF Store/Fetch represents the direction of the transfer.

1 = Processor stores data into memory or peripheral

2 = Processor fetches data from memory or peripheral

DR Drive/Receive is used to control the Driver/Receiver pairs. It specifies that the DE is busy and not available to other devices and times the beginning of an I/O cycle.

AK Acknowledge is used by memory and peripherals to specify that Data or Address has been received from the DE.

RUN Run is used to cause the machine to Run or stop running. Direct memory access devices will drive Run low to stop the processor and get access to the DE Bus. Run is also driven by the Run/Stop/Step switch on a control panel. Macro stepping is accomplished by pulsing the Run line.

FRUN Run Flip Flop tells when the machine has actually stopped or is running. This line drives the Run light on the control panel and specifies to direct memory access devices that the processor will not attempt to use the DE Bus.

RESET Reset clears the machine and causes the microprogram to begin execution at a specified address upon release of the Reset line.

K K specifies the beginning of a microcycle. It is useful for generating time slots in order to become synchronized with the processor and to clock address bytes from the Data Exchange during time slots 6 and 1. K represents time slots 1 and 2.

When interfacing with the 7300 it is necessary to read the Address bytes from the Data Exchange at precise times with respect to the Leading Edge of Drive/Receive. There are two ways to achieve this timing:

1. Construct the Leading Edge of DR using Ø1 clocks and shift this Leading Edge pulse through a short shift register to create the fourth and seventh time slots following the rise of DR.

2. Use the K pulse to create all the time slots of a microcycle on a continuous basis. Utilize time slots 6 and 1 ANDED together with DR to clock the two bytes of address from the DE.

Because of the simplified handshake scheme used with the 7300, the Data is left on the Data Exchange after a Reading process. This data will remain until DR is raised for the next I/O process. Data need not necessarily remain if it can be known how the machine will be microprogrammed, and thus when the Data is being read by the processor.

A timing diagram of each of the I/O processes is shown below to clarify the interrelationship between the handshake lines.

## B. READ MEMORY/PERIPHERAL (Figure 2)

The processor, control panel, or peripheral device reads data by executing the following sequence:

1. Get priority to use the DE Bus.

2. Raise DR and drive the most significant byte of address onto the DE Bus. (SF remains Low.)

3. All receiving devices perform the required timing with respect to DR and clock the first byte of address from the DE Bus.

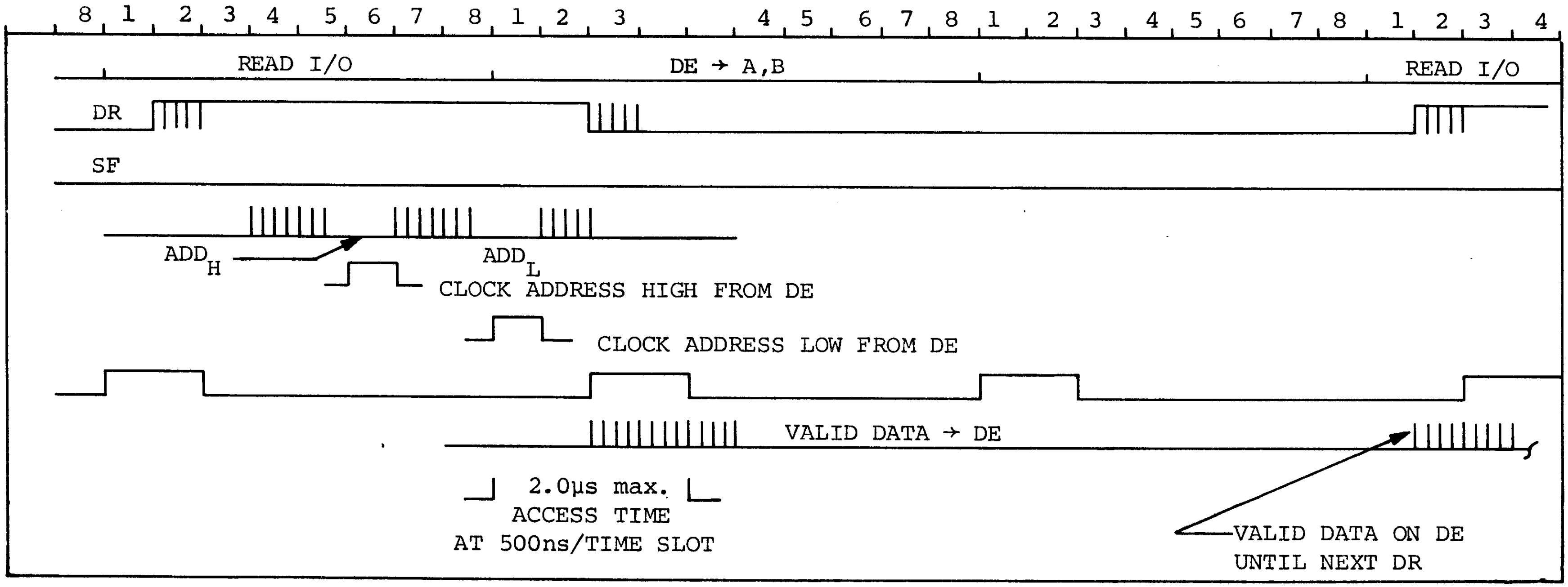


FIGURE 2 - READ INPUT/OUTPUT

4. The second byte of address is placed on the Data Exchange.

5. The receiving devices clock the second byte of address from the DE Bus. Once the second byte has been read, only the specified device being addressed begins the cycle to place data on the DE Bus.

6. The processor lowers DR and removes the second byte of Address from the DE Bus.

7. The Addressed device fetches data and places it on the Data Exchange. If it can not fetch data fast enough to respond as shown in the timing diagram, the processor will have to be microprogrammed to wait one or more microcycles before attempting to read the data. Note that no special handshake Wait logic exist in the 7300 to hold up microcycles until data is valid.

8. Data will remain on the DE Bus until the Leading Edge of DR appears designating the beginning of the next I/O process.

As shown by the timing diagram the minimum cycle time required is 2.0 μs. If one microcycle were skipped before reading, up to 6.0 μs could be used to access the data. If two microcycles are skipped the access time could be up to 10.0 μs, etc.

C. WRITE MEMORY/PERIPHERAL (Figures 3A and 3B)

The sequence for writing is as follows:

1. Get Priority to use the Bus.

2. Raise DR and drive the most significant byte of address onto the DE Bus.

3. All receiving devices perform the required timing with respect to DR and clock the first byte of Address from the DE Bus.

