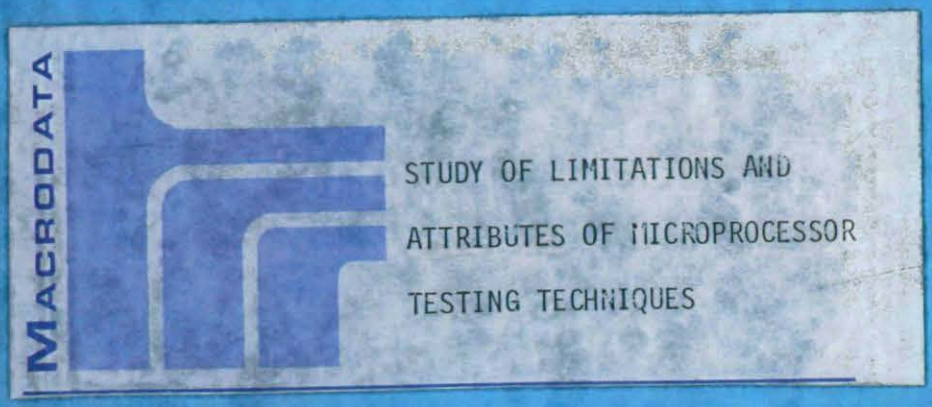


STUDY OF LIMITATIONS AND ATTRIBUTES OF MICROPROCESSOR TESTING TECHNIQUES

Macrodata Corporation
21135 Erwin Street
Woodland Hills, California 91365

STUDY OF LIMITATIONS AND ATTRIBUTES OF MICROPROCESSOR TESTING TECHNIQUES

March 17, 1977

Final Report - NASA Contract NAS8-31954

Prepared By:

Richard McCaskill
Wayne E. Sohl

Prepared For:

George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

TABLE OF CONTENTS

FIGURES

TABLES

I. GENERAL TEST PHILOSOPHY

The problems of testing microprocessors has been elevated past the conventional methods of testing integrated circuits. Just the fact that the microprocessor is not a simple collection of gates in a random format or a well ordered structure, like that of a large scale memory, does not lend itself to conventional means of testing. What is meant by the conventional means of testing is the commonly used DC test checking for inputs and output voltages and currents. This DC testing cannot prove that the microprocessor is operational, because there are from four to six or more levels of logic between the input and output pins. Also the conventional way to test random logic by applying a string of input patterns in a burst will only check for steady-state faults stuck at logic 1 or stuck at logic 0, and will not check for any instruction or data sensitivity.

There presently are many ways that both manufacturers and users are performing testing of microprocessors. These include methods such as self-testing, comparison testing, stored pattern testing, and algorithmic-aided pattern testing.

First Step in Testing

The first item to be considered when testing a microprocessor is to understand the operation and architecture structure of the microprocessor. The operation of the microprocessor is controlled by the execution of an

instruction set unique to each microprocessor. There is a great variety of microprocessors on the market today, ranging from 2- and 4-bit slices to 4-, 8-, and 16-bit complete microprocessor units. But of all the product types, 4-bit slices, like the 2901, and 8-bit microprocessors, like the 8080, have gained the widest acceptance and therefore are good examples to use in describing testing techniques.

In general, a microprocessor has two internal buses: an 8-bit bidirectional data bus, and a 16-bit unidirectional address bus (Figure 1). The data bus carries both the instruction code and data. Instructions are decoded and executed in connection with the appropriate controls in which data going to both the arithmetic logic unit and accumulator can be manipulated by special arithmetic or logical operations. The address bus links the main memory where both instruction codes and data are stored. Stack pointers, program counters, and register files also supply information to the address. Finally, there is an instruction decoder which interprets each instruction and controls all operations of the microprocessor.

Since a microprocessor is a complex sequential logic structure and not simply a few gates or an LSI memory, a true and meaningful test requires the understanding of the hardware architecture and software functionality rather than only the simple logic of the elemental structures.

The hardware architecture is the internal organization with consists of an ordered set of modules, such as the register stack, accumulator, arithmetic logic unit, etc. Software functionality is a set of ordered

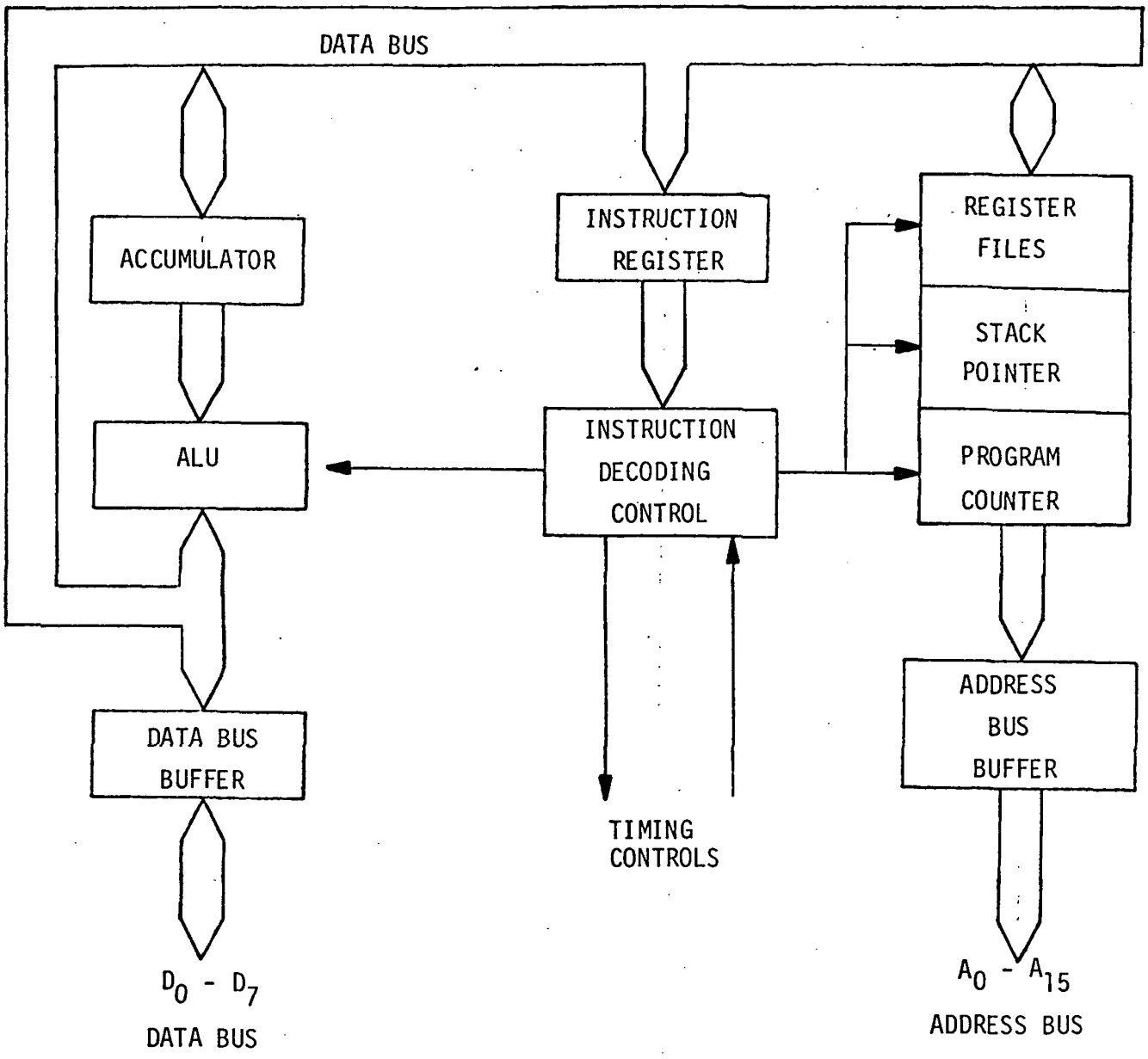


FIGURE 1: Basic MPU Block Diagram

microinstructions which can be used to monitor the operation of these modules.

Upon developing complete knowledge of the microprocessor unit through both areas, one can develop an ordered set of test sequences in the microprocessor's instruction set for testing each module one by one until a complete test has been developed.

In general, a microprocessor has two buses: an address bus and a data bus. The address bus performs two functions, addressing the external memory and/or addressing the internal scratch pad memory. The data bus also performs two functions, supplying input data to the processor and outputting processed data. The data bus links the internal functions of the scratch pad memory, registers, arithmetic logic unit, etc., together.

Modular Breakup

The next step in microprocessor testing is to partition the device into modules, with some modules possibly overlapping. The selection of each module should be accessible from the input/output bus by the execution of microinstructions. In other words, data should be able to be applied to the device input and propagated to the output directly or indirectly by the use of the microprocessor instruction set. The test then shall be generated for each module of the MPU so that a worst case test pattern will be run on that module. For instance, if the module in question is a RAM, a galloping 1's and 0's test pattern is used as this type of pattern is

considered to be worst case.

From the standpoint of software functionality, a set of MPU instructions should be executed when testing the first module. Proceeding toward the second module another set of new microprocessor instructions will be executed. (Some of these instructions may have been executed previously.) This process will then continue until all of the instructions within the instruction set are used while testing each module. Then a final test should execute all instructions to verify that all modules are working together.

Two-fold diagnostic information is provided by this technique. First, from a hardware point of view if a failure occurs, the faulty module is pinpointed. Inherent in this type of modular procedure is the fact that convenient breakpoints exist in a module-by-module basis. Second, in conjunction with each module, a set of microinstructions are executed; if any fault occurs, the specific instruction(s) can be isolated and identified.

Architecture and Test Flow

The architecture of the 2901 lends itself to the modular approach because of its own hardware and microinstruction architecture. Figure 2 illustrates the block diagram of the 2901. In examining this diagram, one will notice that the device can be divided into the following modules: RAM, Q register, arithmetic logic unit (ALU), ALU source decode multiplexer, RAM

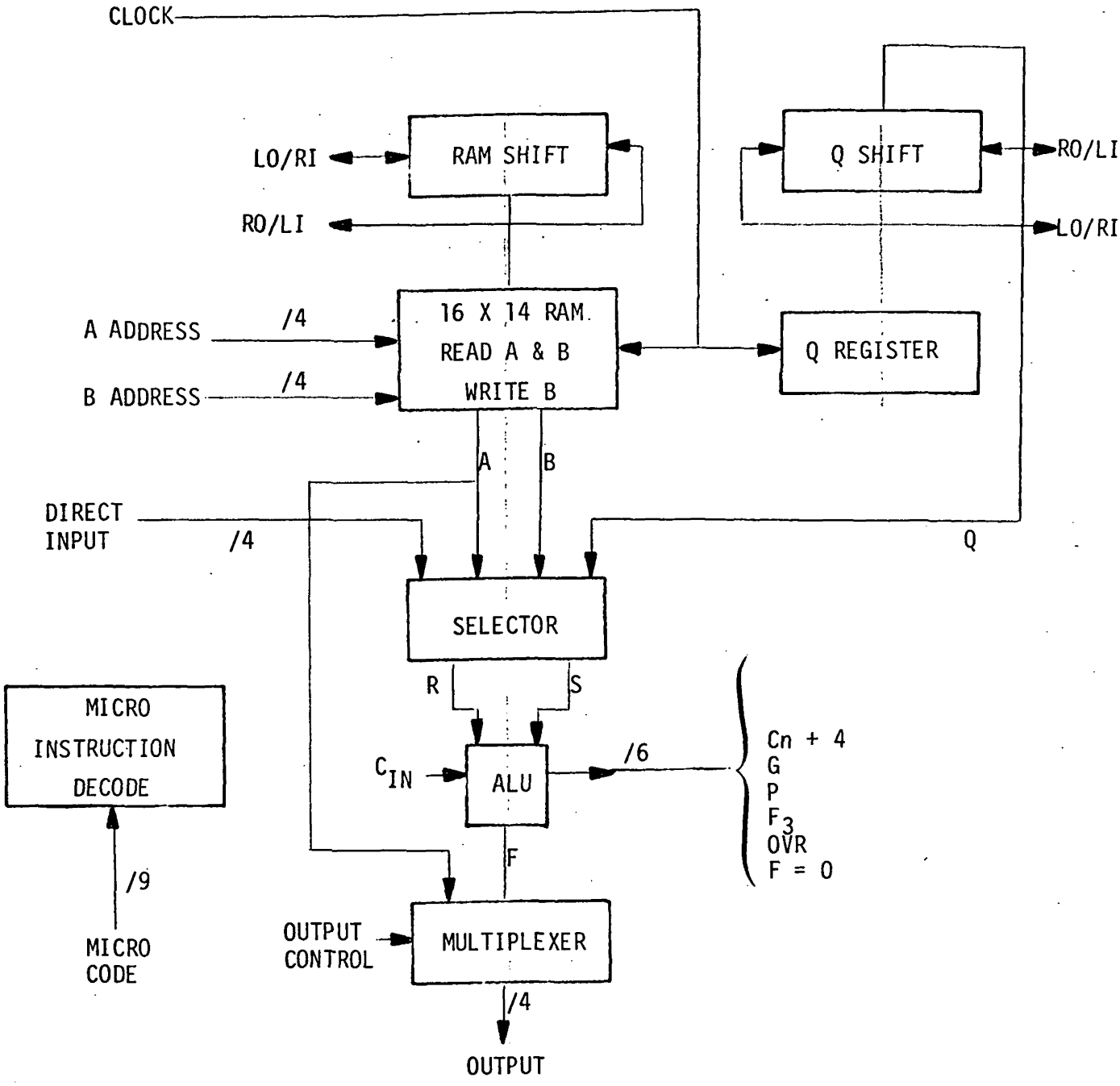


FIGURE 2: 2901 Architecture

and Q register right/left shift logic.

