THE MCS-4 - AN LSI MICRO COMPUTER SYSTEM

F. Faggin, M. Shima*, M. E. Hoff, Jr., H. Feeney, S. Mazor

Intel Corporation, Santa Clara, California
*Busicom Corporation, Tokyo, Japan

ABSTRACT

The MCS-4 is a totally self-contained four-bit general purpose microprogrammable computer in component form. It consists of four basic elements: the CPU (central processing unit), the ROM (read only memory), the RAM (random access memory), and the SR (shift register). They are fabricated by MOS silicon gate technology and packaged in economical sixteen pin DIPs to minimize board area and reduce system cost.

Using combinations of these standard building blocks, any degree of customization may be built into these powerful microprogrammed four-bit computers. Using as few as two devices, a CPU and a ROM, a four-bit microprogrammed dedicated computer may be built for under $50.

This paper describes this new micro-computer set, highlighting the system partitioning and the basic CPU hardware instruction set.

INTRODUCTION

LSI technology provides new components for system design -- microcomputers. With the MCS-4, a four-bit LSI microcomputer set fabricated with the p-channel silicon gate MOS process, the power of a general purpose computer is available to every system designer as an alternative to conventional designs of random logic systems. The MCS-4 can provide the same control and computing functions of a minicomputer in as few as two sixteen pin DIPs and costs nearly two orders of magnitude less.

This set of components is not designed to compete with the minicomputer, but rather to extend the concept of the dedicated computer into new applications where the minimization of cost and size are very important, but where speed is not a mojor factor.

A major goal in the development of the MCS-4 was to devise a computer architecture allowing system partitioning into a minimum number of sixteen pin packages. The result is a set of standard LSI building blocks which are manufacturable in high volume and are flexible enough to be used in a variety of applications.

The partitioning resulted in four MOS microcomputer building blocks:

CPU -- four-bit parallel processor

ROM + I/O -- 256 words x eight bits / four I/O lines metal mask programmable ROM

RAM + output - 80 four-bit characters / four output lines

SR -- ten-bit shift register (serial in, serial out, parallel out)

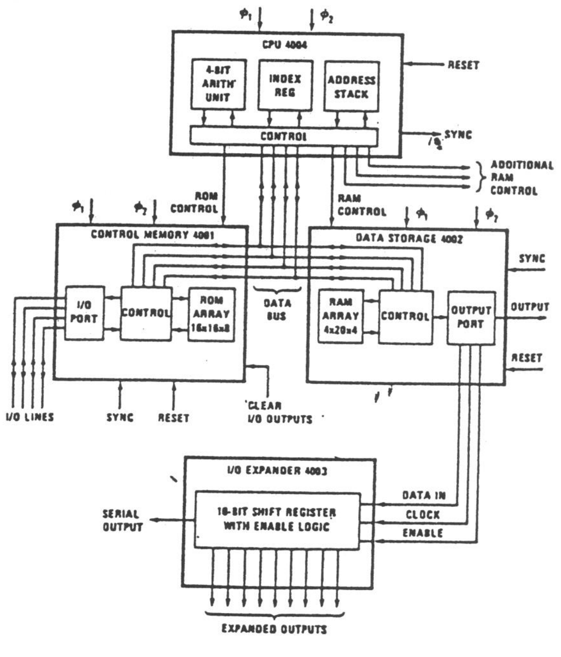


Figure 1. MCS-4$^{T.M.}$ BASIC SYSTEM

Reprinted from the *Proceedings of the IEEE '72 Region Six Conference* and with permission from Intel.

Digital Object Identifier 10.1109/MSSC.2009.931980

Time multiplexing was used extensively to reduce the pin count and minimize circuit area. The MCS-4 is a totally self-contained system; no additional interface components are necessary.

The circuits operate with a single supply voltage of -15v and two non-overlapping clock phases, ϕ_1 and ϕ_2.

The heart of each system is a single chip CPU which performs all the control and data processing functions. Directly interfacing with the CPU are ROMs which store microprograms and data tables and RAMs which store data and pseudo instructions. The MCS-4 communicates with peripheral devices through I/O "ports" provided on each ROM and RAM chip. In addition, ten-bit parallel shift registers can expand the I/O capability of the system.

Figure 1 shows the basic MCS-4 system. All address data and instruction communication is carried out on the four-bit nondedicated data bus; system synchronization and memory control are provided by the CPU.