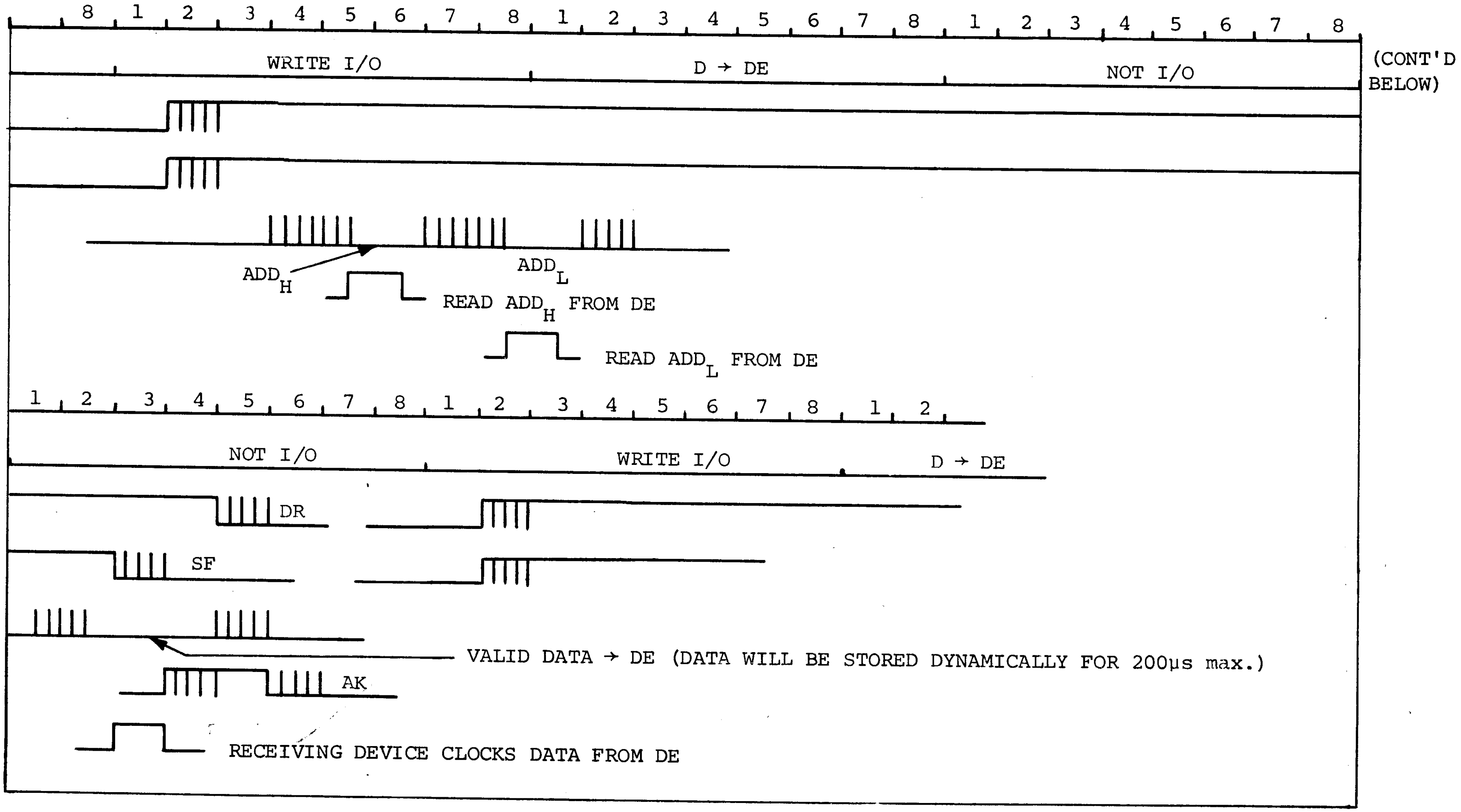


FIGURE 3A - WRITE INPUT/OUTPUT FULL CYCLE

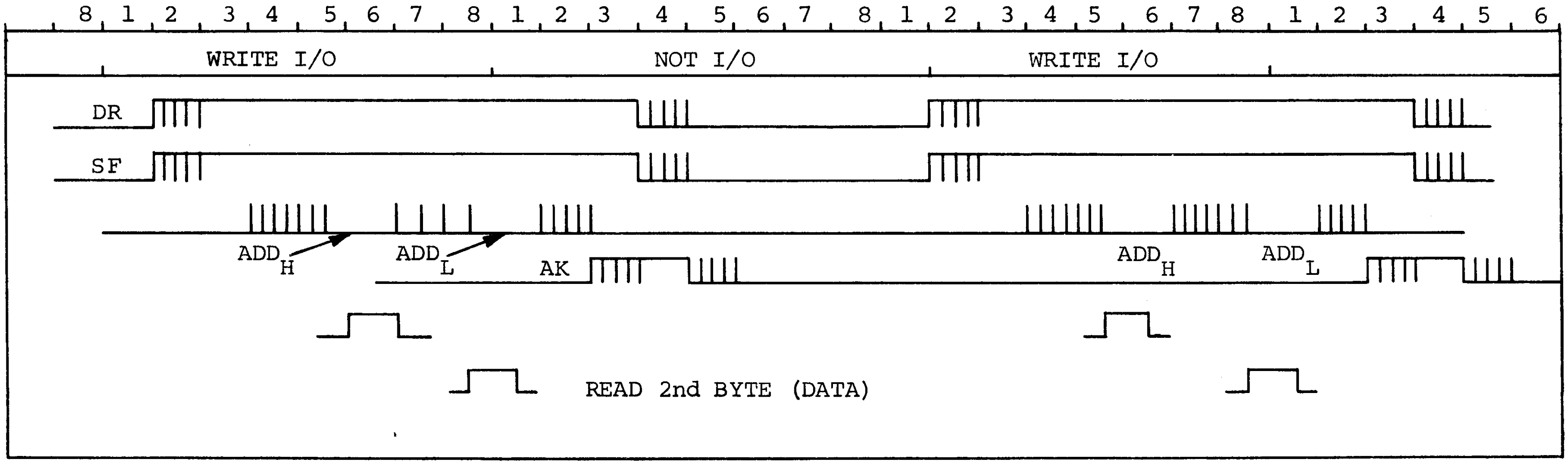


FIGURE 3B - WRITE INPUT/OUTPUT

4. The second byte of address is placed on the DE Bus.

5. The receiving devices clock the second byte of address from the DE Bus. Once the second byte has been read, only the specified device being addressed begins the cycle to read data from the DE Bus.

6. The processor extracts the data to be written from one of its registers, routes it through the ALU and places it on the Data Exchange. ALU operations may be performed as it passes through the ALU.

7. The processor lowers SF to indicate that the data is valid for writing.

8. The receiving device raises AK as soon as it has registered the Data.

9. When the processor senses AK it lowers DR and removes the Data from the DE Bus.

10. When the receiving device senses the absence of DR it lowers AK.

The cycle described above may be referred to as a full cycle. A short cycle consisting of only the transfer of the first two bytes may be desirable when communicating with peripheral devices which need receive only 16 bits. The short cycle can be accomplished by designing the peripheral controller so that it returns AK immediately upon receiving the second byte of Address. At the same time the microprogrammer must program the system to initiate a Write Cycle but not transfer Data. In this manner only one microinstruction is required to communicate with a peripheral. Note that short cycles may be initiated every other microinstruction for the fastest case.

## D. DIRECT MEMORY ACCESS

In performing Direct Memory Access it is necessary for the peripheral device to get priority as well as hold the processor off the Bus. A priority chaining network is used to resolve priority conflicts in multiprocessor and direct memory access systems. The processor is restrained from accessing the Bus by driving its Run line Low. The sequence of operations are as follows (See Figure 4).

1. Request and Get Priority to use the Bus.

2. Drive the Run Line Low.

3. Sense FRUN from the processor to signify that it is no longer running.

4. Sense the absence of Drive/Receive to verify that any previous transfers are completed.

5. Do either of the two types of transfers as described above.

6. Release the request for priority and the Run line.

## E. SYSTEM TIMING

The 7300 is clocked by a 2-phase non-overlapping clock. These clocks provide precise timing for the execution of each of 8 steps which form the microinstruction execution cycle. All microinstructions execute in the same 8 step sequence except when performing read I/O from the Data Exchange. The basic microcycle is shown in Figure 5.

The microinstructions are accessed as 22-bit words but executed one-half at a time. During time slots 7, 8 and 1 the first half is transmitted over the M bus; during 2, 3, 4, 5 and 6 the second half is transmitted. The first half is read from the M Bus during time slot 1 (TS1) and the second half is read during time slot 4 (TS4).

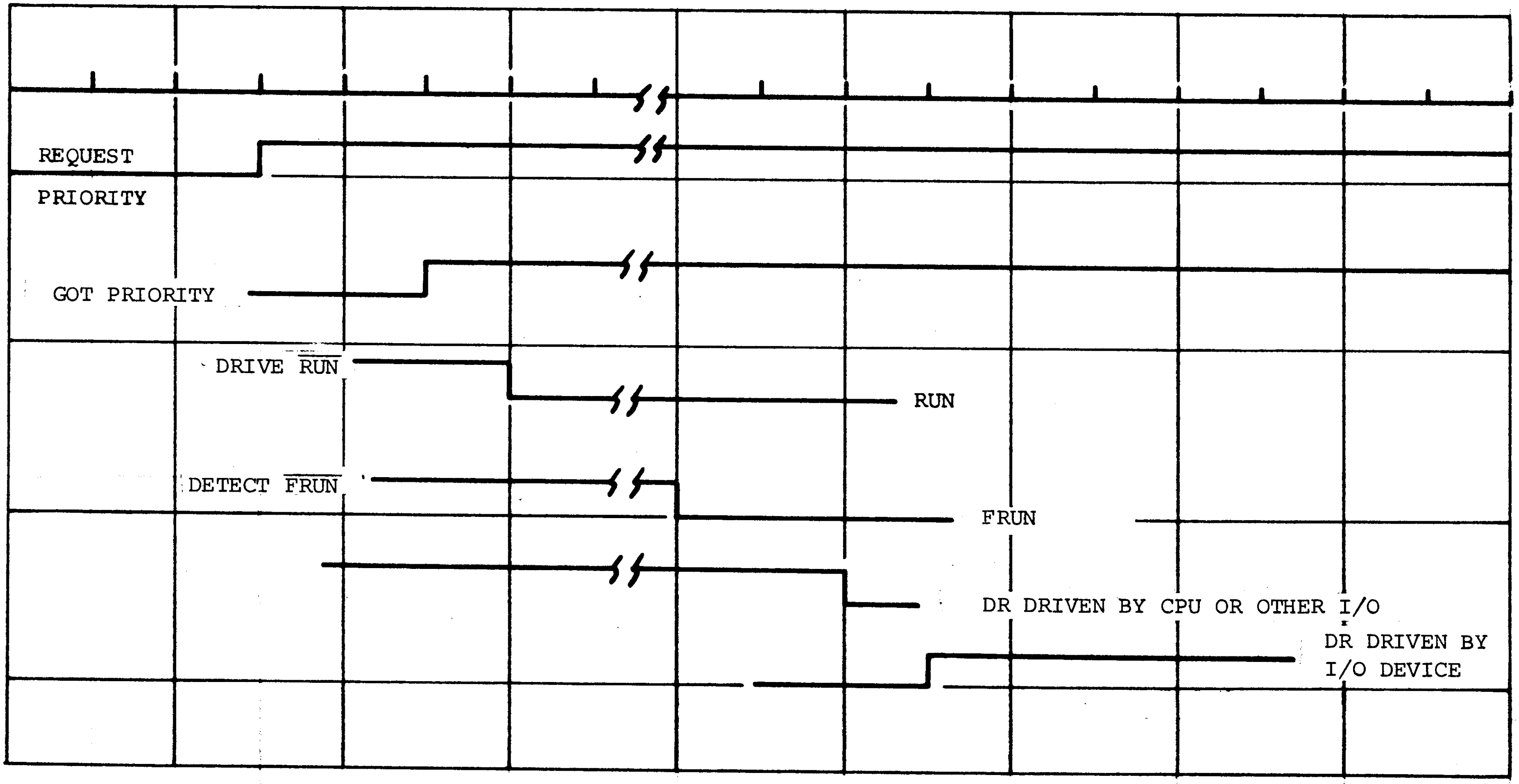

REQUEST
PRIORITY
GOT PRIORITY
DRIVE $\overline{\text{RUN}}$
RUN
DETECT $\overline{\text{FRUN}}$
FRUN
DR DRIVEN BY CPU OR OTHER I/O
DR DRIVEN BY
I/O DEVICE