Once the information has been acquired on the module breakdown, a test flow can be generated. Since the 2901 has an ALU section, the first areas to be tested should be those areas which supply data to the ALU. The most logical of these is in the RAM module and then in the Q register module. Once these modules have been tested, they can be used as reliable data sources for the ALU module test.

A typical test flow for the 2901 would start with the RAM memory, followed by the Q register, ALU source decode multiplexer, ALU, and finally, the RAM and Q register right/left shift logic (Figure 3). During this test flow, all microinstructions for the 2901 will be used.

Test Technique

Formulating a test plan will differ between the manufacturer and user. The reason for this being that the manufacturer has access to the logic diagrams of the device, which the user in most cases cannot obtain, and their quantities are in larger amounts than the user's. Therefore, more elaborate tests can be developed which optimizes test performance and test time. The user has an advantage over the manufacturer because his test, in its simplest form, can be tailored to his specific needs, but the manufacturers' test has to guarantee all operations of the microprocessor. Not receiving schematics, logic diagrams, or other circuit information the user must therefore rely on either vendor supplied test programs or perform

TEST FLOW CHART	FUNCTIONAL TEST DESCRIPTION	TEST PATTERN
RAM Test ↓	A galloping "1" and "0" pattern is applied to the RAM in three combinations. 1. The RAM addressed by the "A" address and tested through the "Y" output port directly. 2. The RAM addressed by the "A" address and tested through the ALU. ALU is held at a fixed instruction. 3. The RAM addressed by the "B" address and tested through the ALU. ALU is held at a fixed instruction.	Approx. 3000
"Q" REGISTER ↓	A number 15 is loaded into the register and then read. Next, a number "0" is loaded and read. This is followed by a 14, 1, 13, 2, etc. until a "0" then a 15 is loaded.	Approx. 100
ALU Source Decode ↓	The ALU Source Decodes are tested to see if all decodes are possible. The test is performed by loading values into the RAM and "Q" register and selecting all decodes while testing for any interaction between bits or selections.	Approx. 50
ALU ↓	A series of numbers are loaded into the RAM and "A" register. These numbers are then used as inputs to the ALU. At the same time, all outputs and flags from the ALU are monitored, while incrementing operations the ALU can perform.	Approx. 1000
RAM and "Q" Register Right/Left Shift Mux.	All numbers from 0 to 15 are shifted through the RAM and "Q" register. While the RAM section is being tested, all locations are tested. After each shift, all possible number combinations are outputted to the output latch without clocking the latch, to see if there is any latch sensitivity.	Approx. 8200

FIGURE 3: 2901 Test Flow Diagram

extensive characterization to generate worst case test patterns. This characterization is needed to guarantee full operation of the microprocessor for a variety of applications in which the device is used.

The Optimum Test

At first glance of the 8080 MPU block diagram (Figure 4), the complexity of the device is not readily indicated. This is because there are only eight data input lines. However, in addition to accepting data from the input bus, the MPU can accept data from internal registers and accumulators. If the MPU could only perform one instruction, a test could be developed without much difficulty, but the MPU is capable of executing many instructions in sequence. Because of this, the number of combinations of instructions and data patterns that the MPU can perform would be extremely long.

A commonly used formula for calculating the total test time to exhaustively test an MPU is $C = 2^{MN}$. Where C is the number of combinations of instructions and data patterns, M is the number of data bits in each word, and N is the number of instructions the MPU is capable of executing.

For example, an 8-bit MPU that only has ten instructions would require 2^{80} test cycles for an exhaustive test of all possible combinations. Assuming a test cycle of 1 us, the MPU would take approximately 38 years to check all combinations of instructions and data patterns.

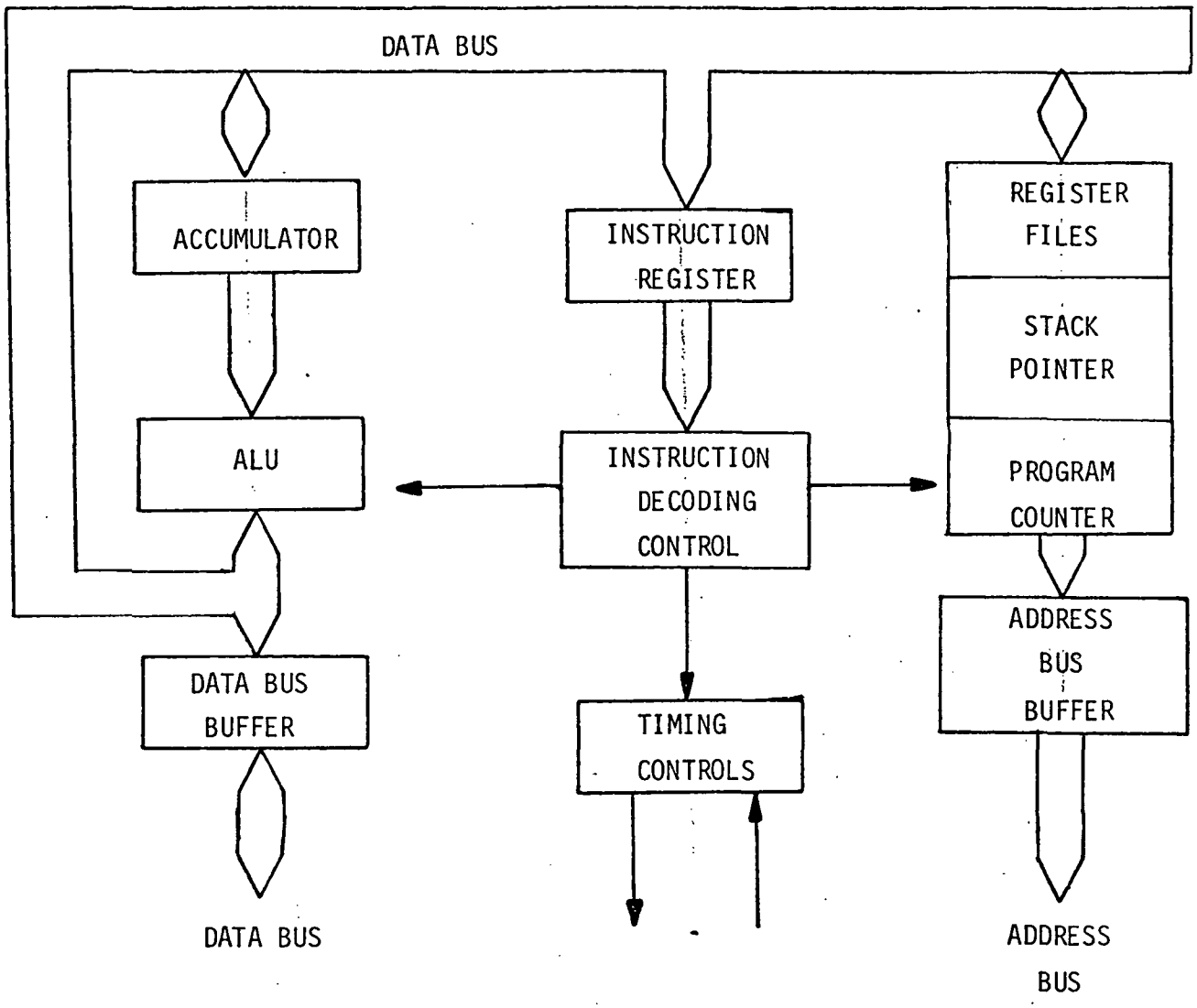


FIGURE 4: 8080 Architecture

The 8080 MPU can perform approximately 76 different instructions. Using the above formula, there would be a total of 2^{608} possible combinations that could be performed. Obviously, this is an astonishing number to exhaustively test the 8080.

Test Techniques

Once realizing that the optimum test cannot be created, one looks for other means to test the MPU. The first approach considered is called self-test. The self-test is the simplest and cheapest means of determining if an MPU is working. Self-test, or in-circuit test, is the technique in which the device is placed in the circuit where it will be used and tested for correct operation. This is utilized by some users who feel the cost of incoming inspection cannot be justified. Therefore, they will typically test the device using several different system operations. The advantage of this testing is that the actual operation of the device is tested in its circuit, eliminating the requirement for a separate costly test system. The disadvantages of this technique is that any of the in-circuit condition changes, like voltage fluctations, temperature, timing, and instruction changes, may not be detected until the unit is in the field. The rework cost of finding and removing a faulty device must be considered before this method of testing is selected. Typical costs for finding and replacing a gate is as follows:

$3.00 to $5.00: Board Level

$30.00 to $50.00: System Level

$300.00 or Move: In Field

Since an MPU is more complex than a gate, the above cost would be multiplied by the complexity factor of the MPU.

The second method of testing is called comparison testing. Comparison testing is the method in which a known good device is compared to the device under test. The hardware required for this type of test is very simple, requiring only a pseudo number generator connected to all inputs and all outputs from the known good device and comparing the device under test (Figure 5). If exact comparison does not occur, the device under test is considered bad. The advantages of this method is that the test system is inexpensive to develop and with a little more hardware added, voltage and timing conditions can be created. Also, if the device is operated for a few minutes, most paths through the device will be checked. Like any test method, it has its disadvantages also. The biggest disadvantage is that this method requires a known good device, which is a problem in itself. Some MPU's have illegal instructions, therefore, no guarantees can be made for the data coming out of the device. Also, critical timing into the device may not be able to be maintained if pseudo numbers are applied to the input of the MPU. Last of all, if the device fails, no failure information can be obtained to determine the cause.

The next method of testing is the stored pattern method which utilizes a known good pattern stored in some form of data memory. This pattern is then applied to the device under test in a burst mode and the device outputs compared to the stored response (Figure 6). There are two means of generating patterns using this method. The first method is to input a test pattern into a known good device and record all input stimuli and output data.