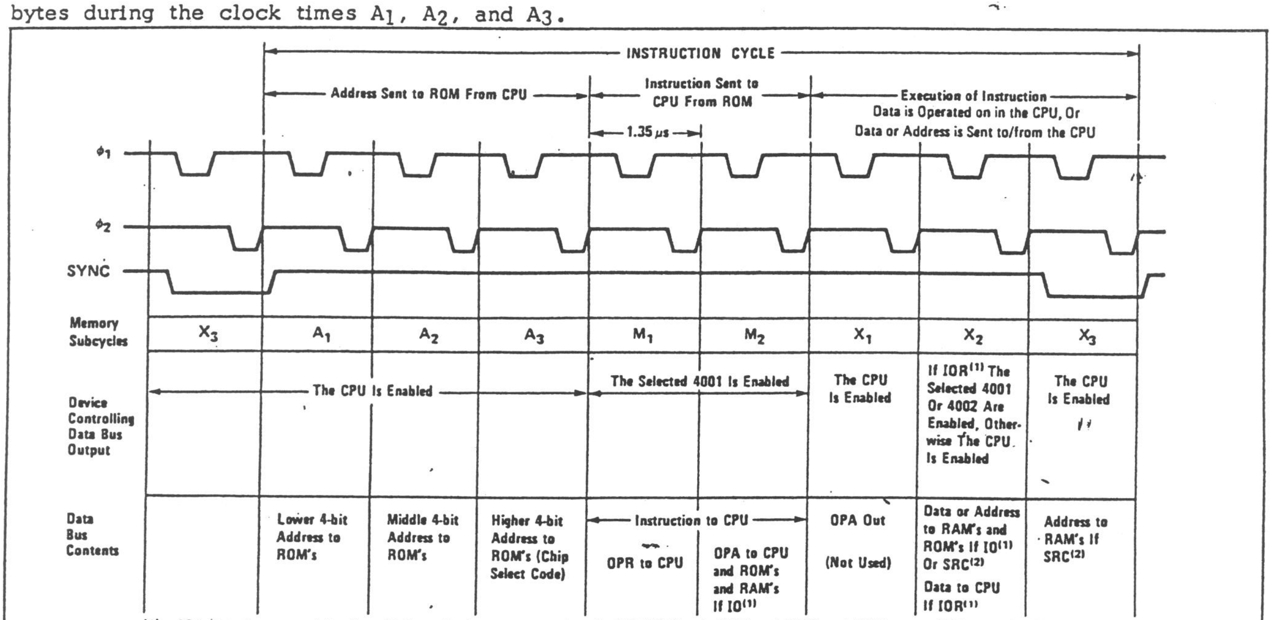
The basic system timing for a typical instruction is shown in Figure 2. A total of eight clock periods is required for the addressing, fetching, and execution of a single word instruction. An instruction cycle is completed in 10.8 µs when operating at a clock frequency of 750 khz. In a typical operating sequence, the CPU sends out a twelve-bit address in three successive four-bit bytes during the clock times A_1, A_2, and A_3. This address is received by the ROMs, and an eight-bit instruction word is selected (one of 4096 instructions). The eight-bit instruction is sent back to the data bus in two four-bit bytes during the following two cycles, M_1 and M_2. During the final three clock cycles, X_1, X_2, and X_3, the instruction is interpreted and executed.

THE CPU (SYSTEM CONTROL)

This single chip CPU is the fundamental component of the system. It provides the complete memory addressing, instruction interpretation, and control for the total system. The CPU contains the following functional blocks (refer to Figure 3):

Address Register and Address Incrementer

The address register is a dynamic RAM array of four twelve-bit words. One word is used to store the effective address (program counter) and the other three words are used as a stack for subroutine calls. Thus nesting up to three levels is possible. The program counter is incremented on each instruction cycle as the address is sent out from the CPU. With the twelve-bit address the CPU can directly address up to sixteen ROMs, each containing 256 eight-bit words.


INSTRUCTION CYCLE
Address Sent to ROM From CPU
Instruction Sent to CPU From ROM
Execution of Instruction Data is Operated on in the CPU, Or Data or Address is Sent to/from the CPU
1.35 µs
φ1
φ2
SYNC
Memory Subcycles
X3
A1
A2
A3
M1
M2
X1
X2
X3
Device Controlling Data Bus Output
The CPU Is Enabled
The Selected 4001 Is Enabled
The CPU Is Enabled
If IOR(1) The Selected 4001 Or 4002 Are Enabled, Otherwise The CPU Is Enabled
The CPU Is Enabled
Data Bus Contents
Lower 4-bit Address to ROM's
Middle 4-bit Address to ROM's
Higher 4-bit Address to ROM's (Chip Select Code)
Instruction to CPU
OPR to CPU
OPA to CPU and ROM's and RAM's If IO(1)
OPA Out (Not Used)
Data or Address to RAM's and ROM's If IO(1) Or SRC(2) Data to CPU If IOR(1)
Address to RAM's If SRC(2)

[1] IO instructions control the flow of information between accumulator in CPU, I/O lines in ROM's and RAM's and RAM storage. IOR stands for IO Read. In this case the CPU will receive data from RAM storage locations or I/O input lines of 4001's.