FIGURE 4 - DIRECT MEMORY ACCESS

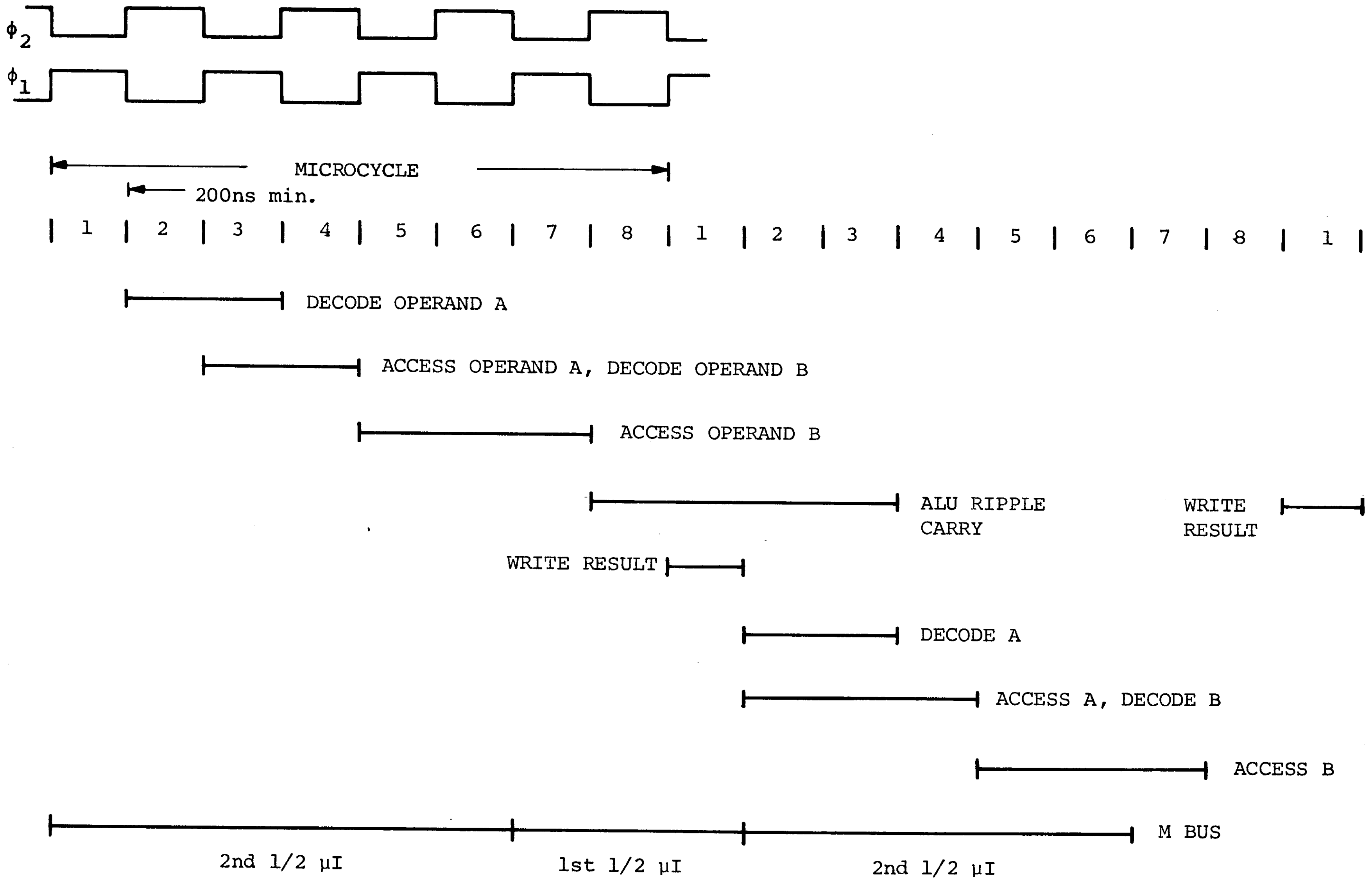


FIGURE 5 - PROCESSOR TIMING

## F. INTERRUPT PROCESSING

Interrupts may be initiated from four sources:

Three Data Exchange Interrupt Lines
Initialization (power up clear) Reset Line

Several types of Interrupt schemes are possible utilizing the existing facilities of the Data Exchange.

Probably the simplest approach is to design the Controllers to drive the Interrupt Attention Line to Initiate an Interrupt. The Processor can then respond to the Interrupt by polling each Peripheral Controller in the order of priority desired to determine the source and type of interrupt. Four levels of priority are provided by the Processor which may be adequate in a small system to alleviate the need for polling.

Another way of implementing interrupt capability is to let the Controllers drive Interrupt Attention and at the same time request Interrupt Priority through a priority chaining network. The Processor will respond to the Interrupt Attention Line by locking up the state of the priority chaining network and performing a Peripheral Input operation as described above.

An I/O Address is set aside for the Interrupt System and all Controllers sense the same address which specifies that the Interrupting Controller with highest priority should return an 8-bit Interrupt Address on the Data Exchange. This operation of gathering the Interrupt Address from the Interrupting Controller works exactly the same as the Peripheral Input process described above. More logic is required in the Controller but a much more efficient Interrupt System is generated.

## III. FUNCTIONAL DESCRIPTION

The 7300 microprocessor is a two-chip microprogrammed CPU. The general logic block diagrams of the two chips are shown in Figures 6 and 7. The characteristic features of the 7300 are:

1. Logical and arithmetic operations are performed on 8-bit bytes by a 4-bit ALU. Decimal arithmetic may be performed efficiently with special microinstructions.

2. Memory access is performed via an 8-bit Data Exchange Bus allowing parallel transfer of the 16-bit address as two byte transfers with subsequent transfer of 8-bit data.

3. A multilevel microinterrupt hardware system is provided. Three levels of priority are provided for the CPU.

4. Data Exchange handshake lines allow peripherals and/or other processors to use the bus to gain direct access to memories and peripherals.

5. Sixteen 8-bit general purpose registers are provided.

6. An 8-bit status register and five additional temporary status bits are provided.

7. A 32-word by 8-bit stack with two 6-bit stack pointers is provided. The stack pointers are binary counters which may address the 32-word stack and the 16 general registers.

8. An 8-bit macroinstruction register is provided for instruction decoding.

9. An Instruction Mapping Array is fed by the macroinstruction register to allow immediate transfer of microprogram control to one of up to 50 locations, determined by the macroinstruction being decoded. Up to three independent mappings of the macroinstruction may be obtained to select, for instance, the proper address preparation, defer and execute sequences to perform the specified macroinstruction.