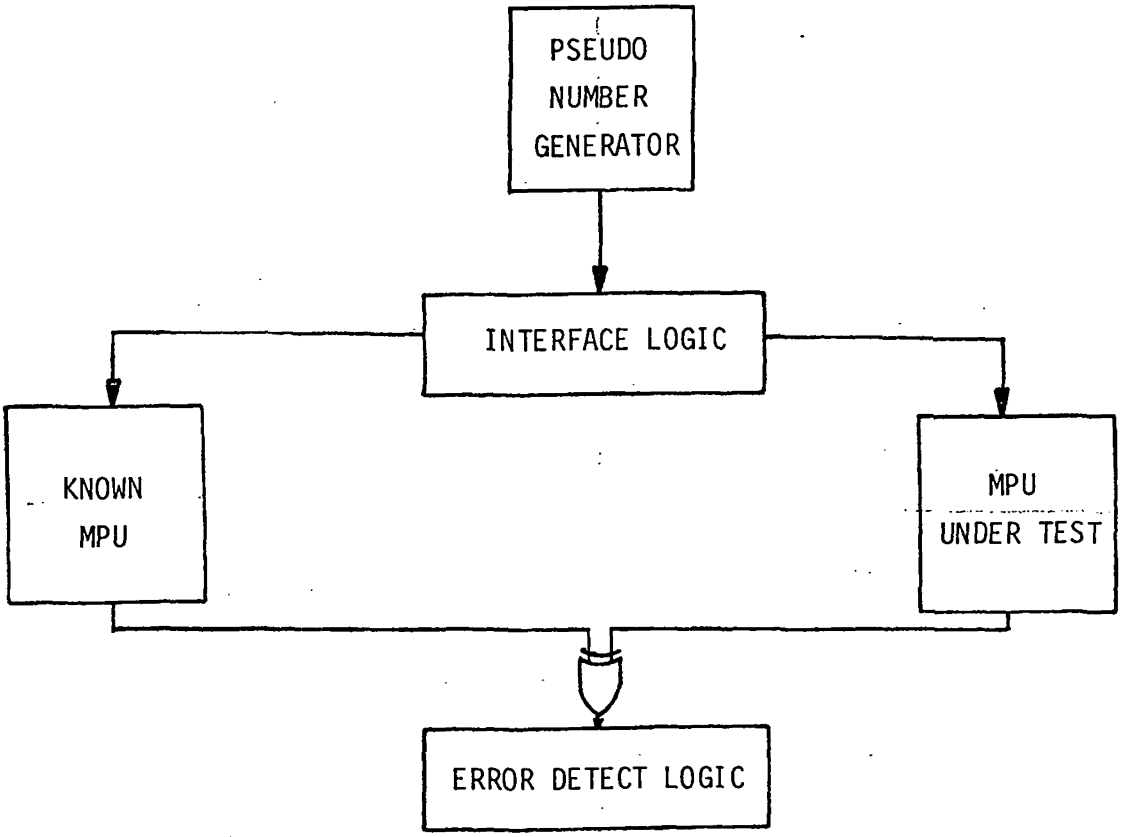


FIGURE 5: Comparison Test

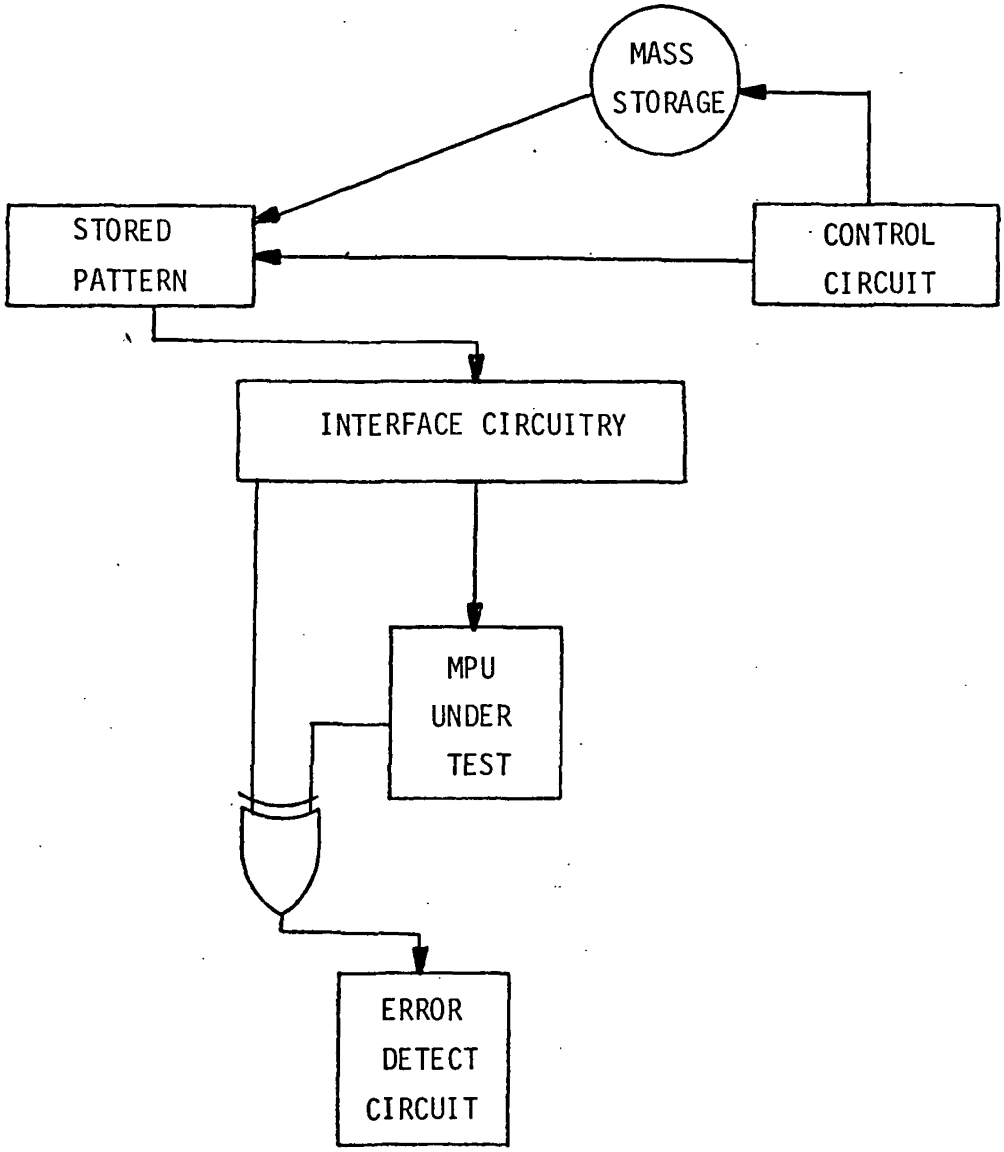


FIGURE 6: Stored Pattern Test

The input patterns would be created from some known application.

The second method of generating the stored pattern would be to develop a software or hardware simulator for the device to be tested. A known instruction sequence would then be stored and used to compare with the device under test. The advantage of this technique is that the user's instruction sequence can be completely tested and that sensitive data paths can be checked with ease. Since the tester that is required to perform this type of test usually incorporates variable voltage and timing circuits, these parameters can also be checked. The main disadvantage of using a known good device for generating the test pattern is a "known good device." What test is available to determine what is a known good device? The disadvantage of the stimulator approach is that a software or hardware similator is required. Since the schematic and logic diagrams for each MPU are not readily available from the vendor, it is difficult for a user to develop the simulator. Even if these could be obtained, it would take a knowledgeable programmer three to six months, at least, to develop the software. Other disadvantages to this method are:

> LARGE, EXPENSIVE MEMORY. High-speed random access memories or shift registers become quite expensive when any great amount of memory is needed. In testing the program counter for the 8080, for example, 262,000 distinct patterns are required. A memory test on the register array of an 8080 takes approximately 50,000 patterns. The cost of memory can quickly become a major part of the total cost of the test system.

LONG TRANSFER TIME. The overhead time required to transfer a long pattern from disc, core, or other mass memory to high-speed RAM can make a large dent in the throughput rate of the test system. If transferring a 1,024-bit pattern from disc to RAM takes 50 milliseconds, a typical figure, transferring the test pattern from the program counter takes 13.1 seconds of overhead time, in addition, to the test execution time ($262 \times 50 \times 10^{-3}$ Seconds).

INFLEXIBLE PROGRAM. The stored program cannot easily be modified while tests are in progress. This rigidity makes it difficult to perform special or unusual tests on a single unit. A substantial amount of off-line software support is needed if such tests are to be accomplished.

The algorithmic test method utilizes a high-speed programmable pattern generator in conjunction with a local buffer memory. The contents of the buffer memory is a test pattern consisting of microprocessor instruction sequences and either full or partial data input and output response patterns. The buffer memory pattern is then applied to the microprocessor under program control of the pattern generator. A distinct advantage of this test method offered by the use of a programmable pattern generator is the ability to choose how the test pattern is applied to the device under test. This will in turn determine whether the stored data pattern and output response of the microprocessor is full or partial.

The first option is to apply the test pattern in a burst mode as in the previously defined stored response approach. In this case, the device data pattern and output response stored in the buffer memory is complete, with the pattern generator acting as a counter to advance the test pattern vectors.

In the second mode, special algorithms are written for the pattern generator which simulate the microprocessor instruction execution. These special algorithms input microprocessor instruction codes and data patterns at the proper point in the instruction cycle, and compare the device output accordingly. However, the device data pattern and output response may be partially stored in the buffer memory and partially generated in real time by the pattern generator algorithms. The effect is to enhance the MPU test program by allowing a significant increase in the number of test patterns used, enable additional tests to be performed that would be difficult, if possible at all by any of the previous methods, and reduce the total amount of stored test vectors. A disadvantage here is that in addition to the buffer test pattern required, a separate program for the pattern generator may be necessary which increases the complexity of the total effort.

This technique, which eliminates the delay time in transferring patterns to mass memory, is extremely efficient and flexible in generating patterns for logic modules such as binary counters, random access and read only memories, shift registers, as well as microprocessors.

When used in conjunction with the module approach, algorithm pattern generation permits faults to be diagnosed so that the particular module or instruction which caused a failure can be isolated. The disadvantages of this method is that a sophisticated tester is required. The programmer needs to be knowledgeable of both operation of the MPU and the test system itself to develop the program.

The recommended approach to be described is a combination of stored pattern and algorithmic techniques. This approach was selected because of its ease of program development (stored pattern) and its thorough testing ability (algorithmic).

II. BASIC BLOCK DIAGRAM

As shown in Figure 4, the basic microprocessor unit includes a data and address bus, accumulator, arithmetic logic unit, register files, stack pointer, program counter, and timing controls. In the following figures, Figures 7 through 11, the 8080, 8008, 2901, 6800, and 1802, block diagrams are illustrated.

8080

Using the 8080 (Figure 7) as a reference, all other MPU's are structured very similar. Other than their instruction set, they differ as described below.

8008

The 8008 (Figure 8) is very similar in architecture to the 8080. The basic difference is that the 8008 has seven 14-bit stack registers for storage of return addresses as a result of subroutine calls. The 8080 has one 16-bit pointer for controlling an external memory stack allowing more than seven levels of subroutine testing.

2901

The 2901 (Figure 9) differs the most from the 8080. The 2901 is only the process portion of a basic MPU, a 4-bit processor, which lacks any

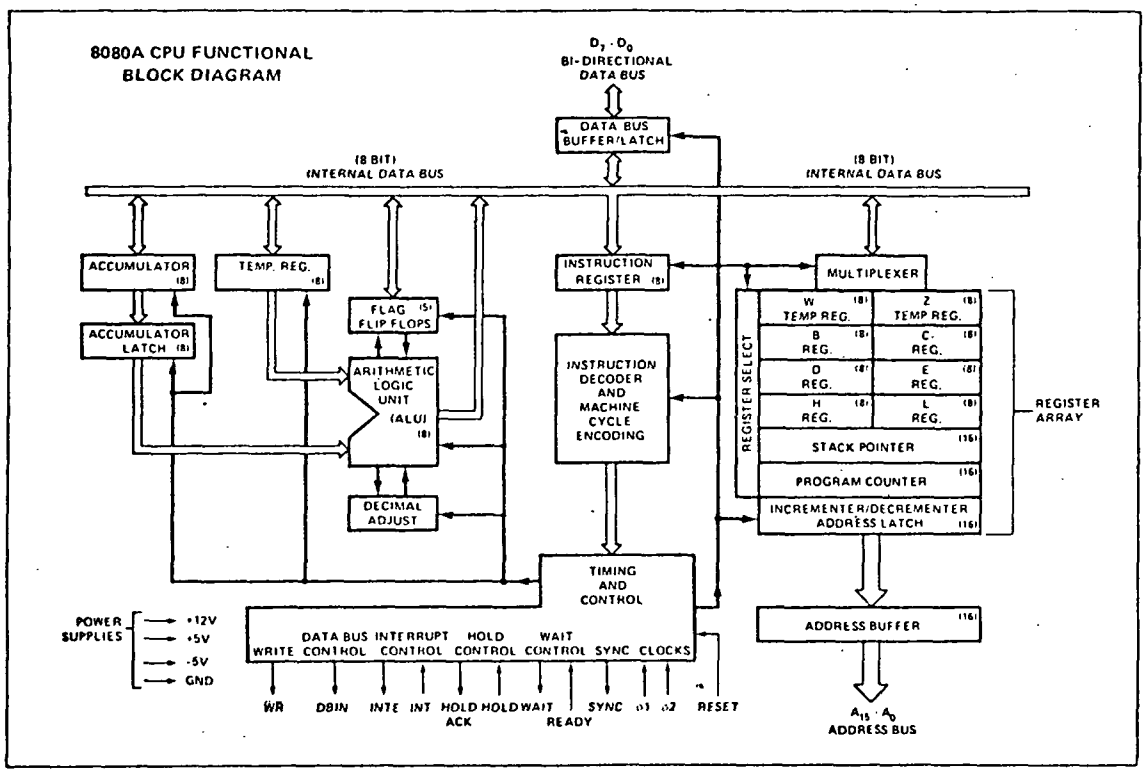


FIGURE 7: 8080 Block Diagram

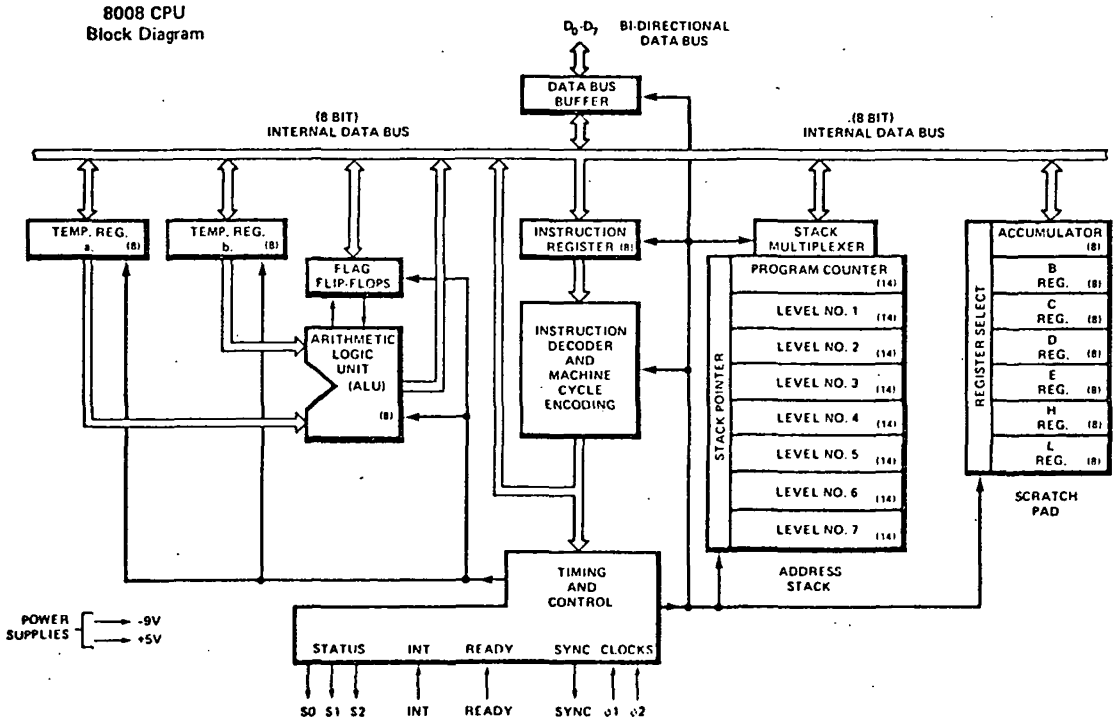


FIGURE 8: 8008 Block Diagram

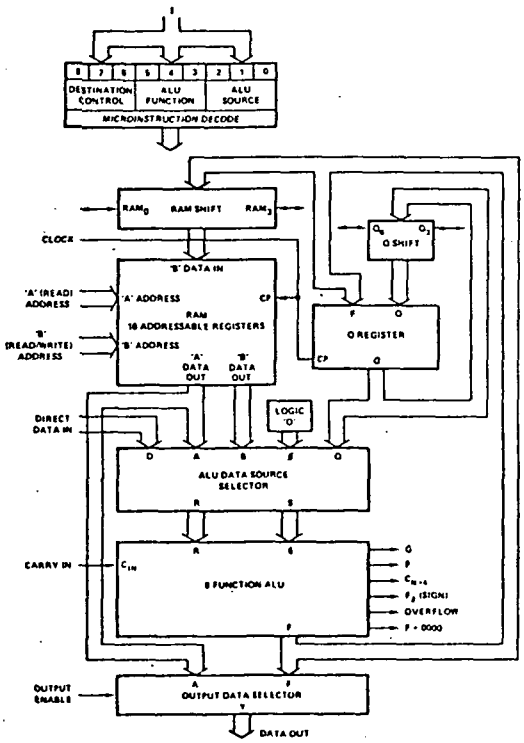


FIGURE 9: 2901 Block Diagram

ordered instruction set. Therefore, the 2901 does not have an instruction decoder. The 2901 does have a register array (16 words X 4-bits), an accumulator (4-bits), and an arithmetic logic unit (ALU). The 2901 does not have a program counter to control from which memory location the next instruction will be fetched. This is controlled by external circuitry. Last of all, the 2901 cannot execute a jump or subroutine call by itself; thus, it also lacks a stack pointer.