[2] The SRC instruction designates the chip number and address for a following IO instruction.

Figure 2. MCS-4T.M. BASIC INSTRUCTION CYCLE

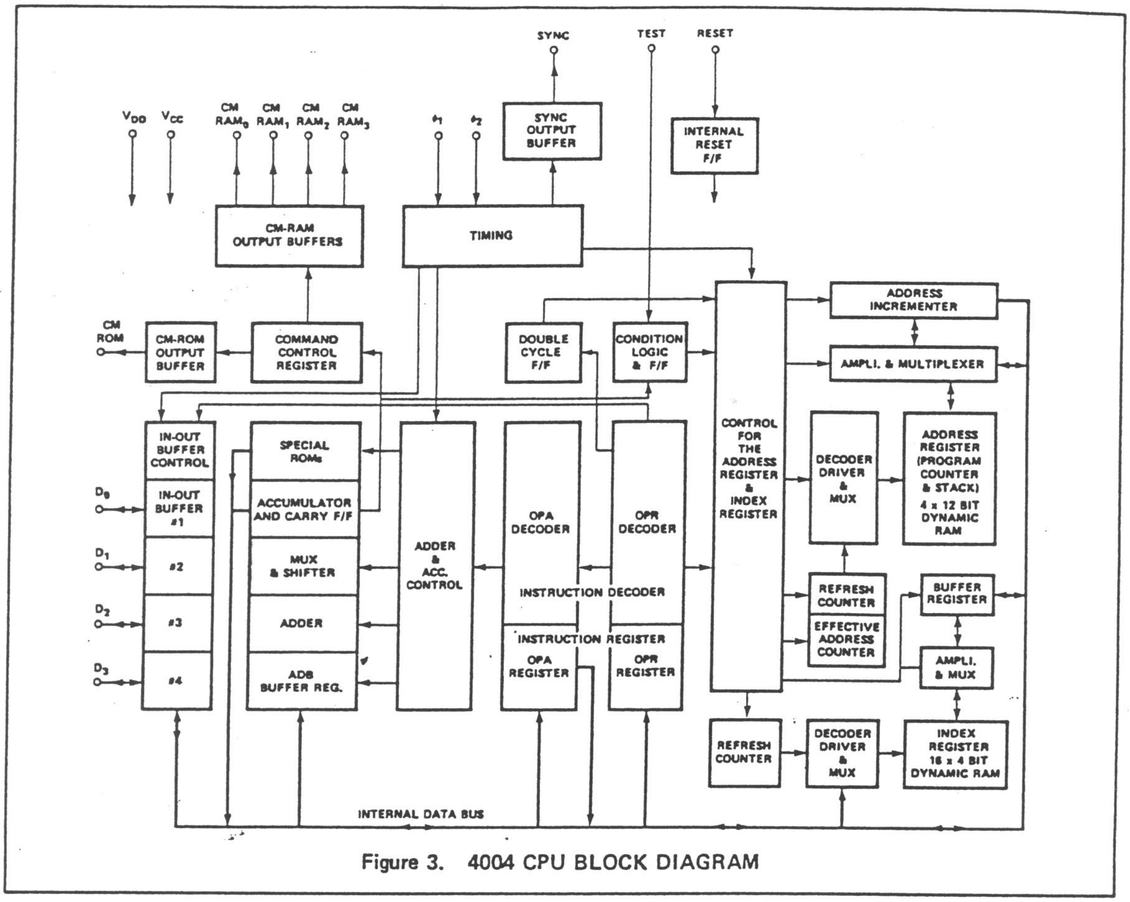


Figure 3. 4004 CPU BLOCK DIAGRAM

Index Register

Sixteen four-bit general purpose registers are provided for use as scratch pad registers or memory pointers. These registers may be accessed individually or in eight-bit register pairs.

Four-bit Arithmetic Unit

This is a four-bit parallel adder with ripple through carry. All arithmetic instructions are executed using this functional block. Since this system operates on four-bit bytes of data, it can be used directly for computation in BCD. The result of an operation between two BCD numbers is computed in binary; using the decimal adjust accumulator (DAA) instruction, the result of the operation and the carry bit are converted back to BCD.

Instruction Register and Decoder

Eight-bit instructions are received, stored, and decoded to generate control signals for all other functional blocks.

In addition to the basic functional blocks, internal timing, ROM and RAM enable control, and data bus I/O control are included in the peripheral circuitry.

The instruction repertoire is permanently stored in an associative memory in the CPU (the instruction decoder). It consists of three basic groups of instructions:

Sixteen machine instructions (basic control of the address and index registers)

Fifteen I/O and RAM instructions (communication with peripheral devices through the ROM and RAM I/O ports and character storage in RAM memory)

Fourteen accumulator group instructions (basic accumulator processing instructions)

The complete instruction set including the mnemonic, binary code, and description is presented in Figure 4.