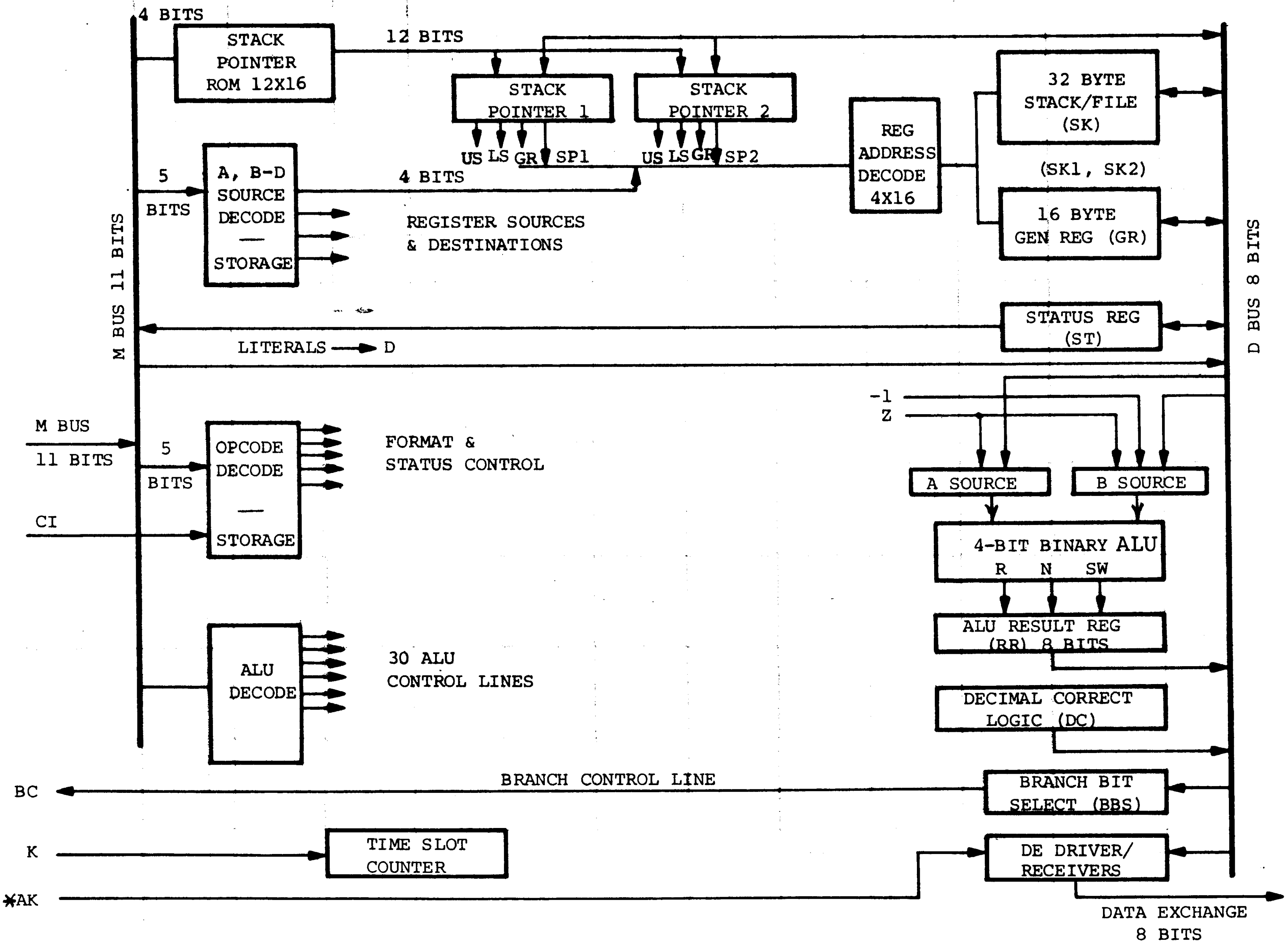


FIGURE 6 REGISTERS AND ALU CHIP

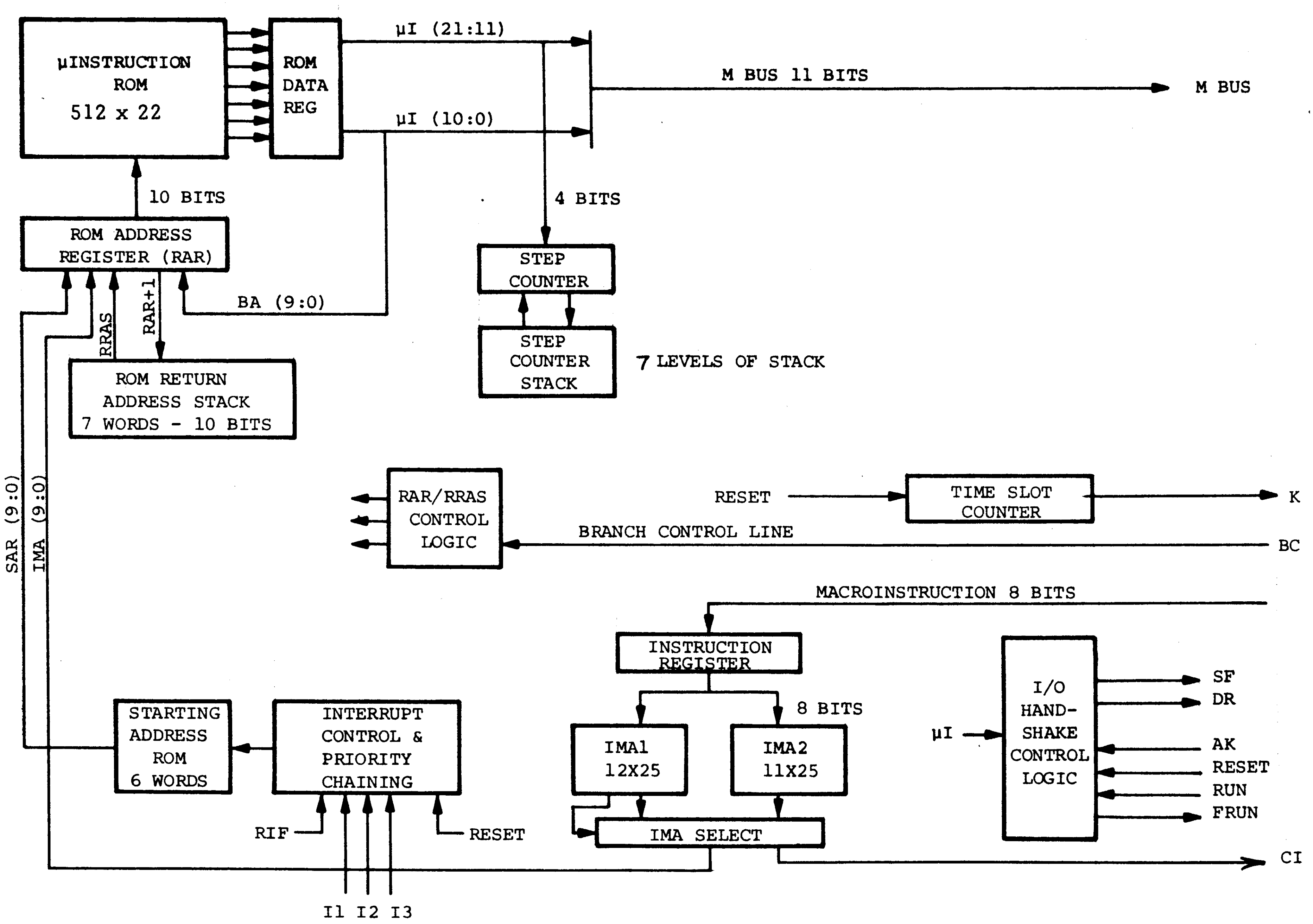


FIGURE 7 - MICROINSTRUCTION ROM CHIP

10. A 9-bit Microinstruction ROM Return Address Stack (RRAS) in addition to a 9-bit ROM Address Register (RAR) allows 7 levels of microinstruction subroutining or coroutining. When the RAR is loaded with a new address, the old address may be stored in the RRAS (Branch and Mark type operation). The RAR may be loaded from the RRAS (Return from subroutine type operation) or the contents of the RAR and RRAS may be swapped (CoCall type operation). The microinstruction CoCall capability effectively multiplies the power of the mapping array for multiple byte fetch and execute operations and other interlinked segmented tasks.

11. Microprogramming is facilitated by a powerful microinstruction set. Microinstructions are 22 bits long. There are several different microinstruction formats to optimize microinstruction bit usage.

12. Microinstructions are stored in a 512-word by 22-bit Read Only Memory (ROM). This provides sufficient storage for large, powerful microprograms.

13. Pipelining, or simultaneous execution of different phases of several microinstructions, allows an execution rate of up to 250,000 microinstructions per second (4.0 μs per microcycle). I/O operations may be performed in parallel with other microinstruction operations, allowing memory access times to be overlapped with execution times on both the micro and macro levels.

14. The processor operates on power supplies of +5V and -12V (nominal), and requires two +5V to -12V (nominal) clock signals. The clock signals are free-running and need not be gated by any other signals. All other inputs and outputs are TTL compatible.

15. The execution of one microinstruction is initiated each microcycle. A microcycle consists of eight time states (four clock cycles), TS1 through TS8.

## A. REGISTERS AND ALU CHIP (RALU)

The Registers, Arithmetic and Logic Unit chip (RALU) contains the following basic data manipulating components of the processor:

### 1. Data Exchange Drivers and Receivers

The eight data exchange lines are bidirectional input and output lines which form the path for all data and addresses in and out of the CPU. Sixteen-bit addresses are transmitted with two 1-byte transfers with the most significant byte being transmitted first. The 65 384-byte address space may be divided up in any manner between memory and peripheral addresses. The I/O handshake lines do not differentiate between transfers for memories and for peripherals.

### 2. Arithmetic and Logic Unit (ALU)

The ALU is the main data processing unit of the CPU. It includes an 8-bit fully parallel carry adder/subtractor and, in addition, it can perform various logical and shifting operations on two 8-bit operands. It receives its operands from a single 8-bit internal bus, the D Bus, and outputs to another Temporary ALU Result Register (RR) and the D Bus. Carries into the ALU may be constant, or may be determined by bits of the permanent or temporary status. The temporary status bits are loaded according to the ALU result. Decimal corrections are made by recirculating the result of a binary add through the ALU a second time and adding A Literal in accordance with the contents of the RR and the carries from the two BCD digits.