6800

The 6800 (Figure 10) is structured similar to the 8080 but does not contain a register array. External RAM is used for all scratch pad operations. Also, the 6800 includes two accumulators as opposed to one provided by the 8080.

1802

The 1802 (Figure 11) architecture is similar to the 8080 except that the program counter and stack pointer are included as part of the register array. Also, instead of having a 16-bit address bus it has an 8-bit bus, which multiplexes the address in 8-bit bytes.

EXPANDED BLOCK DIAGRAM

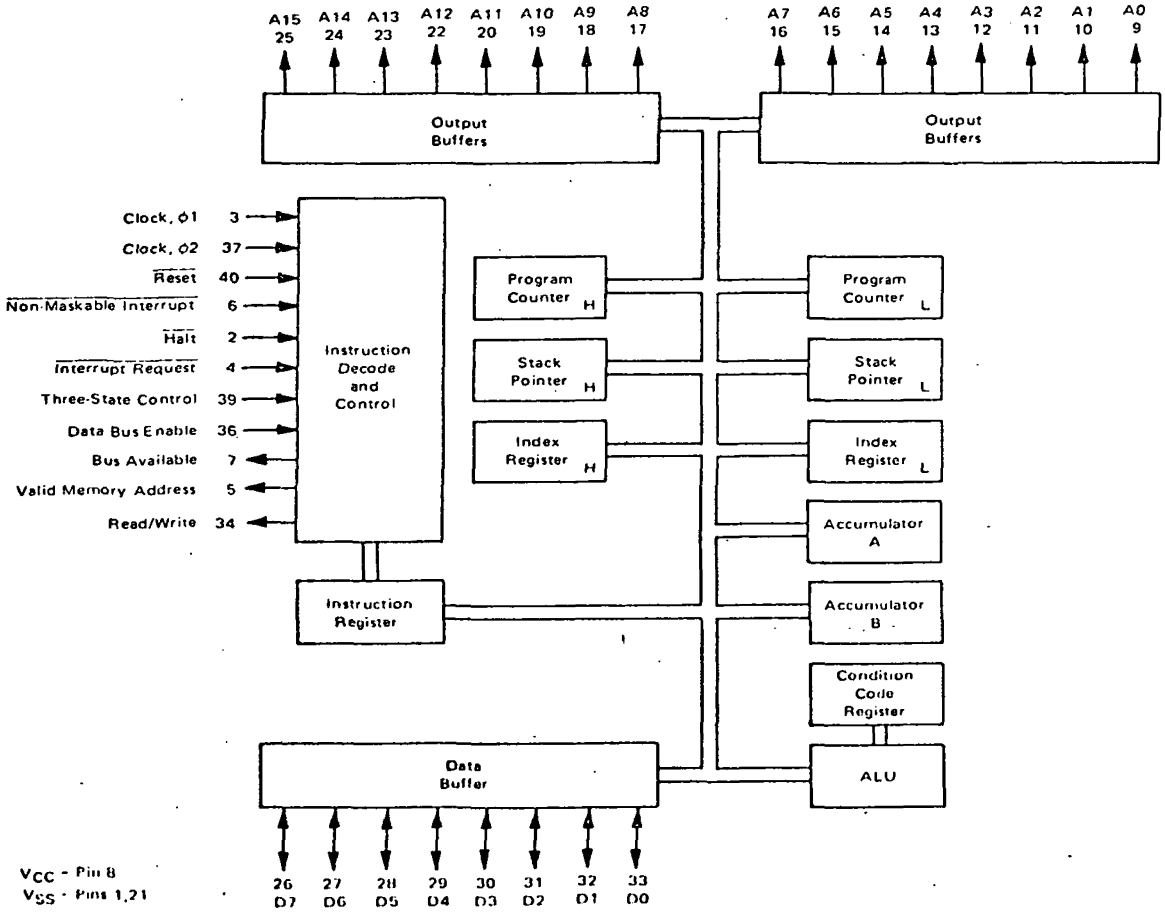


FIGURE 10: 6800 Block Diagram

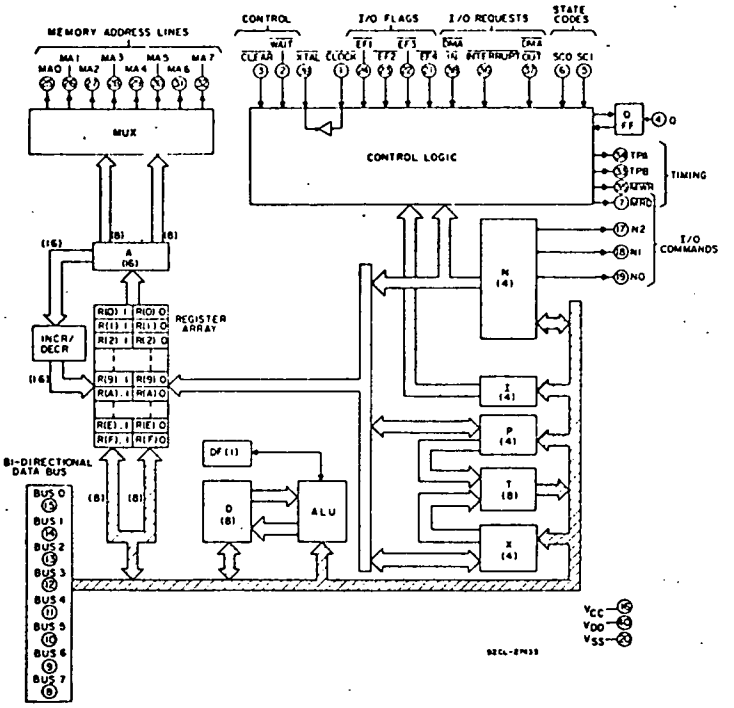


Figure 11: 1802 Block Diagram

III. MODULAR APPROACH

As previously shown, all microprocessor units have a similar architecture from which a basic test philosophy can be adopted. This philosophy is to develop an approach to test each module separately accomplishing the following goals:

A. Verify the functionality of each module within the device using the input/output pins of the device and its instruction set.

B. Test for destructive interaction between functional modules.

C. Verify all timing, status information, and interrupt operations of the device.

IV. DESCRIPTIONS OF THE MODULAR TEST APPROACH

Since each MPU is structured around a similar architecture, a common test approach can be adopted and applied to each device. Once this approach has been established, further requirements are to implement the approach according to the specific architecture and instruction set of each device. The following is a basic description of a generalized test approach for each module previously described.

A. Program Counter

1. Verify reset state.
2. Verify that the counter can be incremented through its maximum range.
3. Check any possible register transfer to the program counter.

B. Register Arrays

1. Verify that each register can be loaded individually, if possible, and its contents stored to the data bus.
2. Verify register-to-register and register-to-output transfers with all possible number combinations.
3. If the registers can be incremented and/or decremented, verify that they can accomplish this through their complete range.

C. Stack Pointer

1. Verify that the stack pointer (registers) can be loaded.
2. Check to see if stack pointer transfers are valid.
3. Verify increment and decrement operations.

D. Arithmetic Logic Unit

1. Verify ADD operations, with and without carry.
2. Verify a SUBTRACT operation, with and without a borrow.
3. Verify all shift left or shift right operations.
4. Verify rotation of a numerical value, if applicable.
5. Check all logical operations, for example, AND, OR, EOR, etc., when applicable.

E. Accumulator

1. Test to see if it can be loaded and read.
2. Check for any transfer operation that can be performed.
3. Verify that the accumulator can be incremented and decremented.

F. Timing and Control

1. Verify that all control timing occurs at correct reference points, for example, data bus enable, sync signals, write enables, etc.

2. Exercise all control operations on the device to verify operation, for example, WAIT, HOLD, INTERRUPT, etc.
3. Verify any status flags that are produced during an arithmetic operation, such as carries, negative or positive numbers, overflows, etc.

G. Instruction Decodes

1. Verify full operation by execution of the complete instruction set.
2. Verify execution of branch and jump operations.
3. Test for interaction between all modules, and verification of all data paths between modules.

V. PROCESSOR TEST DESCRIPTIONS

A. 8080

The 8080 is an 8-bit microprocessor using an N-channel silicon gate MOS process. The 8080 can be divided into the following modules based on its functional block diagram (see Figure 7).

Functional Module Breakup

1. Timing and Control
2. Instruction Decoder
3. Program Counter
4. Register Array
5. Stack Pointer
6. Accumulator
7. Arithmetic Logic Unit (ALU)

Due to the complexity of some tests on the modules, a flow chart of the recommended test will be used to ease the burden of understanding the test.

Timing and Control Test

The first test on the 8080 is to verify the operation of all timing and control signals. This test was selected first because the basic operation of the MPU requires that timing and control be present.

TEST 1, RESET: Verify that the Hold Acknowledge (HLDA) appears following the rising edge of clock Ø1 and that the Data and Address buses go into a tristate condition following the rising edge of clock Ø2. Verify that the Interrupt Enable (INTE) is reset. Last, following the removal of the reset, the Program Counter is equal to 0, which will appear on the address bus. When performing a reset note that the reset signal should be present for at least four clock periods.

TEST 2, TIMING: Execute a NOP instruction following a reset, verify that the SYNC signal occurs within the first clock cycle, that the DBIN signal occurs in the second clock cycle, and finally, that the Program Counter increments and that it is present on the address bus during the fourth clock cycle. Follow this NOP instruction with a Store Accumulator (STA) direct instruction and verify that the Write ($\overline{WR}$) goes low during the third clock cycle of that instruction.

TEST 3, HOLD: Present a Hold signal to the 8080 and verify that during T2 time cycle Hold Acknowledge (HLDA) appears and the Address and Data buses go to tristate. Upon removing the Hold signal, verify that HLDA is removed, and the buses are enabled. During the time that the Hold signal is present, the 8080 should be in a Hold operation for the time

that the Hold signal is present.

TEST 4, INT: Execute an Enable Interrupts (INTE) instruction, followed by a few NOP instructions, and present an Interrupt Request (INT) to the 8080 during an NOP instruction cycle. Verify that the Interrupt Enable is present during T1 time of the next instruction. This INTE signal should not go high until T1 time. Upon presenting a reset signal to the 8080, verify that the INTE signal is removed.

Instruction Decoder Test

The next test on the 8080 should check the Instruction Decoder. This test is used to verify that the complete device is operational and that it will execute all instructions in the instruction set. This recommended test is designed to test all instructions but not all data patterns. Table 1 is a listing of the recommended instruction sequence.