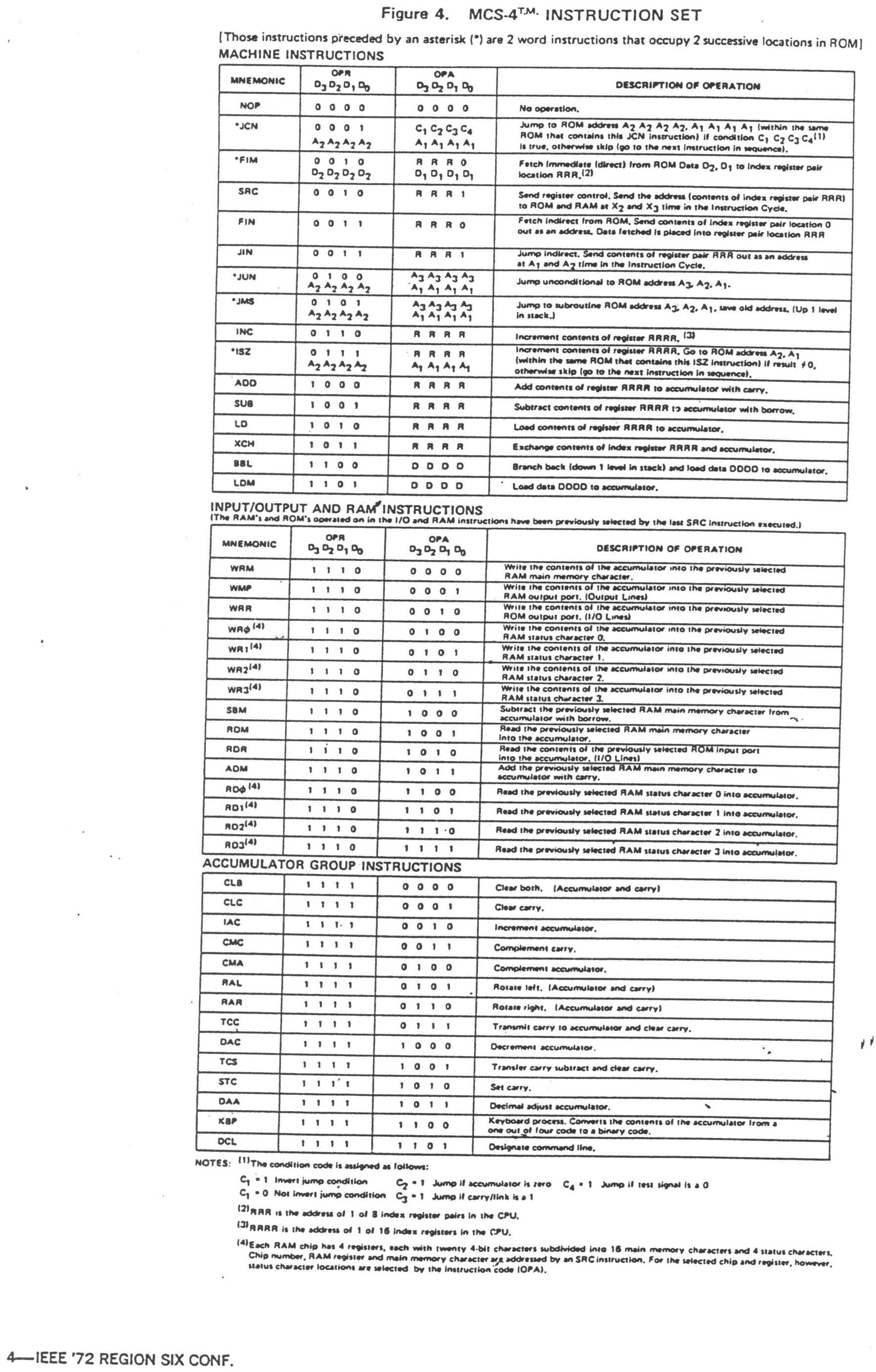


Figure 4. MCS-4T.M. INSTRUCTION SET

[Those instructions preceded by an asterisk (*) are 2 word instructions that occupy 2 successive locations in ROM]