## 3. Decimal Arithmetic

The arithmetic unit of the 7300 performs decimal arithmetic using the 2-4-2-1 decimal code. This code is listed below:

| Binary Count | 2-4-2-1 Decimal Code |
|---|---|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | |
| 0110 | |
| 0111 | |
| 1000 | |
| 1001 | |
| 1010 | |
| 1011 | 5 |
| 1100 | 6 |
| 1101 | 7 |
| 1110 | 8 |
| 1111 | 9 |

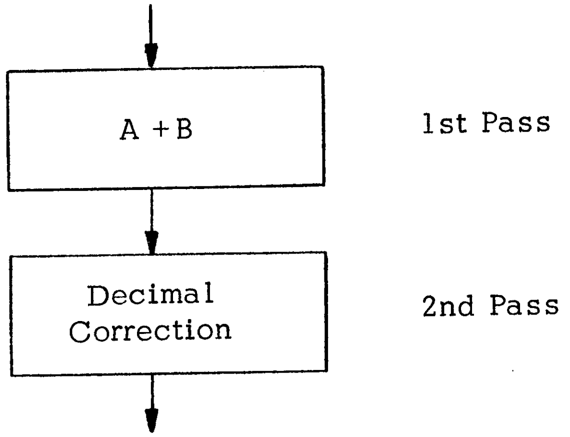
This code has several advantages over other decimal codes, such as BCD, Excess -3, 8-4-$\bar{2}$-$\bar{1}$, 5-2-1-1, when implementing decimal arithmetic. The overwhelming advantages of 2-4-2-1 are.

1. The code self-complementing, allowing the binary complementer to be used.
2. The code for decimal zero is the same as the code for binary zero, allowing the binary test and branch to be used.

3. The code for decimal one is the same as the code for binary one, allowing the means of presetting the input carry to be used when incrementing in decimal.

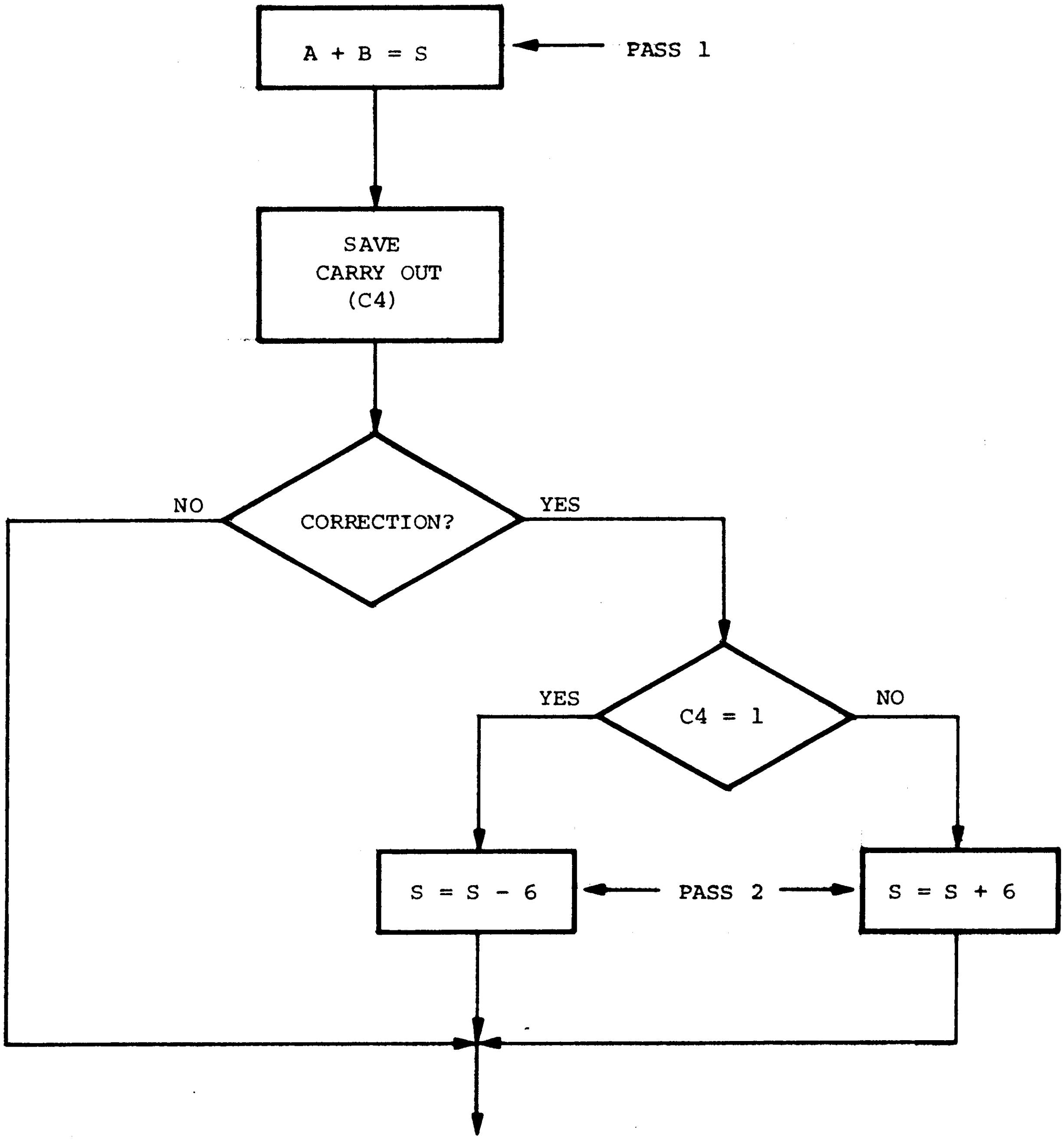
4. The decimal correction logic is no worse than that for other codes.

Decimal addition is performed in the 7300 with two passes through the Arithmetic/Logic Unit (ALU). The first pass performs the addition and the second pass performs the decimal correction.



The 7300 is organized as a byte or two digit machine. This allows the microprogrammer to perform the decimal addition and correction of two digits in two microinstructions or two passes of the ALU.

During the first pass, the two operands (two digits each) are added as two 8-bit binary numbers allowing the carry from the least significant digit to ripple on to the most significant digit. The carry out of the most significant digit is saved at this time if needed for the next decimal add. During the second pass the decimal correction code is given allowing the ALU to add a decimal correction of +6 or -6 to each of the digits from the immediately previous addition if needed. This process is then repeated until the number of required digits have been exercised. The flow chart for the decimal addition of one and two digit words are shown in Figures 8 and 9 respectively.