Program Counter Test

This test includes a reset, which clears the Program Counter, and 2^{16} NOP instructions or any other instruction(s) to verify that the counter will increment through all possible addresses. A flow chart of this test is illustrated in Figure 12. This test will verify that the Program Counter resets and increments. The only operation

```
NOP                 *-----------------------------*
EI                  * LOCATIONS 0-15 ARE FOR THE  *
DI                  * HALT-HOLD-INTERUPT ROUTINE  *
EI                  *-----------------------------*
LXI SP
 (VERIFIES HIGH IMPEDANCE DURING HOLD & HALT)
 SPL=02   SP=0102
 SPH=01
RST (AT 0038)
 PCH(00) TO (SP-1)
 PCL(07) TO (SP-2)  SP-2=0100
EI
HALT
RST (AT 0000)
 PCH(00) TO (SP-1)
 PCL(3A) TO (SP-2)  SP-2=00FE
LXI B
 C=02
 B=01               *-----------------------------*
LXI D               * MAIN INSTRUCTION SEQUENCE  *
 E=08               * STARTS AT LOCATION 16       *
 D=04               *-----------------------------*
LXI H
 L=20
 H=10
LXI SP
```

TABLE 1: Recommended Instruction Sequence

TABLE 1 Continued

```
NOP
 SPL=FE  SP=00FE
 SPH=00
LDA  (B3B2)TO A
 B2
 B3
 (00FF) TO A   A=40
STA  A TO (B3B2)
 B2
 B3
 40 TO FFFF
POP PSW
 (SP) TO F     F=06
 (SP+1) TO A   A=80      SP+2=0100
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)             SP-2=00FE
PUSH B
 B TO (SP-1)
 C TO (SP-2)             SP-2=00FC
PUSH D
 D TO (SP-1)
 E TO (SP-2)             SP-2=00FA
PUSH H
 H TO (SP-1)
 L TO (SP-2)   SP-2=00F8
```

TABLE 1 Continued

```
NOP

POP B

 (SP) TO C      C=04

 (SP+1) TO B    B=02        SP+2=00FA

POP D

 (SP) TO E      E=10

 (SP+1) TO D    D=08        SP+2=00FC

POP H

 (SP) TO L      L=40

 (SP+1) TO H    H=20        SP+2=00FE

MOV M,A

 A TO (HL)

MOV M,B

 B TO (HL)

MOV M,C

 C TO (HL)

MOV M,D

 D TO (HL)

MOV M,E

 E TO (HL)

MOV M,H

 H TO (HL)

MOV M,L

 L TO (HL)

XCHG  D=20,H=08,E=40,L=10

MOV M,D
```

TABLE 1 Continued

```
NOP
 D TO (HL)
MØV M,E
 E TO (HL)
MØV M,H
 H TO (HL)
MØV M,L
 L TO (HL)
XTHL
 (SP) TO L      L=FE
 (SP+1) TO H    H=ØØ
 OLD H TO (SP+1)
 OLD L TO (SP)
MØV M,H
 H TO (HL)
MØV M,L
 L TO (HL)
PCHL   PC=(HL)              PC=ØØFE
SPHL   SP=(HL)              SP=ØØFE
DAD SP  HL=HL+SP            HL=Ø1ØØ
PUSH H
 H TO (SP-1)
 L TO (SP-2)                SP-2=ØØFC
MØV M,H
 H TO (HL)
MØV M,L
```

TABLE 1 Continued

```
NOP
 L TØ (HL)
DAD B    HL=HL+BC   HL=01FC+0204=0400
MØV M,H
 H TØ (HL)
MØV M,L
 L TØ (HL)
PAD D    HL=HL+DE   HL=0400+2040=2440
MØV M,H
 H TØ (HL)
MØV M,L
 L TØ (HL)
DAD H    HL=HL+HL   HL=2440+2440=4880
MØV M,H
 H TØ (HL)
MØV M,L
 L TO (HL)
STAX B
 A TO (BC)
STAX D
 A TØ (DE)
LDAX B
 ØØ TØ A FROM (BC)
MOV M,A
 A TØ (HL)
LDAX D
```

TABLE 1 Continued

```
NOP

 FF TØ A FROM (DE)

MØV M,A

 A TO (HL)

1NX B  BC+1=0205

1NX D  DE+1=2041

1NX H  HL+1=4881

1NX SP SP+1=FD

PUSH H

 (H) TØ (SP-1)

 (L) TO (SP-2)  SP-2=FB

MØV M,B

 B TØ (HL)

MØV M,C

 C TØ (HL)

MØV M,D

 D TØ (HL)

MØV M,E

 E TØ (HL)

MØV M,H

 H TØ (HL)

MØV M,L

 L TØ (HL)

DCX B  BC-1=0204

DCX D  DE-1=2040

DCX H  HL-1=4880
```

TABLE 1 Continued

```
NOP
DCX SP SP-1=FA
PUSH H
 H TO (SP-1)
 L TO (SP-2) SP-2=F8
MOV M,B
 B TO (HL)
MOV M,C
 C TO (HL)
MOV M,D
 D TO (HL)
MOV M,E
 E TO (HL)
MOV M,H
 H TO (HL)
MOV M,L
 L TO (HL)
CMA   COMPLEMENT A (=00)
STC   SET CARRY =1
PUSH PSW
 A TO (SP-1)
 F TO (SP-2) SP-2=F6
CMC   COMP. CARRY (=0)
IN
 B2   DEV=0F
 9D TO A FROM 0F0F
```

TABLE 1 Continued

```
NOP

PUSH PSW

 A TØ (SP-1)

 F TØ (SP-2) SP-2=F4

DAA A TO BCD  A=Ø3,SET FØ,F4

PUSH PSW

 A TØ (SP-1)

 F TØ (SP-2) SP-2=F2

DAA   A TØ BCD  A=69 CLEAR F4

PUSH PSW

 A TØ (SP-1)

 F TØ (SP-1) SP-2=FØ

SHLD

 B2

 B3

 L TØ (B3B2)

 H TØ (B3B2+1)

LHLD

 B2

 B3

 ØØ TØ L FRØM B3B2

 FF TØ H FRØM B3B2+1

MØV M,H

 H TO (HL)

MØV M,L

 L TØ (HL)
```

TABLE 1 Continued

```
NOP
INR M
 FF FROM (HL) +1=00
 00 TO (HL)
DCR M
 00 FROM (HL) -1=FF
 FF TO (HL)
NOP
OUT
 B2  DEV=AA
 69 TO DEV AAAA FROM A
MOV A,M
 01 TO A FROM (HL)
MOV B,M
 08 TO B FROM (HL)
MOV C,M
 04 TO C FROM (HL)
MOV D,M
 08 TO D FROM (HL)
MOV E,M
 10 TO E FROM (HL)
MOV H,M
 20 TO H FROM (HL)
MOV L,M
 40 TO L FROM (HL)
MOV M,A
```

TABLE 1 Continued

```
NOP
 A TØ (HL)
MØV M,B
 B TØ (HL)
MØV M,C
 C TØ (HL)
MØV M,D
 D TØ (HL)
MØV M,E
 E TØ (HL)
MØV M,H
 H TØ (HL)
MØV M,L
 L TØ (HL)
MVI A
 B2  Ø2 TØ A
MVI B
 B2  Ø4 TØ B
MVI C
 B2  Ø8 TØ C
MVI D
 B2  1Ø TØ D
MVI E
 B2  2Ø TØ E
MVI H
 B2  4Ø TØ H
NOP
MVI L
 B2  8Ø TØ L
MØV M,A
 A TØ (HL)
MØV M,B
 B TØ (HL)
MØV M,C
 C TØ (HL)
MØV M,D
 D TØ (HL)
MØV M,E
 E TØ (HL)
MØV M,H
 H TØ (HL)
MØV M,L
 L TØ (HL)
MVI M
 B2
 FF TØ (HL)
INR A    A+1=Ø3
MØV M,A
 A TØ (HL)
INR B    B+1=Ø5
MØV M,B
 B TØ (HL)
```

TABLE 1 Continued

```
NOP
INR C    C+1=9
MOV M,C
 C TO (HL)
INR D    D+1=11
MOV M,D
 D TO (HL)
INR E    E+1=21
MOV M,E
 E TO (HL)
INR H    H+1=41
MOV M,H
 H TO (HL)
INR L    L+1=81
MOV M,L
 L TO (HL)
DCR A    A-1=02
MOV M,A
 A TO (HL)
DCR B    B-1=04
MOV M,B
 B TO (HL)
DCR C    C-1=08
MOV M,C
 C TO (HL)
DCR D    D-1=10
```

```
NOP
MOV M,D
 D TO (HL)
DCR E    E-1=20
MOV M,E
 E TO (HL)
DCR H    H-1=40
MOV M,H
 H TO (HL)
DCR L    L-1=80
MOV M,L
 L TO (HL)
MOV A,B  A=04
MOV B,C  B=08
MOV C,D  C=10
MOV D,E  D=20
MOV E,H  E=40
MOV H,L  H=80
MOV L,A  L=04
MOV M,A
 A TO (HL)
MOV M,B
 B TO (HL)
MOV M,C
 C TO (HL)
MOV M,D
```

TABLE 1 Continued

```
NOP
 D TO (HL)
MOV M,E
 E TO (HL)
MOV M,H
 H TO (HL)
MOV M,L
 L TO (HL)
MOV A,C    A=10
MOV B,D    B=20
MOV C,E    C=40
MOV D,H    D=80
MOV E,L    E=04
MOV M,A
 A TO (HL)
MOV M,B
 B TO (HL)
MOV M,C
 C TO (HL)
MOV M,D
 D TO (HL)
MOV M,E
 E TO (HL)
MOV H,A    H=10
MOV L,B    L=20
MOV M,H
```

```
NOP
 H TO (HL)
MOV M,L
 L TO (HL)
MOV A,D    A=80
MOV B,E    B=04
MOV C,H    C=10
MOV D,L    D=20
MOV M,A
 A TO (HL)
MOV M,B
 B TO (HL)
MOV M,C
 C TO (HL)
MOV M,D
 D TO (HL)
MOV E,A    E=80
MOV H,B    H=04
MOV L,C    L=10
MOV M,E
 E TO (HL)
MOV M,H
 H TO (HL)
MOV M,1
 L TO (HL)
MOV E,M
```

TABLE 1 Continued

```
NOP
 (HL) TO E    E=40
MOV H,M
 (HL) TO H    H=08
MOV L,M
 (HL) TO L    L=02
MOV A,E      A=40
MOV E,B      E=04
MOV B,H      B=08
MOV H,C      H=10
MOV C,L      C=02
MOV L,D      L=20
MOV D,A      D=40
MOV M,A
 A TO (HL)
MOV M,B
 B TO (HL)
MOV M,C
 C TO (HL)
MOV M,D
 D TO (HL)
MOV M,E
 E TO (HL)
MOV M,H
 H TO (HL)
MOV M,L
NOP
 L TO (HL)
MOV A,H      A=10
MOV H,D      H=40
MOV D,B      D=08
MOV B,L      B=20
MOV L,E      L=04
MOV E,C      E=02
MOV C,A      C=10
MOV M,A
 A TO (HL)
MOV M,B
 B TO (HL)
MOV M,C
 C TO (HL)
MOV M,D
 D TO (HL)
MOV M,E
 E TO (HL)
MOV M,H
 H TO (HL)
MOV M,L
 L TO (HL)
MOV A,L      A=04
MOV L,H      L=40
MOV H,E      H=02
```

TABLE 1 Continued

```
NOP
MOV E,D    E=08
MOV D,C    D=10
MOV C,B    C=20
MOV B,A    B=04
MOV M,A
 A TO (HL)
MOV M,B
 B TO (HL)
MOV M,C
 C TO (HL)
MOV M,D
 D TO (HL)
MOV M,E
 E TO (HL)
MOV M,H
 H TO (HL)
MOV M,L
 L TO (HL)
POP PSW          SP=00F0 (SEE LINE 186)
 (SP)    TO F    F=02
 (SP+1) TO A     A=80       SP+2=00F2
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00F0
ADD B  A=A+B=80+04=84    F=86    A=84
```

TABLE 1 Continued

```
NOP
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00EE
ADD C  A=A+C    A=A4    F=82
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00EC
ADD D  A=A+D    A=B4    F=86
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00EA
ADD E  A=A+E    A=BC    F=82
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00E8
ADD H  A=A+H    A=BE    F=86
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00E6
ADD L  A=A+L    A=FE    F=82
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00E4
ADD M  A+(HL)
 (HL) IN=01     A=FF    F=86
```

TABLE 1 Continued

```
NOP
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00E2
ADD A   A=A+A    A=FE    F=93
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00E0
ADC B   A=A+B+1  A=03    F=17
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00DE
ADC C   A=A+C+1  A=24    F=06
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00DC
ADC D   A=A+D    A=34    F=02
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00DA
ADC E   A=A+E    A=3C    F=06
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00D8
ADC H   A=A+H    A=3E    F=02
PUSH PSW
```

TABLE 1 Continued

```
NOP
 A TO (SP-1)                                  A=3E
 F TO (SP-2)    SP-2=00D6                     B=04
ADC M   A=A+(HL) A=00   F=57                  C=20
 (HL) IN=C2                                   D=10
PUSH PSW                                      E=08
 A TO (SP-1)                                  H=02
 F TO (SP-2)    SP-2=00D4                     L=40
ADC L   A=A+L+1 A=41    F=06
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00D2
ADC A   A=A+A   A=82    F=86
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00D0
SUB B   A=A-B   A=7E    F=06
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00CE
SUB C   A=A-C   A=5E    F=12
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00CC
SUB D   A=A-D   A=4E    F=16
PUSH PSW
```

TABLE 1 Continued

```
NOP
 A TO (SP-1)
 F TO (SP-2)    SP-2=00CA
SUB E  A=A-E    A=46    F=12
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00C8
SUB H  A=A-H    A=44    F=16
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00C6
SUB L  A=A-L    A=04    F=12
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00C4
SUB A  A=A-A    A=00    F=56
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00C2
SUB M  A=A-(HL)
 (HL) IN=FF     A=01    F=03
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00C0
SBB B  A=A-B-1  A=FC    F=87
PUSH PSW
```

```
NOP
 A TO (SP-1)
 F TO (SP-2)    SP-2=00BE
SBB C   A=A-C-1 A=DB    F=96
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00BC
SBB D   A=A-D   A=CB    F=92
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00BA
SBB E   A=A-E   A=C3    F=96
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00B8
SBB H   A=A-H   A=C1    F=92
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00B6
SBB L   A=A-L   A=81    F=96
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00B4
SBB M   A=A-(HL)
 (HL) IN=82     A=FF    F=87
PUSH PSW
```