MACHINE INSTRUCTIONS

MNEMONIC	OPR $D_3 D_2 D_1 D_0$	OPA $D_3 D_2 D_1 D_0$	DESCRIPTION OF OPERATION
NOP	0 0 0 0	0 0 0 0	No operation.
*JCN	0 0 0 1 $A_2 A_2 A_2 A_2$	$C_1 C_2 C_3 C_4$ $A_1 A_1 A_1 A_1$	Jump to ROM address $A_2\ A_2\ A_2\ A_2$, $A_1\ A_1\ A_1\ A_1$ (within the same ROM that contains this JCN instruction) if condition $C_1\ C_2\ C_3\ C_4$(1) is true, otherwise skip (go to the next instruction in sequence).
*FIM	0 0 1 0 $D_2 D_2 D_2 D_2$	R R R 0 $D_1 D_1 D_1 D_1$	Fetch immediate (direct) from ROM Data D_2, D_1 to index register pair location RRR.(2)
SRC	0 0 1 0	R R R 1	Send register control. Send the address (contents of index register pair RRR) to ROM and RAM at X_2 and X_3 time in the Instruction Cycle.
FIN	0 0 1 1	R R R 0	Fetch indirect from ROM. Send contents of index register pair location 0 out as an address. Data fetched is placed into register pair location RRR
JIN	0 0 1 1	R R R 1	Jump indirect. Send contents of register pair RRR out as an address at A_1 and A_2 time in the Instruction Cycle.
*JUN	0 1 0 0 $A_2 A_2 A_2 A_2$	$A_3 A_3 A_3 A_3$ $A_1 A_1 A_1 A_1$	Jump unconditional to ROM address A_3, A_2, A_1.
*JMS	0 1 0 1 $A_2 A_2 A_2 A_2$	$A_3 A_3 A_3 A_3$ $A_1 A_1 A_1 A_1$	Jump to subroutine ROM address A_3, A_2, A_1, save old address. (Up 1 level in stack.)
INC	0 1 1 0	R R R R	Increment contents of register RRRR. (3)
*ISZ	0 1 1 1 $A_2 A_2 A_2 A_2$	R R R R $A_1 A_1 A_1 A_1$	Increment contents of register RRRR. Go to ROM address A_2, A_1 (within the same ROM that contains this ISZ instruction) if result ≠ 0, otherwise skip (go to the next instruction in sequence).
ADD	1 0 0 0	R R R R	Add contents of register RRRR to accumulator with carry.
SUB	1 0 0 1	R R R R	Subtract contents of register RRRR to accumulator with borrow.
LD	1 0 1 0	R R R R	Load contents of register RRRR to accumulator.
XCH	1 0 1 1	R R R R	Exchange contents of index register RRRR and accumulator.
BBL	1 1 0 0	D D D D	Branch back (down 1 level in stack) and load data DDDD to accumulator.
LDM	1 1 0 1	D D D D	Load data DDDD to accumulator.

INPUT/OUTPUT AND RAM INSTRUCTIONS

(The RAM's and ROM's operated on in the I/O and RAM instructions have been previously selected by the last SRC instruction executed.)

MNEMONIC	OPR $D_3 D_2 D_1 D_0$	OPA $D_3 D_2 D_1 D_0$	DESCRIPTION OF OPERATION
WRM	1 1 1 0	0 0 0 0	Write the contents of the accumulator into the previously selected RAM main memory character.
WMP	1 1 1 0	0 0 0 1	Write the contents of the accumulator into the previously selected RAM output port. (Output Lines)
WRR	1 1 1 0	0 0 1 0	Write the contents of the accumulator into the previously selected ROM output port. (I/O Lines)
WRϕ (4)	1 1 1 0	0 1 0 0	Write the contents of the accumulator into the previously selected RAM status character 0.
WR1 (4)	1 1 1 0	0 1 0 1	Write the contents of the accumulator into the previously selected RAM status character 1.
WR2 (4)	1 1 1 0	0 1 1 0	Write the contents of the accumulator into the previously selected RAM status character 2.
WR3 (4)	1 1 1 0	0 1 1 1	Write the contents of the accumulator into the previously selected RAM status character 3.
SBM	1 1 1 0	1 0 0 0	Subtract the previously selected RAM main memory character from accumulator with borrow.
RDM	1 1 1 0	1 0 0 1	Read the previously selected RAM main memory character into the accumulator.
RDR	1 1 1 0	1 0 1 0	Read the contents of the previously selected ROM input port into the accumulator. (I/O Lines)
ADM	1 1 1 0	1 0 1 1	Add the previously selected RAM main memory character to accumulator with carry.
RDϕ (4)	1 1 1 0	1 1 0 0	Read the previously selected RAM status character 0 into accumulator.
RD1 (4)	1 1 1 0	1 1 0 1	Read the previously selected RAM status character 1 into accumulator.
RD2 (4)	1 1 1 0	1 1 1 ·0	Read the previously selected RAM status character 2 into accumulator.
RD3 (4)	1 1 1 0	1 1 1 1	Read the previously selected RAM status character 3 into accumulator.