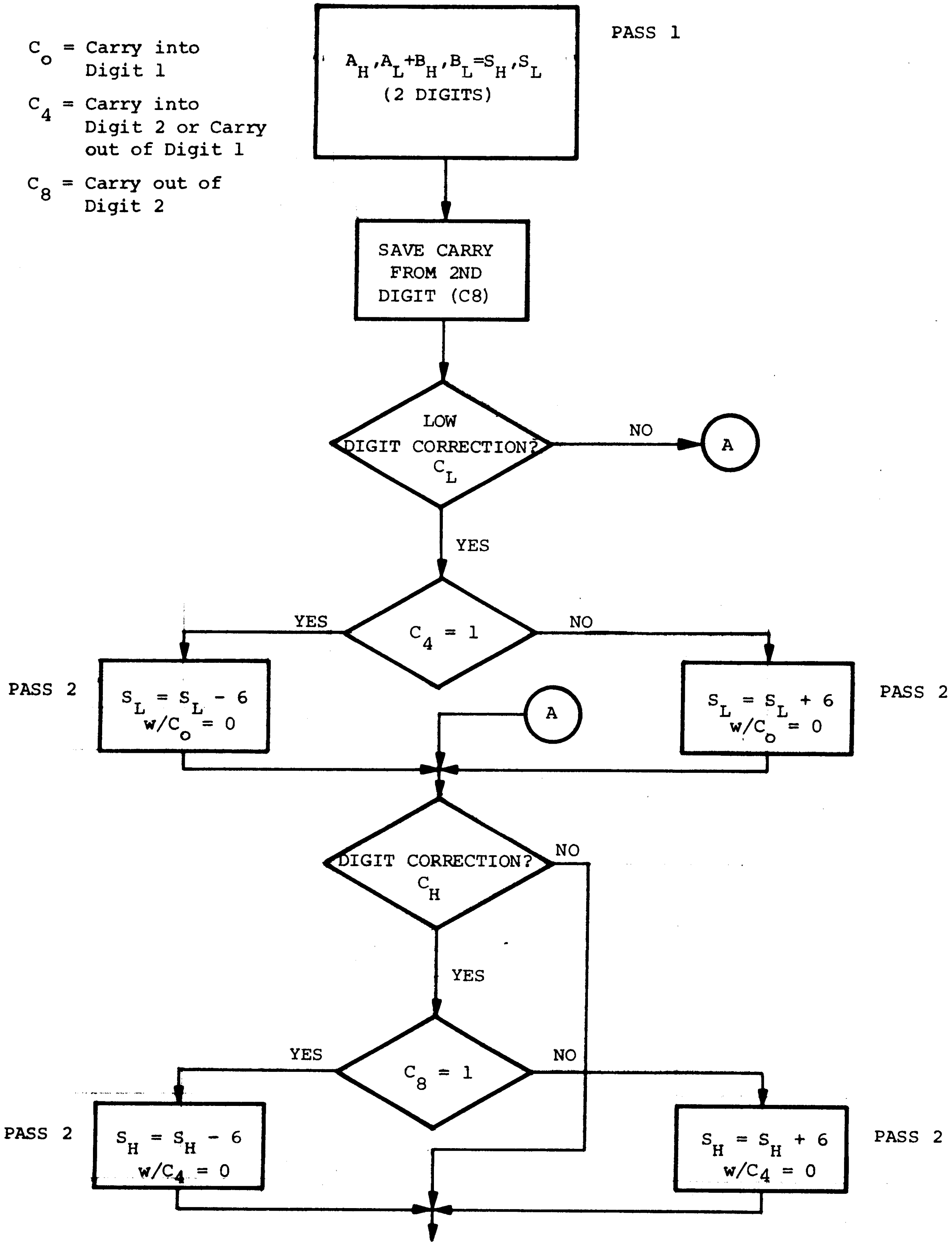
C4 = Carry Out

FIGURE 8 - 2-4-2-1 DECIMAL ARITHMETIC ON ONE DIGIT

$C_o$ = Carry into Digit 1

$C_4$ = Carry into Digit 2 or Carry out of Digit 1

$C_8$ = Carry out of Digit 2



PASS 1
$A_H, A_L + B_H, B_L = S_H, S_L$
(2 DIGITS)
SAVE CARRY FROM 2ND DIGIT (C8)
LOW DIGIT CORRECTION? $C_L$
NO
A
YES
$C_4 = 1$
YES
NO
PASS 2
$S_L = S_L - 6$ w/$C_o$ = 0
A
$S_L = S_L + 6$ w/$C_o$ = 0
PASS 2
DIGIT CORRECTION? $C_H$
NO
YES
$C_8 = 1$
YES
NO
PASS 2
$S_H = S_H - 6$ w/$C_4$ = 0
$S_H = S_H + 6$ w/$C_4$ = 0
PASS 2

FIGURE 9 - 2-4-2-1 DECIMAL ARITHMETIC ON TWO DIGITS

The correction logic required is that which detects binary codes 5 through A, inclusive. These are the invalid codes for the 2-4-2-1 decimal representation. The equations would be as follows:

For the least significant digit,

$C_L = B4 \cdot \overline{B3}\ (\overline{B1} + \overline{B2}) + \overline{B4} \cdot B3\ (B1 + B2)$

where, B1, B2, B3, B4 represent Bits 1-4 respectively of the lsd.

For the most significant digit,

$C_H = B8 \cdot \overline{B7}\ (\overline{B5} + \overline{B6}) + \overline{B8} \cdot B7\ (B5 + B7)$

where, B5, B6, B7, B8 represent Bits 1-4 respectively of the msd.

As can be seen in Figure 9, when correcting the two digits on the second or correction pass, the low order digit carry (C4) is not allowed to ripple to the high order digit. The high order digit carry input is forced to zero.

4. Signed Decimal Arithmetic

Signed decimal addition and subtraction is performed by the algebraic addition of the two operands. For subtraction the sign of the subtrahend is complemented before the addition is performed. Figure 10 shows the flow chart for this process.

5. A and B Source Registers

The information fed to the ALU is stored in the A and B source registers. These registers are dynamic storage nodes which are normally updated each microinstruction. They may be left unmodified if specified by a previous A or previous B source field. Storage in these registers may last for up to 200 $\mu$s.

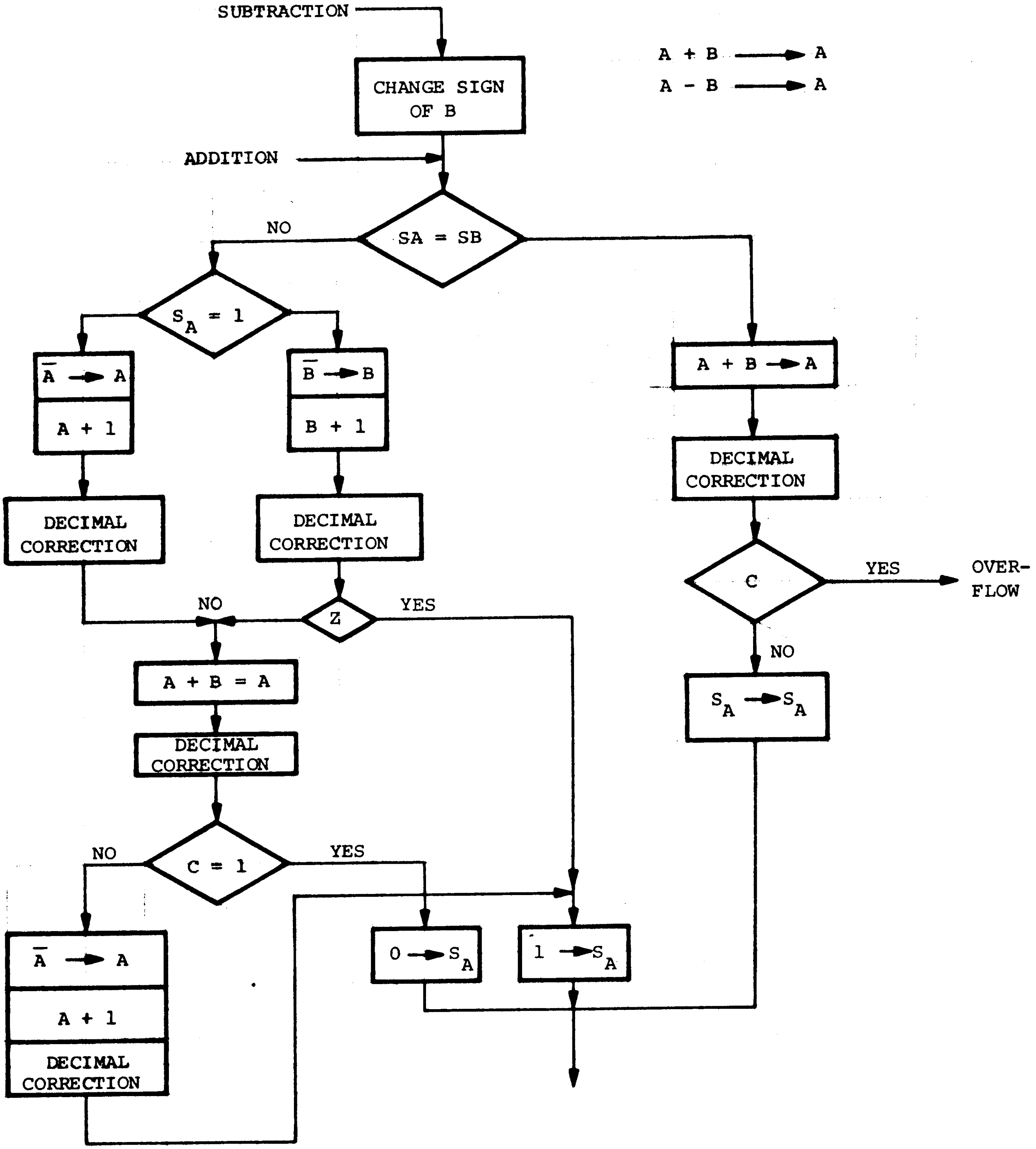


FIGURE 10 - ALGEBRAIC ADDITION AND SUBTRACTION OF TWO NUMBERS WITH SIGN MAGNITUDE REPRESENTATION

6. Status Register

The Status Register consists of three temporary status bits, TM, TC and TZ and three permanent or final status bits C, M and Z as shown below. The temporary bits hold conditions resulting from recent microinstruction and are loaded automatically as a result of these microinstructions, while the final status bits are changed only by certain special Register Control Format OpCodes and are usually used to hold long-range status. The three final status bits may be transferred to and from the D Bus and all of them may be directly loaded with data from the D Bus. All of the temporary and final status bits may be tested, set and reset by Literal Format Microinstructions.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | C | M | Z |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Final Status Register

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | TC | TM | TZ |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Temporary Status Register

7. Temporary Status Bits

TM

TM is loaded by most ALU operations with the most significant bit of the result and is thus a negative result.

TC

TC is loaded by arithmetic ALU operations and shift operations. For arithmetic operations it is loaded with the Carry Out of the high order bit of the adder. For left shift operations it is loaded with the bit that is shifted out of the most significant bit. For right shift operations it is loaded with the bit that is shifted out of the least significant bit.

TZ

TZ is loaded by most ALU operations. It is set if the result is zero and cleared otherwise.