TABLE 1 Continued

```
NOP
 A TO (SP-1)
 F TO (SP-2)    SP-2=00B2
SBB A   A=A-A-1 A=FF    F=87
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00B0
ADI   A=A+(B2)
 B2=01          A=00    F=57
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00AE
ACI   A=A+(B2)+1
 B2=FF          A=00    F=57
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00AC
SUI   A=A-(B2)
 B2=01          A=FF    F=87
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=00AA
SBI   A=A-(B2)-1
 B2=40          A=BE    F=96
PUSH PSW
 A TO (SP-1)
```

TABLE 1 Continued

NOP			
F TO (SP-2)	SP-2=00A8		
ORA B A IOR B	A=BE	F=86	A=BE
PUSH PSW			B=04
A TO (SP-1)			C=20
F TO (SP-2)	SP-2=00A6		D=10
XRA B A XOR B	A=BA	F=82	E=08
PUSH PSW			H=02
A TO (SP-1)			L=40
F TO (SP-2)	SP-2=00A4		
ORA C A IOR C	A=BA	F=82	
PUSH PSW			
A TO (SP-1)			
F TO (SP-2)	SP-2=00A2		
XRA C A XOR C	A=9A	F=86	
PUSH PSW			
A TO (SP-1)			
F TO (SP-2)	SP-2=00A0		
ORA D A IOR D	A=9A	F=86	
PUSH PSW			
A TO (SP-1)			
F TO (SP-2)	SP-2=009E		
XRA D A XOR D	A=8A	F=82	
PUSH PSW			
A TO (SP-1)			
F TO (SP-2)	SP-2=009C		

TABLE 1 Continued

```
NOP
ORA E   A IOR E   A=8A       F=82
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=009A
XRA E   A XOR E   A=82       F=86
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0098
ORA H   A IOR H   A=82       F=86
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0096
XRA H   A XOR H   A=80       F=82
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0094
ORA L   A IOR L   A=C0       F=86
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0092
XRA L   A XOR L   A=80       F=82
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0090
ORA A   A IOR A   A=80       F=82
```

TABLE 1 Continued

```
NOP
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=008E
XRA A   A XOR A   A=00        F=46
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=008C
ORA M   A IOR (HL)
 (HL)IN=BE        A=BE        F=86
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=008A
XRA M   A XOR (HL)
 (HL)IN=78        A=C6        F=86
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0088
ANA M   A=A AND (HL)
 (HL)IN=FC                  A=C4        F=92
PUSH PSW
 A TO(SP-1)
 F TO(SP-2)     SP-2=0086
ANA A   A=A AND A           A=C4        F=82
PUSH PSW
 A TO (SP-1)
```

TABLE 1 Continued

```
NOP
 F TO (SP-2)    SP-2=0084
ANA B   A=A AND B          A=04       F=02
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0082
ORI     A=A IOR (B2)
 B2=7C                     A=7C       F=02
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0080
ANA C   A=A AND C          A=20       F=12
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=007E
ANI     A=A AND (B2)
 B2=65                     A=20       F=02
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=007C
XRI     A=A XOR (B2)
 B2=5C                     A=7C       F=02
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=007A
ANA D   A=A AND D          A=10       F=12
```

TABLE 1 Continued

NOP			
PUSH PSW			
A TO (SP-1)			
F TO (SP-2)	SP-2=0078		A=10
CMP C A-C	F=97		B=04
PUSH PSW			C=20
A TO (SP-1)			D=10
F TO (SP-2)	SP-2=0076		E=08
RAR	A=88	F=96	H=02
PUSH PSW			L=40
A TO (SP-1)			
F TO (SP-2)	SP-2=0074		
ANA E A=A AND E		A=08	F=12
PUSH PSW			
A TO (SP-1)			
F TO (SP-2)	SP-2=0072		
RAR		A=04	F=12
PUSH PSW			
A TO (SP-1)			
F TO (SP-2)	SP-2=0070		
RRC		A=02	F=12
PUSH PSW			
A TO (SP-1)			
F TO (SP-2)	SP-2=006E		
ANA H A=A AND H		A=02	F=02
PUSH PSW			

TABLE 1 Continued

```
NOP
 A TO (SP-1)
 F TO (SP-2)    SP-2=006C
RLC                          A=04        F=02
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=006A
CMP A    A-A                             F=56
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0068
CMP B    A-B                             F=56
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0066
CMP D    A-D                             F=93
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0064
CMP E    A-E                             F=87
PUSH PSW
 A TO(SP-1)
 F TO(SP-2)     SP-2=0062
CMP H    A-H                             F=12
PUSH PSW
 A TO (SP-1)
```

TABLE 1 Continued

```
NOP
 F TO (SP-2)    SP-2=0060
CMP M    A-(HL)                          F=12
 (HL)IN=00
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=005E
CMP L    A-L                             F=93
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=005C
RAL                          A=09        F=92
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=005A
ORA L   A=A IOR L            A=49        F=02
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0058
ANA L   A=A AND L            A=40        F=12
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0056
CPI      A-(B2)                          F=07
 B2=FF
PUSH PSW
```

TABLE 1 Continued

```
NOP
 A TO (SP-1)
 F TO (SP-2)    SP-2=0054
RLC                          A=80        F=06
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0052
RLC                          A=01        F=07
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=0050
RRC                          A=80        F=07
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=004E
RAL                          A=01        F=07
PUSH PSW
 A TO (SP-1)
 F TO (SP-2)    SP-2=004C
JMP   (B3B2) TO PC
 B2
 B3
JC    (B3B2) TO PC   (CARRY=1)
 B2
 B3
JNC   NO JUMP,CARRY=1
```

TABLE 1 Continued

```
NOP
 B2                                        A=01
 B3                                        F=07
JZ    NO JUMP,ZERO=0                CARRY,PARITY SET
 B2
 B3
JNZ   (B3B2) TO PC
 B2
 B3
JM    NO JUMP,SIGN=0
 B2
 B3
JP    (B3B2) TO PC
 B2
 B3
JPE   (B3B2) TO PC  PARITY=1
 B2
 B3
JPO   NO JUMP
 B2
 B3
CALL  (B3B2) TO PC
 B2
 B3              PC+1
 PCH TO (SP-1)
 PCL TO (SP-2) SP-2=004A
```

TABLE 1 Continued

```
NOP

RET

 (SP)    TO PCL

 (SP+1) TO PCH   SP+2=004C

CC    CALL,CARRY=1  (B3B2)TO PC

 B2

 B3              PC+1

 PCH TO (SP-1)

 PCL TO (SP-2) SP-2=004A

RC    RET,CARRY=1

 (SP)    TO PCL

 (SP+1) TO PCH SP+2=004C

CNC   NO CALL,CARRY=1

 B2

 B3

RNC   NO RET,CARRY=1

CZ    NO CALL,ZERO=0

 B2

 B3

RZ    NO RET,ZERO=0

CNZ   CALL,ZERO=0  (B3B2)  TO PC

 B2

 B3        PC+1

 PCH TO (SP-1)

 PCL TO (SP-2) SP-2=004A

RNZ   RET, ZERO=0
```

TABLE 1 Continued

```
NOP
 (SP)    TO PCL
 (SP+1) TO PCH SP+2=004C
CM    NO CALL,SIGN=0
 B2
 B3
RM    NO RET, SIGN=0
CP    CALL,SIGN=0 (B3B2)TO PC
 B2
 B3  PC+1
 PCH TO (SP-1)
 PCL TO (SP-2) SP-2=004A
RP
 (SP)    TO PCL
 (SP+1) TO PCH SP+2=004C
CPE    CALL,PARITY=1 (B3B2)TO PC
 B2
 B3     PC+1
 PCH TO (SP-1)
 PCL TO (SP-2) SP-2=004A
RPE    RET,PARITY=1
 (SP)    TO PCL
 (SP+1) TO PCH SP+2=004C
CPO   NO CALL,PARITY=1
 B2
 B3
```

TABLE 1 Continued

```
NOP
RPO       NO RET,PARITY=1
POP PSW
 (SP)  TO F                              A=03
 (SP+1)TO A     SP+2=004E                F=C2
PUSH PSW                              (SIGN,ZERO=1)
 A TO (SP-1)
 F TO (SP-2)    SP-2=004C
JMP    (B3B2) TO PC
 B2
 B3
JC     NO JUMP,CARRY=0
 B2
 B3
JNC    (B3B2) TO PC
 B2
 B3
JZ     (B3B2) TO PC
 B2
 B3
JNZ    NO JUMP,ZERO=1
 B2
 B3
JM     (B3B2) TO PC  SIGN=1
 B2
 B3
```

```
NOP
JP     NO JUMP
 B2
 B3
JPE    NO JUMP, PARITY=0
 B2
 B3
JPO    (B3B2) TO PC
 B2
 B3
CALL   (B3B2) TO PC
 B2
 B3                PC+1
 PCH TO (SP-1)
 PCL TO (SP-2)   SP-2=004A
RET
 (SP)    TO PCL
 (SP+1) TO PCH   SP+2=004C
CC     NO CALL,CARRY=0
 B2
 B3
RC     NO RET
CNC    (B3B2) TO PC , CARRY=0
 B2
 B3                PC+1
 PCH TO (SP-1)
```

TABLE 1 Continued

```
NOP
 PCL TO (SP-2)  SP-2=004A
RNC    RET, CARRY=0
 (SP)   TO PCL
 (SP+1) TO PCH  SP+2=004C
CZ     CALL,ZERO=1
 B2   (B3B2)TO PC
 B3            PC+1
 PCH TO (SP-1)
 PCL TO (SP-2) SP-2=004A
RZ    RET
 (SP)   TO PCL
 (SP+1) TO PCH SP+2=004C
CNZ    NO CALL,ZERO=1
 B2
 B3
RNZ    NO RETURN
CM     CALL,SIGN=1 (B3B2) TO PC
 B2
 B3     PC+1
 PCH TO (SP-1)
 PCL TO (SP-2) SP-2=004A
RM     RETURN
 (SP)   TO PCL
 (SP+1) TO PCH SP+2=004C
CP   NO CALL,SIGN=1
```

TABLE 1 Continued

```
NOP

 B2

 B3

RP    NO RET

CPE   NO CALL,PARITY=0

 B2

 B3

RPE   NO RET

CPO   CALL, PARITY=0 (B3B2) TO PC

 B2

 B3      PC+1

 PCH TO (SP-1)

 PCL TO (SP-2) SP-2=004A

RPO   RETURN

 (SP)    TO PCL

 (SP+1) TO PCH SP+2=004C

RST AT 0000    PC+1

 PCH TO (SP-1)

 PCL TO (SP-2) SP-2=004A

RET

 (SP)    TO PCL

 (SP+1) TO PCH SP+2=004C

RST AT 0008    PC+1

 PCH TO (SP-1)

 PCL TO (SP-2) SP-2=004A

RET
```

```
NOP

 (SP)    TO PCL

 (SP+1) TO PCH SP+2=004C

RST AT 0010    PC+1

 PCH TO (SP-1)

 PCL TO (SP-2) SP-2=004A

RET

 (SP)    TO PCL

 (SP+1) TO PCH SP+2=004C

RST AT 0018    PC+1

 PCH TO (SP-1)

 PCL TO (SP-2) SP-2=004A

RET

 (SP)    TO PCL

 (SP+1) TO PCH SP+2=004C

RST AT 0020    PC+1

 PCH TO (SP-1)

 PCL TO (SP-2) SP-2=004A

RET

 (SP)    TO PCL

 (SP+1) TO PCH SP+2=004C

RST AT 0028    PC+1

 PCH TO (SP-1)

 PCL TO (SP-2) SP-2=004A

RET

 (SP)    TO PCL
```

TABLE 1 Continued

```
NOP
 (SP+1) TO PCH SP+2=004C
RST AT 0030   PC+1
 PCH TO (SP-1)
 PCL TO (SP-2) SP-2=004A
RET
 (SP)   TO PCL
 (SP+1) TO PCH SP+2=004C
RST AT 0038   PC+1
 PCH TO (SP-1)
 PCL TO (SP-2) SP-2=004A
RET
 (SP)   TO PCL
 (SP+1) TO PCH SP+2=004C
NOP
```