ACCUMULATOR GROUP INSTRUCTIONS

CLB	1 1 1 1	0 0 0 0	Clear both. (Accumulator and carry)
CLC	1 1 1 1	0 0 0 1	Clear carry.
IAC	1 1 1· 1	0 0 1 0	Increment accumulator.
CMC	1 1 1 1	0 0 1 1	Complement carry.
CMA	1 1 1 1	0 1 0 0	Complement accumulator.
RAL	1 1 1 1	0 1 0 1	Rotate left. (Accumulator and carry)
RAR	1 1 1 1	0 1 1 0	Rotate right. (Accumulator and carry)
TCC	1 1 1 1	0 1 1 1	Transmit carry to accumulator and clear carry.
DAC	1 1 1 1	1 0 0 0	Decrement accumulator.
TCS	1 1 1 1	1 0 0 1	Transfer carry subtract and clear carry.
STC	1 1 1′ 1	1 0 1 0	Set carry.
DAA	1 1 1 1	1 0 1 1	Decimal adjust accumulator.
KBP	1 1 1 1	1 1 0 0	Keyboard process. Converts the contents of the accumulator from a one out of four code to a binary code.
DCL	1 1 1 1	1 1 0 1	Designate command line.

NOTES: [1] The condition code is assigned as follows:

C_1 = 1 Invert jump condition C_2 = 1 Jump if accumulator is zero C_4 = 1 Jump if test signal is a 0
C_1 = 0 Not invert jump condition C_3 = 1 Jump if carry/link is a 1

[2] RRR is the address of 1 of 8 index register pairs in the CPU.

[3] RRRR is the address of 1 of 16 index registers in the CPU.

[4] Each RAM chip has 4 registers, each with twenty 4-bit characters subdivided into 16 main memory characters and 4 status characters. Chip number, RAM register and main memory character are addressed by an SRC instruction. For the selected chip and register, however, status character locations are selected by the instruction code (OPA).

THE ROM (CONTROL MEMORY AND I/O)

This device performs two very distinct but independent functions in the system, storage of the instruction sequence, and input/output functions for communication with peripheral devices. The four input/output lines are provided on the ROM rather than the CPU to reduce the individual package pin count. Inputs or outputs may be custom selected to individual system requirements at the same time that the metal mask ROM program is prepared. As more ROMs are added to a system, the number of I/O ports is also increased.

The ROM memory array is organized as 256 eight-bit words. When a particular ROM word is addressed by the CPU, the address is stored in a register included in the ROM and the eight-bit ROM word is multiplexed into two four-bit bytes and sent to the system data bus. Since the CPU can directly address up to sixteen ROMs, a binary code for each ROM must also be programmed in the metal mask.

The ability to program both the memory and the I/O ports provides custom "personality" of the system along with the economic advantage of using high-volume standard LSI devices.

THE RAM (DATA STORAGE AND OUTPUTS)

The RAM also provides two distinct and independent functions, data and pseudo-instruction storage and output communication with external peripheral devices.

The RAM is organized as four registers, each containing sixteen four-bit main memory characters and four four-bit status characters. If the system is operating with decimal arithmetic (all decimal numbers represented in BCD), each register in the RAM can store a complete sixteen digit decimal number. The status characters provide additional storage for the sign, decimal point position, exponent of the number, or other control information. RAM and I/O line addressing is accomplished in a rather unique way. The five CPU command lines (CM-ROM, CM-RAM$_i$) control the way in which ROM and RAM chips interpret the data on the data bus.

RAMs can be arranged in four banks of four chips each, each bank controlled by a separate CM-RAM$_i$ line (refer to Figure 5).