## 8. Final Status Bits

C

C may be set, cleared or loaded with the contents of TC. It is usually used to indicate a carry or shift out of the ALU as a result of the previous macroinstruction.

M

M may be set, cleared or loaded with the contents of TM. It is usually used to indicate a negative signed arithmetic result of the previous macroinstruction.

Z

Z may be set, cleared or loaded with either the contents of TZ or the AND of the contents of TZ with the old contents of Z.

## 9. General Registers

The General Registers consist of sixteen 8-bit registers implemented as a 16-word by 8-bit RAM with one output and one input port. Only one operand may be fetched at a time from the General Registers and placed on the D Bus to be used by the ALU, loaded into the Final or Temporary Status, outputted to DE (7:0), loaded into the stack pointers or loaded into the stacks. The General Registers may be used to hold addresses, data, macroinstruction words, etc., in any combination.

## 10. Stack and Stack Pointers

The Stack consists of a 32-word by 8-bit, single port RAM. Unlike the General Registers, which are directly addressed by microinstructions, the Stack is addressed by the contents of two address counters called the Stack Pointers, each of which addresses only one location in the Stack at a given time. The contents of that location may be put on the D Bus or may be loaded from the D Bus.

The two pointers give the effect of having two stacks and allows for list type processing of two multiple precision integers. The pointers may be used in an unmodified manner or they may be incremented or decremented in order to implement first-in last-out push-down stacks. The two pointers may also be utilized to construct double buffer schemes of data movement.

If the Stack Pointers (SP1 and SP2) are incremented the contents of their address is accessed before incrementation. If the Stack Pointers are decremented the contents of their address is accessed before decrementation.

When the incrementation or decrementation appears in the B-D source field the same address is used for both source and destination.

Each microinstruction is to be treated independently such that Stack Pointer operations are not dependent on previous or subsequent microinstructions.

When using the Stack as one or two sets of general registers the General Register field of the macroinstruction is.loaded from the D Bus into the Stack Pointer as follows:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Stack Pointers cause the access of zero values when addressing beyond the boundaries of the general register and stack addressing space.

The Stack Pointer ROM may be mask-programmed with up to 16 pairs of Stack Pointer values. When a Branch and Load Stack pointer operation is performed one of the pairs of values are loaded into each pointer. This enables the programmer to set up the pointers when branching to routines which are written for any combination of pointer values.

### 11. ALU Result Register (RR)

The RR dynamically holds the result of the ALU operation until it can be loaded onto the D Bus and clocked into the selected Destination register. The result of an ALU operation is not clocked into the Destination register until after the next two A and B sources have been fetched and loaded. The previous ALU result may be loaded as a source into the A and the B source registers in cases where sequential subsequent microinstructions utilize results of previous microinstructions. The RR is implemented with dynamic storage nodes.

## B. MICROINSTRUCTION ROM CHIP (MIR)

The MIR (Figure 7) contains the Microprogram Control Unit of the CPU and I/O handshake logic. The I/O handshake logic consists of decoding and sequencing logic which causes the handshake lines to be operated as described above for the Data Exchange. The CPU does not contain any special logic for causing the microinstruction execution to hold up and wait for data to become available. If the DE is referenced as a source for data, it will be read and assumed to be valid. Thus the microprogrammer must be aware of memory and peripheral timing to be assured that data is valid when accessed.

The Microprogram Control Unit consists of the following components:

1. Macroinstruction Register (IR)

The IR is an 8-bit register which holds macroinstructions for decoding by the Instruction Mapping Array. The IR is always loaded by a Register Control Microinstruction with Decode I specified in the OpCode field.

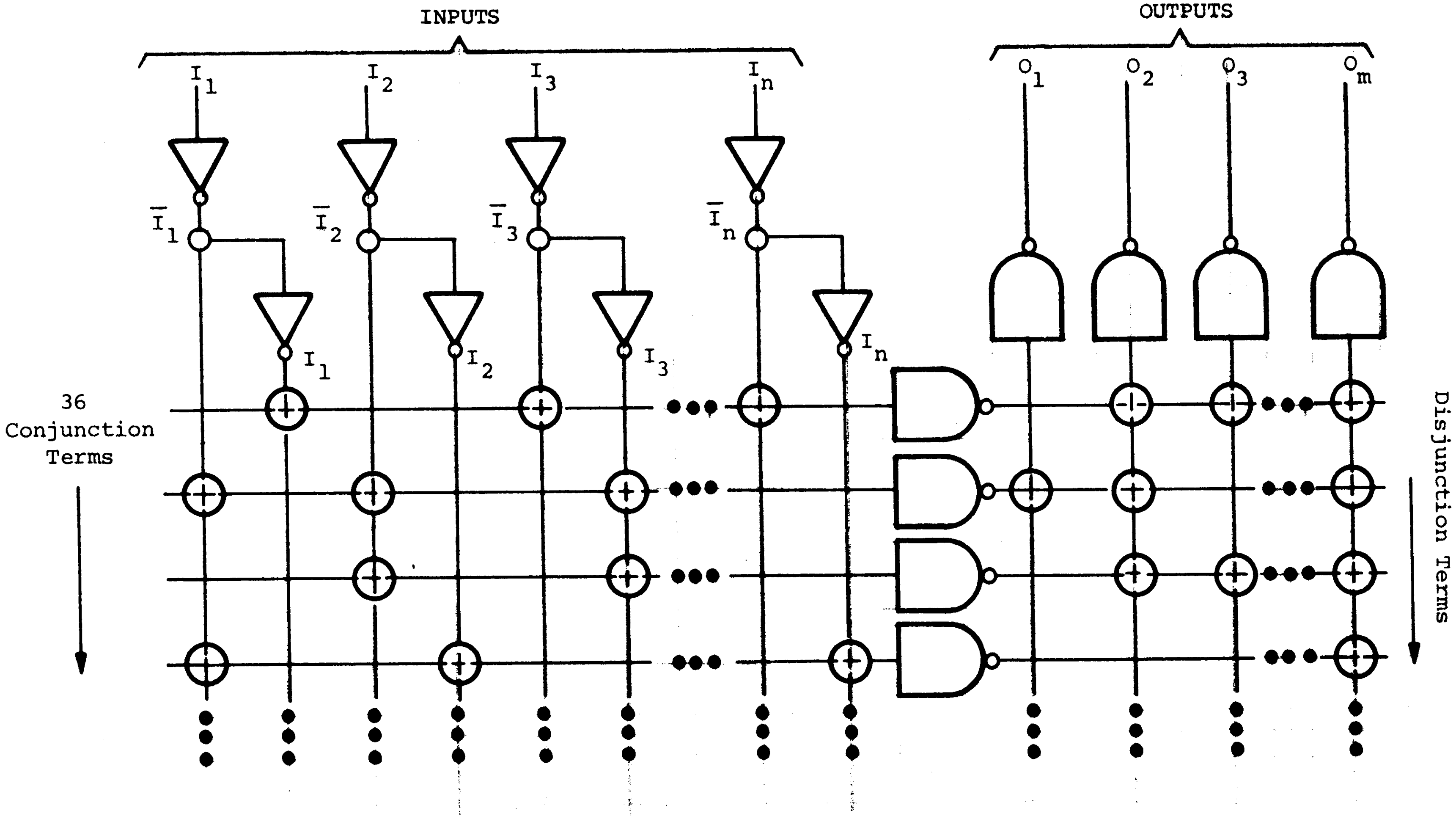
2. Instruction Mapping Arrays (IMA)

The IMAs and their associated control and multiplexing circuitry convert the outputs of the IR into the microinstruction ROM addresses of the proper microinstruction routines to perform the specified macroinstruction.

a. Mapping Array Design

A diagram of one of the IMAs is shown in Figure 11. Each of the n inputs is inverted and the true and complement signals for each are run to the inputs of 25 2n-input NOR gates, called conjunction (AND) gates. These NOR gates are exactly like the Address Decode Network for any MOS ROM; each input device may be fabricated or not according to the oxide mask used to make the chip. Thus, it is possible to specify simply and easily which inputs are to be connected to the conjunctions (AND function). In Figure 11, connected inputs are shown as circles at the intersections of the input lines and conjunction lines. For instance, the first conjunction is connected to $I_1$, $\bar{I}_3$ and $\bar{I}_n$. Its output is zero unless $I_1$ is zero and $I_3$ and $I_n$ are one, but it does not depend upon the state of $I_2$ since neither $I_2$ nor $\bar{I}_2$ is connected as an input.