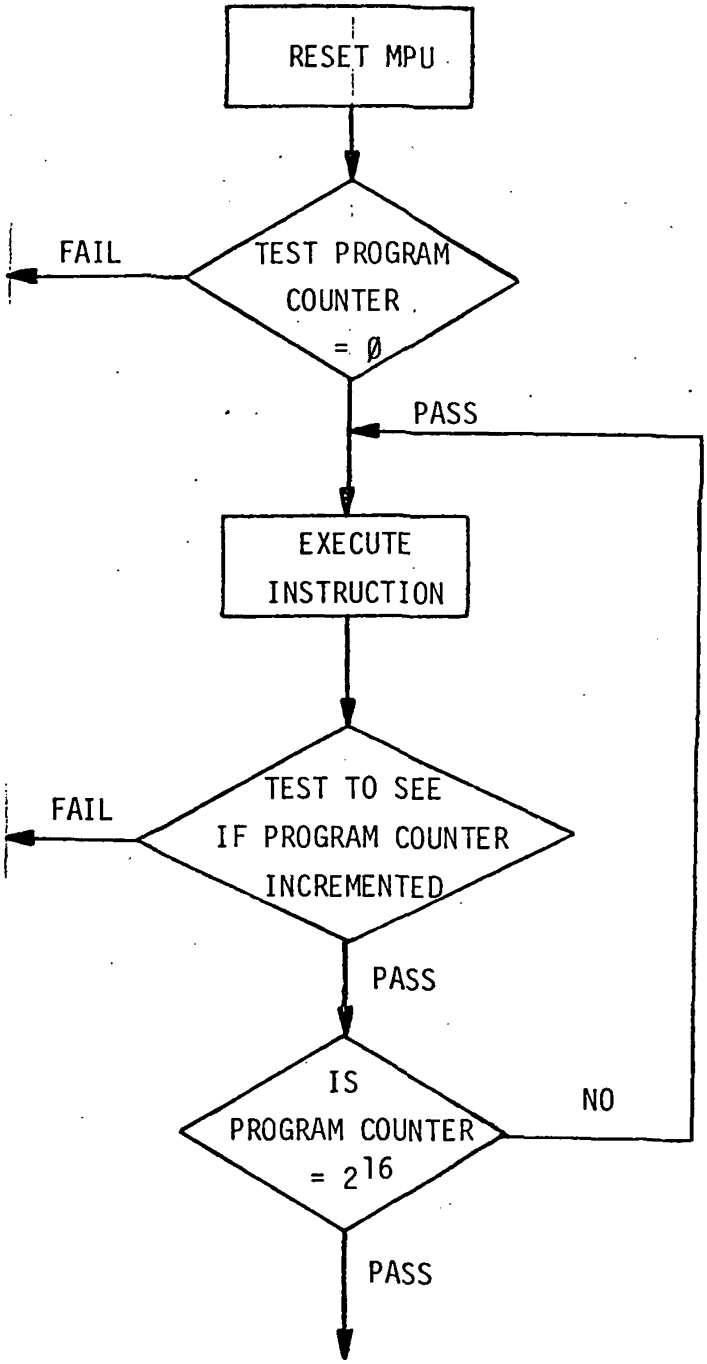


FIGURE 12: Program Counter Test

not checked is the Register Transfer to the Program Counter. This operation is verified during the Register Array Test.

Register Array Test

The test on the Register Array is accomplished by two tests. One verifies that each register is independent of any other register, and two, that any register can be moved from one register to another with any data combination. The test to determine uniqueness of each register is to first load each register (B, C, D, E, H, and L) with unique data and read to verify the load operation. The test is performed using the instruction shown in Table 2. An explanation of the instruction mnemonics appears in Table 3.

The next test on the Register Array will verify that the registers can move from one register to another and move any data combination. This test will also check register-to-Program Counter transfers. This recommended test loads the H and L registers directly with a pattern of 0's, transfers the H register contents to all other registers, and outputs from the H and L registers through the Program Counter. The pattern is incremented until all 256 numerical combinations have been checked. A flow chart of this test is illustrated in Figure 13. The instructions that should be used for this test are LXIH, PCHL, and $MOV_{r1,r2}$.

Stack Pointer Test

The Stack Pointer test is just like the Program Counter test, both

Instruction	Register	Value	
1. MVI	A	$(000)_8$	Load Routine
2. MVI	B	$(001)_8$	
3. MVI	C	$(002)_8$	
4. MVI	D	$(004)_8$	
5. MVI	E	$(010)_8$	
6. MVI	H	$(020)_8$	
7. MVI	L	$(040)_8$	
8. LXI	SP	$(125)_8(252)_8$	Test Routine
9. MOV	M,A		
10. MOV	M,B		
11. MOV	M,C		
12. MOV	M,D		
13. MOV	M,E		
14. MOV	M,H		
15. MOV	M,L		

TABLE 2: Register Array Test

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
$MOV_{r1,r2}$	Move register to register	0	1	D	D	D	S	S	S	5
MOV M, r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r, M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable Interrupt	1	1	1	1	0	0	1	1	4
NOP	No operation	0	0	0	0	0	0	0	0	4

NOTES: 1. DDD or SSS - 000 B - 001 C - 010 D - 011 E - 100 H - 101 L - 110 Memory - 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

TABLE 3: 8080 Instruction Mnemonics

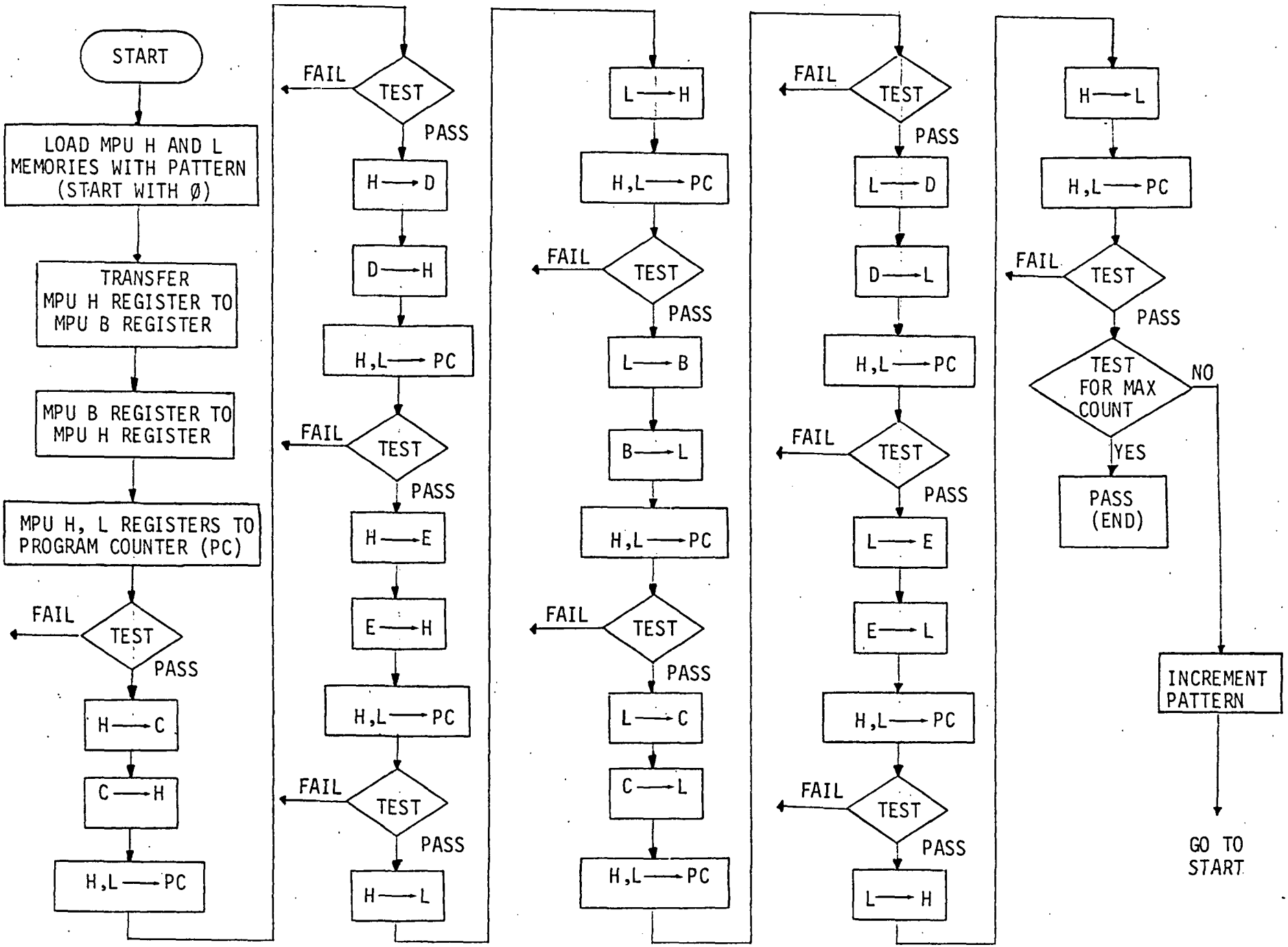


FIGURE 13: Register Array Test

are 16-bit registers with the additional feature of incrementing and decrementing. Therefore, the test on the Stack Pointer should check for incrementing and decrementing, and the ability to load and transfer to another register. Figure 14 is a flow chart of a recommended test and Table 4 lists the instructions used during the test.

Accumulator Test

The accumulator in the 8080 is 8-bits wide. A recommended test on the accumulator is to verify, load, readback, rotate, and transfer operation through its entire range. The recommended instructions (see Table 1) to be used during this test are $MOV_{A,M}$, $MOV_{M,A}$, CMA, RCL, RRC, RAL, and RAR. A flow chart of the recommended test is shown in Figure 15.

Arithmetic Logic Unit Test

The Arithmetic Logic Unit (ALU) is 8-bits wide and used to perform all arithmetic and logical data operations in the 8080. The ALU has been left until last because error analysis is simplified once all other modules have been verified. A recommended test for the ALU is to test all ALU data paths and related instructions through its range. All instructions are used during this test which operation on the ALU, such as ADD, ADC, SUB, SBB, etc. A flow chart of this recommended test is shown in Figure 16.

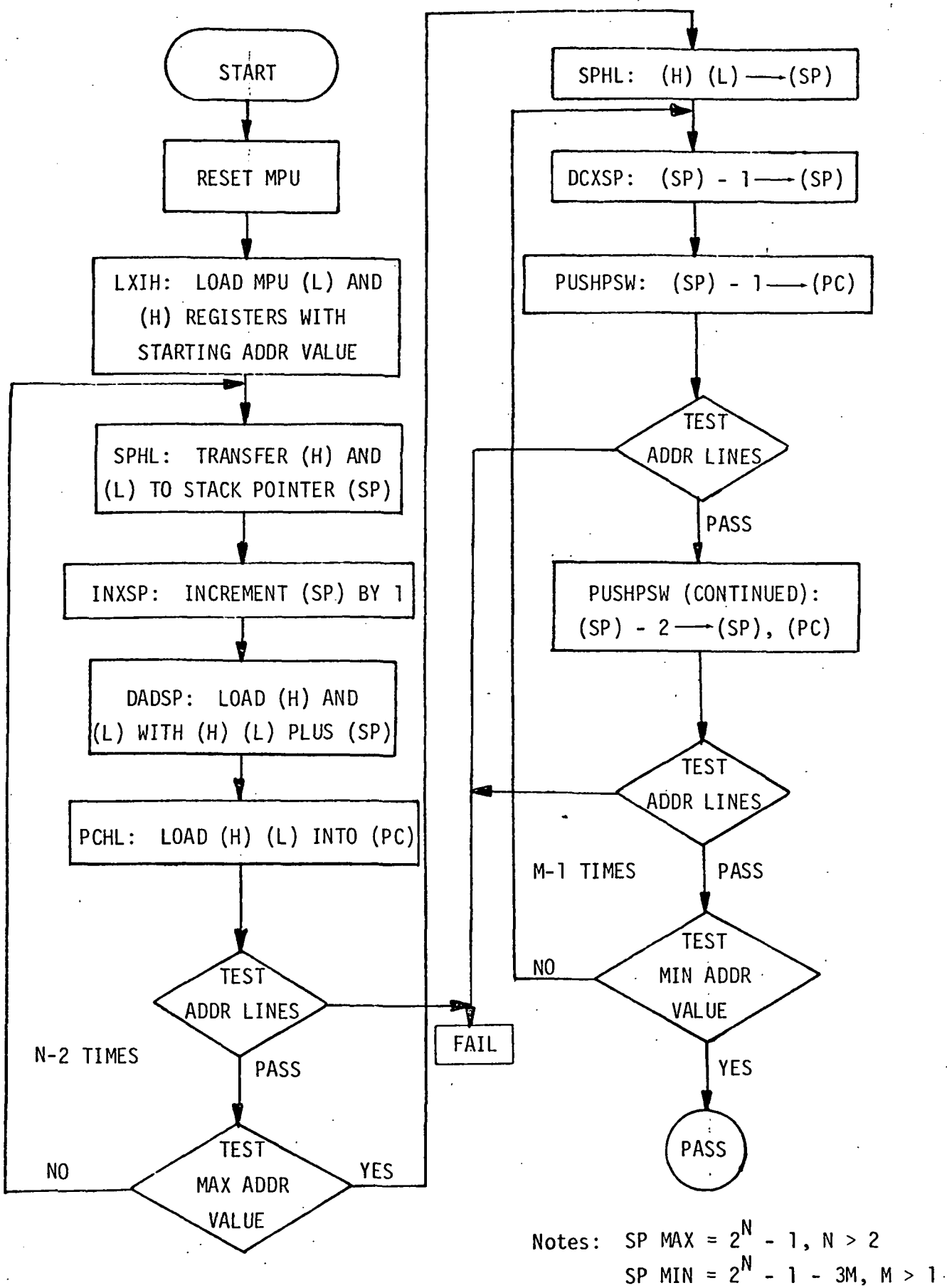


FIGURE 14: Stack Pointer Test

MNEMONICS	CYCLES	CLOCKS	CODE (DATA WORD) 7 6 5 4 3 2 1 Ø	OPERATION
LXIH	3	10	0 0 1 0 0 0 0 1	
$<B_2>$				$<B_2>$ ⟶ (L)
$<B_3>$				$<B_3>$ ⟶ (H)
SPHL	1	5	1 1 1 1 1 0 0 1	(H) (L) ⟶ (SP)
INXSP	1	5	0 0 1 1 0 0 1 1	(SP) + 1 (SP)
DADSP	3	10	0 0 1 1 1 0 0 1	(H) (L) + (SP) ⟶ (H) (L)
PCHL	1	5	1 1 1 0 1 0 0 1	(H) (L) ⟶ (PC)
DCXSP	1	5	0 0 1 1 1 0 1 1	(SP) - 1 ⟶ (SP)
PUSHPSW	3	11	1 1 1 1 0 1 0 1	(A) ⟶ [SP-1], (F) ⟶ [SP-2]