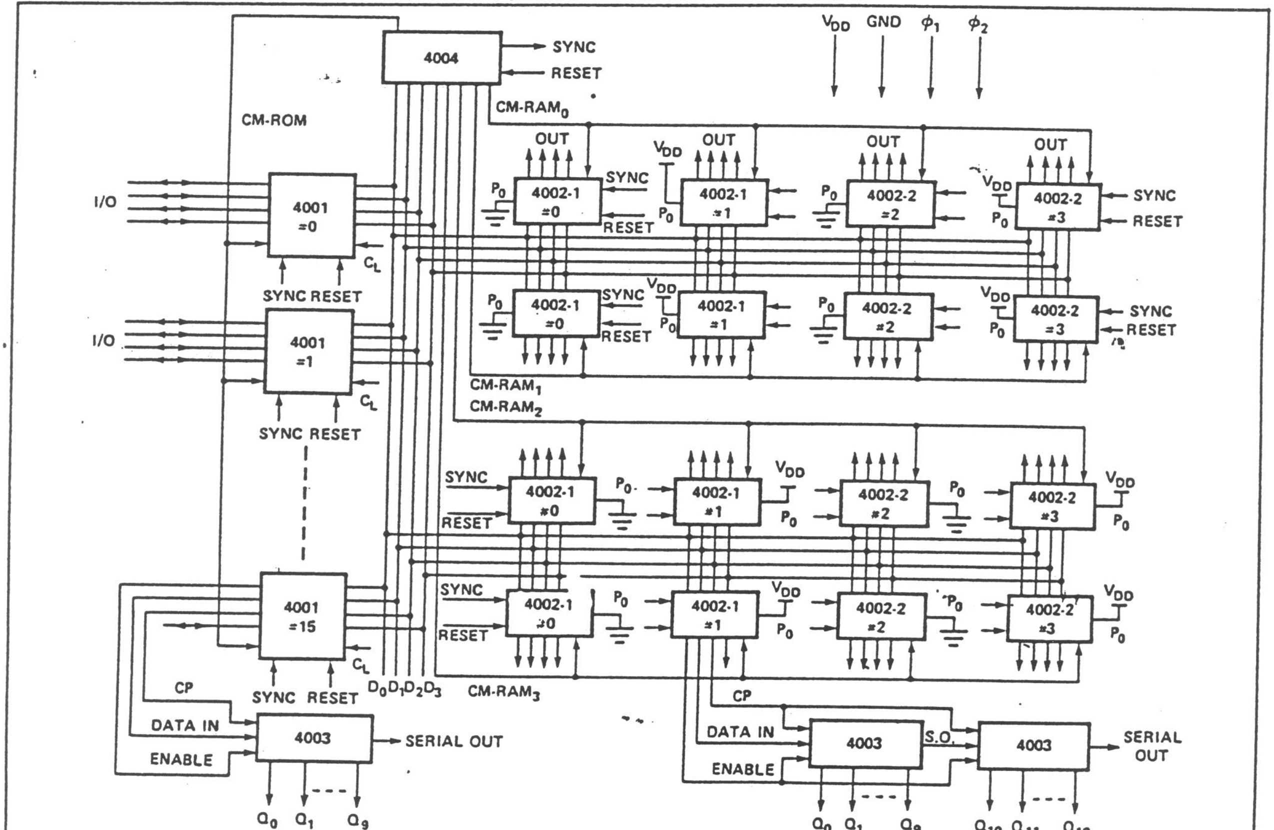
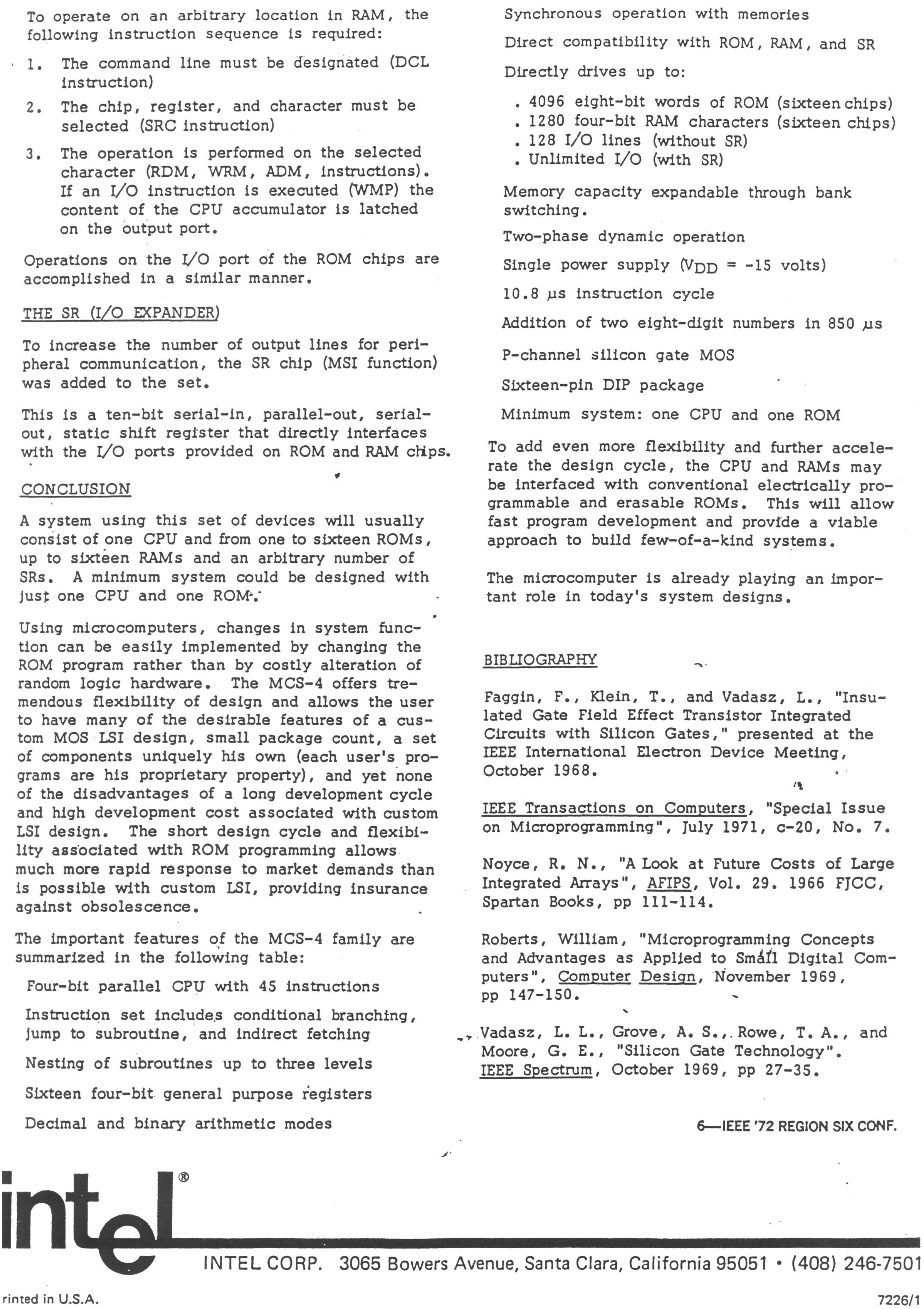