The outputs of the conjunction gates form the inputs of another array of NOR gates, called disjunctions (OR function). Again, the inputs may or may not be connected to the gates, as specified by the programmer. If a conjunction output is one for which there is an input device in a disjunction term gate, the output of that disjunction will be zero. If no such conjunction term is one, the output (disjunction term) will be one. In most applications, the conjunction terms will be disjoint; i.e., at most, one conjunction will be one for a given input word. In this case, the outputs for a given



$$\overline{I}_1 \wedge 1 \wedge I_3 \wedge \ldots \wedge I_n = O_1, \overline{O}_2, \overline{O}_3, \ldots, \overline{O}_m$$

$$I_1 \wedge I_2 \wedge \overline{I}_3 \wedge \ldots \wedge 1 = \overline{O}_1, \overline{O}_2, O_3, \ldots, \overline{O}_m$$

$$1 \wedge I_2 \wedge \overline{I}_3 \wedge \ldots \wedge 1 = O_1, \overline{O}_2, \overline{O}_3, \ldots, \overline{O}_m$$

$$I_1 \wedge \overline{I}_2 \wedge 1 \wedge \ldots \wedge \overline{I}_n = O_1, O_2, O_3, \ldots, \overline{O}_m$$

FIGURE 11 MAPPING ARRAY CIRCUIT

input word will be determined solely by whether there is an input device in each disjunction gate for the conjunction term that is one; a device equals a zero, no device equals a one.

### b. Decoder Design

As shown in Figure 12 , the Macroinstruction Decode Array consists of an 8-input, 11-output mapping array (IMA 1), a 9-input, 12-output mapping array (IMA 2) and output selector gates which select the outputs of one of the two arrays to provide the 10-bit MIR Address and a tenth bit called CI, to be discussed later.

To make the macroinstruction decoding flexible, three different types of decode microinstructions are provided - Decodes 1, 2 and 3. (Decode 1, in addition to initiating the decode operation, loads the IR with the contents of the DE Bus). The conditions which determine the Mapping Array output to be used are the 12th output bit of Mapping Array 2 and whether a Decode 3 operation is being performed.

If a Decode 3 operation is being performed, the outputs of Mapping Array 1 are always used. If a Decode 1 or Decode 2 operation is being performed, Mapping Array 2 outputs are used unless the extra, 12th output bit of Mapping Array 2 is one. Note that Decode 1 and Decode 2 outputs are differentiated whether a Decode 1 or a Decode 2 operation being performed is one of the inputs to Mapping Array 2.. (Decode 3 is a Don't Care condition for Mapping Array 2 since the outputs of that array are never used for a Decode 3.)

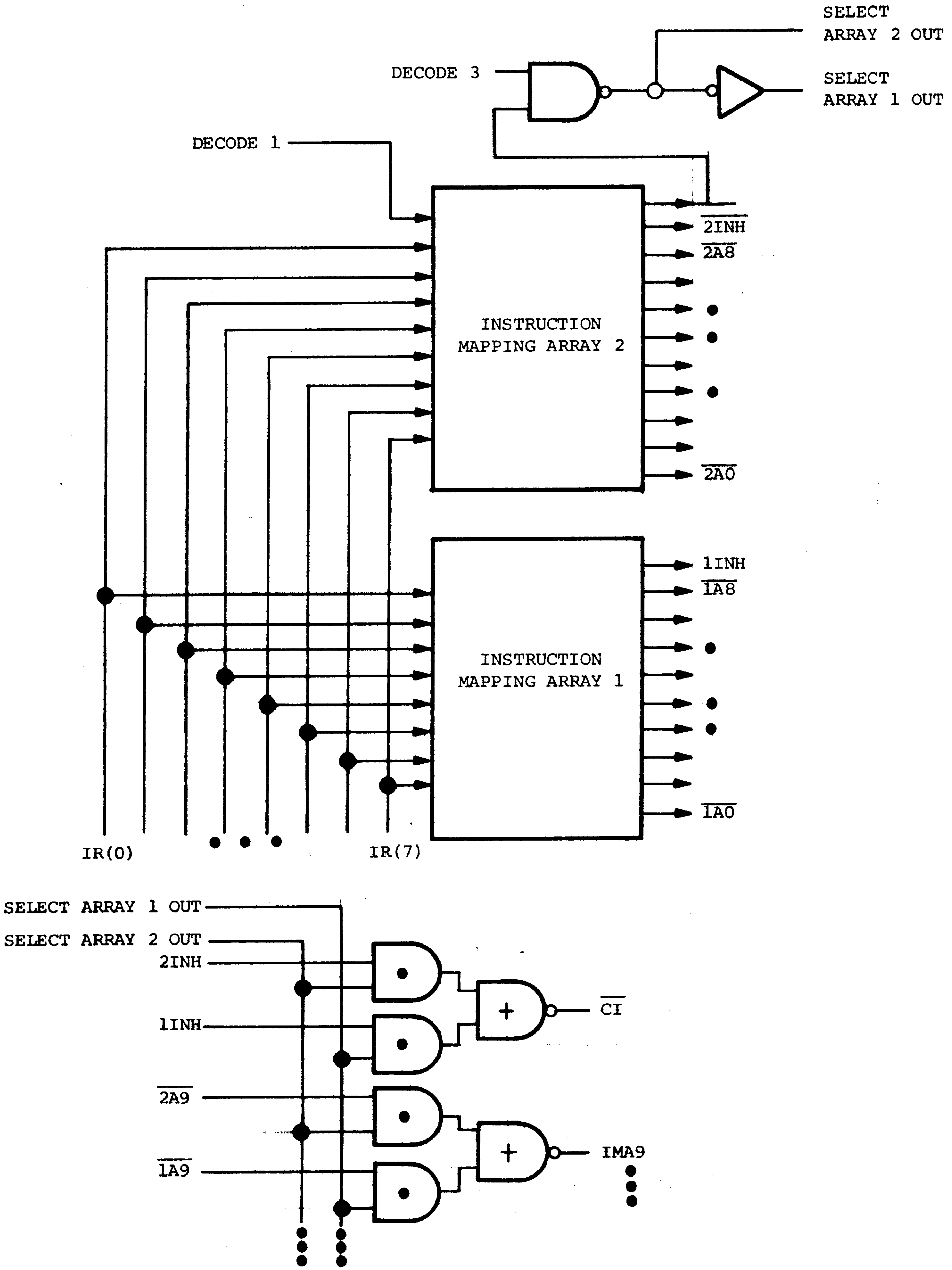


FIGURE 12 - MICROINSTRUCTION DECODER

### 3. Microprogram Read-Only Memory (MIROM)

The MIROM is a 512-word by 22-bit mask-programmable ROM which contains the microprograms. It receives 9 address input bits from the RAR and outputs 22 microinstruction bits each microcycle. The outputs are transmitted to the RALU in two groups of 11 bits on the M Bus. The high-order bits, MI (21:11), are transmitted during TS7, TS8 and TS1, and the low-order bits, MI (10:0), are transmitted during TS2, TS3 and TS4.

### 4. ROM Address Register (RAR) and ROM Return Address Stack (RRAS)

The RAR is a 9-bit polynomial counter based on a primitive polynomial. Each microcycle, it is incremented (shifted) to point to the next MIROM address. It can increment through all 511 non-zero addresses, but cannot shift out of zero; thus, if the RAR is loaded with zero it will remain zero until loaded with another address. The RRAS provides a first-in last-out stack for return addresses from microinstruction subroutines and coroutines.

The RAR + 1 is loaded into the RRAS each time a Branch and Mark operation is used. The RRAS is loaded into the RAR for Microreturn operations and a swap occurs for CoCall operations. No overflow conditions are detected for the RRAS so it is necessary for the microprogrammer to keep track of the number of microsubroutine levels being used.

Decode operations cause the RAR to be pushed into the RRAS. The RRAS is implemented as a single-port 7-word RAM in a manner similar to the Stack on the RALU chip. Branch operations always execute in two microcycles worth of time. The first microcycle is used to access the register specified by the A source field and select a specified bit. The Branch Control Line BC then represents the one or zero state of the selected bit at the end of the first microcycle. The Branch Address is conditionally clocked into the RAR and the second microcycle is used to access the next microinstruction.

### 5. Starting Address ROM (SAR) and Microinterrupt Control Logic

The SAR is a 5-word by 9-bit mask-programmable ROM which produces a microinstruction address to be loaded into the RAR when a Return to Fetch microinstruction (normally the last microinstruction in each execute routine) is executed. The proper address for the SAR is selected by the Microinterrupt Control Logic, a priority-chaining network. The levels of priority for the five inputs to the chaining network in decreasing order of priority are:

1. Reset
2. Interrupt 1
3. Interrupt 2
4. Interrupt 3
5. Return to Instruction Fetch

If a reset occurs, the microinstruction execution is halted immediately. The other control inputs are sensed only at the Return to Instruction Fetch time.

### 6. Step Counter (SC)/Step Counter Stack (SCS)

The 7300 may be utilized to implement macro-operations which do multiple precision arithmetics. An SC is included to make it easy to call a microroutine and execute it a specific number of times without the need for including a software step count function in each subroutine. The SC is loaded by a Branch Microinstruction and may be interleaved up to seven times by subroutines which push and pop its contents into the SCS. Each time the Branch and Mark and Load Step Counter operation is performed the SC is loaded into the SCS. Each time a Return from Microsubroutine Loop (RLR) SC is decremented. If an RLR is executed and the SC is equal to zero, then the top of the SCS is loaded into the SC and the SCS is popped.

Thus the SC may be interleaved just as microsubroutines are interleaved. The SCS is seven levels deep. The SC is a 4-bit polynomial counter.