Figure 5. MCS-4T.M. SYSTEM INTERCONNECTION



To operate on an arbitrary location in RAM, the following instruction sequence is required:

1. The command line must be designated (DCL instruction)
2. The chip, register, and character must be selected (SRC instruction)
3. The operation is performed on the selected character (RDM, WRM, ADM, instructions). If an I/O instruction is executed (WMP) the content of the CPU accumulator is latched on the output port.

Operations on the I/O port of the ROM chips are accomplished in a similar manner.

THE SR (I/O EXPANDER)

To increase the number of output lines for peripheral communication, the SR chip (MSI function) was added to the set.

This is a ten-bit serial-in, parallel-out, serial-out, static shift register that directly interfaces with the I/O ports provided on ROM and RAM chips.

CONCLUSION

A system using this set of devices will usually consist of one CPU and from one to sixteen ROMs, up to sixteen RAMs and an arbitrary number of SRs. A minimum system could be designed with just one CPU and one ROM.

Using microcomputers, changes in system function can be easily implemented by changing the ROM program rather than by costly alteration of random logic hardware. The MCS-4 offers tremendous flexibility of design and allows the user to have many of the desirable features of a custom MOS LSI design, small package count, a set of components uniquely his own (each user's programs are his proprietary property), and yet none of the disadvantages of a long development cycle and high development cost associated with custom LSI design. The short design cycle and flexibility associated with ROM programming allows much more rapid response to market demands than is possible with custom LSI, providing insurance against obsolescence.

The important features of the MCS-4 family are summarized in the following table:

Four-bit parallel CPU with 45 instructions

Instruction set includes conditional branching, jump to subroutine, and indirect fetching

Nesting of subroutines up to three levels

Sixteen four-bit general purpose registers

Decimal and binary arithmetic modes

Synchronous operation with memories

Direct compatibility with ROM, RAM, and SR

Directly drives up to:

- 4096 eight-bit words of ROM (sixteen chips)
- 1280 four-bit RAM characters (sixteen chips)
- 128 I/O lines (without SR)
- Unlimited I/O (with SR)

Memory capacity expandable through bank switching.

Two-phase dynamic operation

Single power supply (V_{DD} = -15 volts)

10.8 μs instruction cycle

Addition of two eight-digit numbers in 850 μs

P-channel silicon gate MOS

Sixteen-pin DIP package

Minimum system: one CPU and one ROM

To add even more flexibility and further accelerate the design cycle, the CPU and RAMs may be interfaced with conventional electrically programmable and erasable ROMs. This will allow fast program development and provide a viable approach to build few-of-a-kind systems.

The microcomputer is already playing an important role in today's system designs.

BIBLIOGRAPHY

Faggin, F., Klein, T., and Vadasz, L., "Insulated Gate Field Effect Transistor Integrated Circuits with Silicon Gates," presented at the IEEE International Electron Device Meeting, October 1968.

IEEE Transactions on Computers, "Special Issue on Microprogramming", July 1971, c-20, No. 7.

Noyce, R. N., "A Look at Future Costs of Large Integrated Arrays", AFIPS, Vol. 29. 1966 FJCC, Spartan Books, pp 111-114.

Roberts, William, "Microprogramming Concepts and Advantages as Applied to Small Digital Computers", Computer Design, November 1969, pp 147-150.

Vadasz, L. L., Grove, A. S., Rowe, T. A., and Moore, G. E., "Silicon Gate Technology". IEEE Spectrum, October 1969, pp 27-35.

INTEL CORP. 3065 Bowers Avenue, Santa Clara, California 95051 • (408) 246-7501

rinted in U.S.A.

7226/1