TABLE 4: Stack Pointer Test Instructions

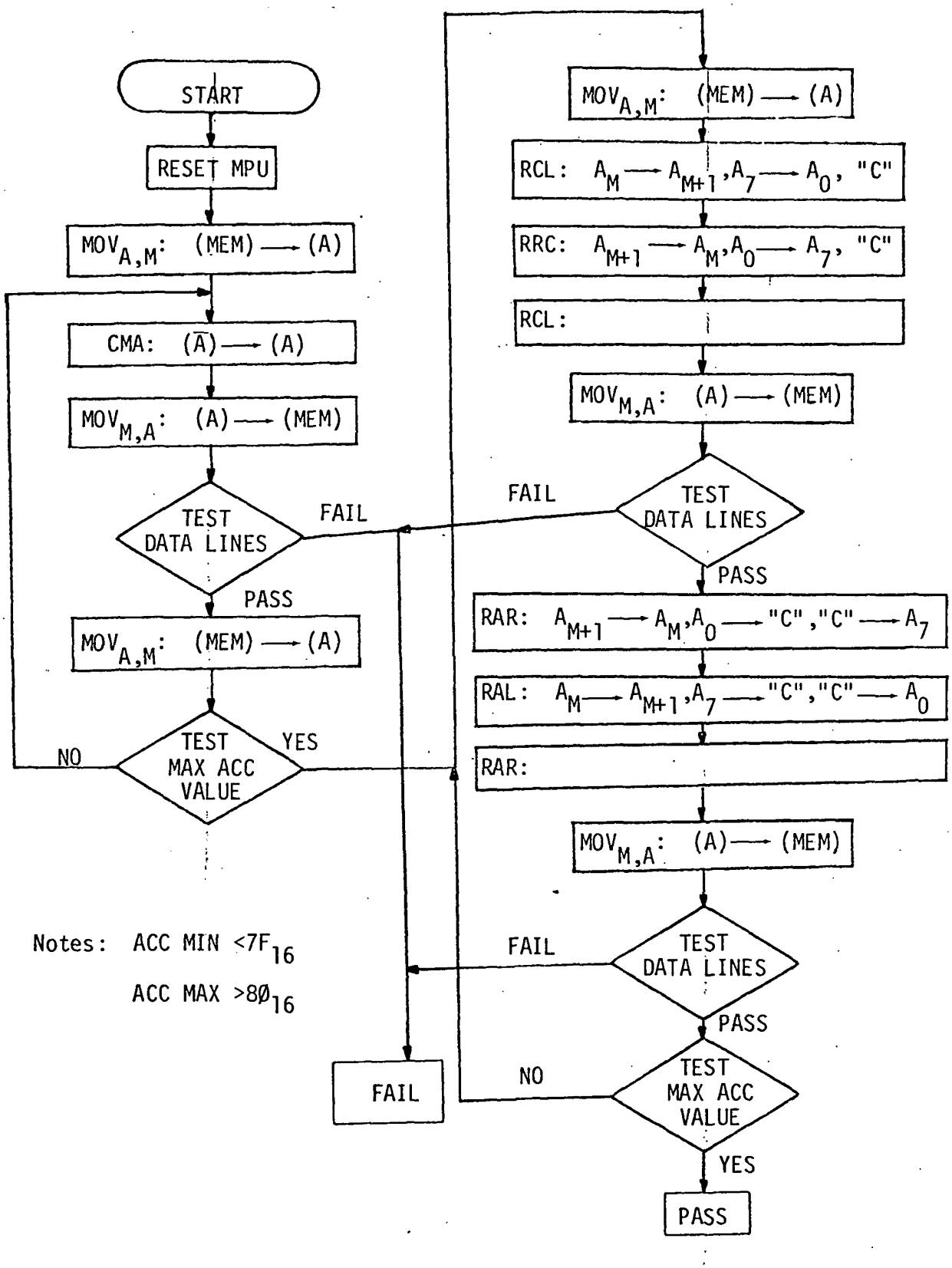
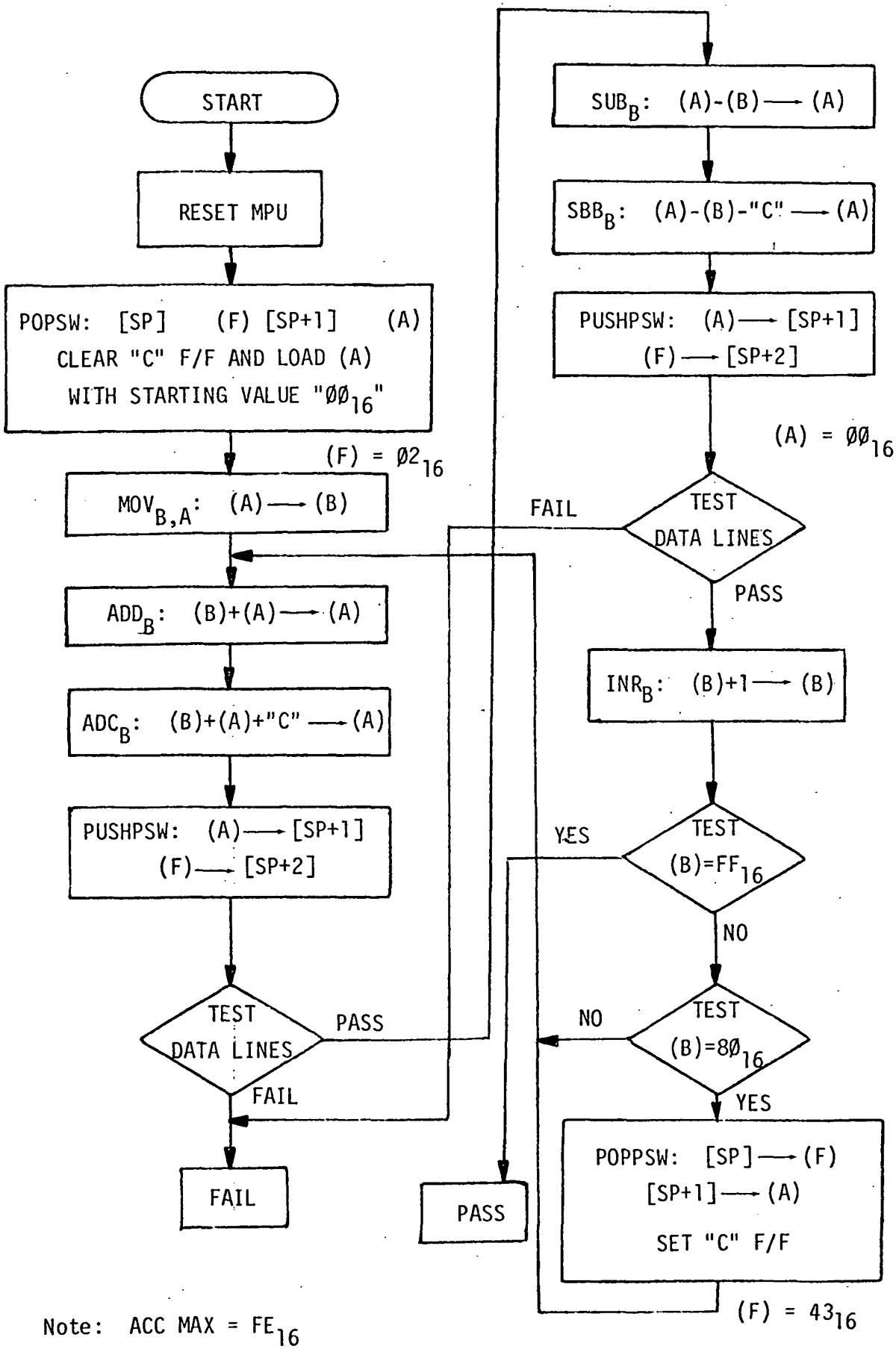


FIGURE 15: Accumulator Test



Note: ACC MAX = FE_{16}

FIGURE 16: Arithmetic Logic Unit Test

B. 8008

The test program for the 8008 is divided into the major sections listed below.

1. Accumulator
2. Register Array
3. Arithmetic Logic Unit (ALU)
4. Address Stack
5. Input/Output Instruction
6. Halt and Interrupt

Accumulator

The accumulator of the 8008 is part of the scratch pad register array with an address of $\emptyset\emptyset\emptyset_7$. This register is a working register for the arithmetic and logical instructions. Intially, a verification test would be implemented in a series of MOV instructions to load and store data to verify the basic functionality. Data patterns should consist of all 1's, all 0's, CHECKERBOARD, inverted CHECKERBOARD, 1_7, 2_7, 4_7, 10_7, 20_8, 40_8, and 100_8.

Register Array

The scratch pad register array test is designed to verify that each register can increment and decrement throughout its entire range, that each

register can be transferred to all other registers of the array, and that the H and Z registers can properly provide a correct address for the $MOV_{R,M}$ and $MOV_{M,R}$ instructions.

Initially the device is reset and all registers of the array loaded to a different value with the $MOV_{R,M}$ instruction, except register R which is set to 0. Register R1 is now incremented from 0 to 255 to 0, to verify the wrap-around characteristic. After each increment, the contents of the register is examined using the $MOV_{M,R}$ instruction.

At the completion of this process, the contents of all remaining registers are stored and verified. Register R1 is now decremented from 0 to 255 to 0, verifying the underflow characteristic. After each decrement, the register contents are stored and verified using the $MOV_{M,R}$ instruction. At the completion of this process, the contents of all other registers are read and verified.

The increment/decrement test is now performed on all remaining registers of the array.

Transfer Operations

In order to accomplish transfer operations and preserve the unique identification of all other registers, the previously verified instructions of $MOV_{R,M}$, VMIr, $MOV_{M,R}$, INRr, and DECr will be used. Initially

all registers are cleared. Register R1 will be tested first. Register R1 is incremented and transferred to R2. R1 is again incremented and transferred to R3. This process is repeated until all registers have received data from R1. The accumulator is the last register to receive data. All register contents are now stored and verified starting the the accumulator. After repeating this process 42 times, the sequence is repeated, only this time register R2 is used as the origin of all data to be transferred. All remaining registers, except the accumulator, are verified in the same manner.

Since the accumulator cannot increment, the test for verification of transfer is accomplished in a slightly different manner. Again, all registers are set to 0. Register B is then incremented and transferred to the accumulator, which is in turn transferred to register C. Register B is again incremented and transferred to the accumulator, which is now transferred to register D. This process is repeated until all registers have received data from the accumulator. All register contents are now stored and verified. This process is repeated 51 times. During this test procedure the transfer of the accumulator contents to register B is not possible, since register B is being used to generate the test pattern internal to the device. Therefore, it is necessary to repeat this test using register C to generate the internal test pattern, transferring its contents to the accumulator and then transferring the accumulator to register B, incrementing register B, and then storing all registers. The purpose of incrementing register B is to preserve the unique addressing of that register for transfer verification.

Arithmetic Logic Unit (ALU)

The previously verified instructions are now utilized to test the add, subtract, logical, and shift instructions. Results of the add and subtract instructions effect all condition flip-flops, while the rotate and shift instructions effect only the carry bit. The logical instructions do not effect the condition flip-flops.

The condition flip-flops cannot be gated to the data or address bus for purposes of verification. Therefore, it will be necessary to use the conditional jump instructions, JC, JZ, JM, JPE, JNC, JNZ, JP, and JPO.

After each arithmetic operation it is necessary to execute all six conditional jump instructions to test for proper operation of the condition flip-flops.

The data chosen should generate the criteria to set and reset all condition flip-flops resulting in patterns that will verify that the ALU can recognize a 0, negative number, even parity, and a carry, or borrow.

The data patterns required for proper verification of the ALU should be designed such that execution of the arithmetic or logical instruction being tested generates the following results:

Arithmetic Instruction

1. Positive Number
2. Negative Number
3. Even Parity
4. Non-Even Parity
5. Carry (Borrow)
6. No Carry (Borrow)
7. Zero Value
8. Non-zero Value

Logical Instructions

1. Positive Number
2. Negative Number
3. Zero Value
4. Non-zero Value
5. Even Parity
6. Non-Even Parity

Rotate Instructions

1. Carry
2. No Carry
3. Shift a 1 Through Carry
4. Shift a 0 Through Carry
5. Shift a 1 Through a Field of 0's
6. Shift a 0 Through a Field of 0's

Address Stack

Testing of the address stack is designed to verify that all levels of the stack are operative in response to the CALL and RETURN instructions. The device test program simulates repeated subroutine CALL's and RETURN's nesting up to the seven allowable levels. The program should be structured so that the carry feature from the lower order 8-bits of the address to the higher order 6-bits is verified. In addition, all conditional CALL and conditional RETURN instructions are verified. The jump instruction should also be included in this test as an easy means of manipulating the contents of the Program Counter in generating the return addresses to be stored in the address stack.

Input/Output Instructions

Verification of the input/output instructions consist of executing a series of eight input instructions each followed by an output instruction. During this sequence, the code for the selected input and output port is different so that all combinations are tested. The actual data used to write into the accumulator is not of critical importance.

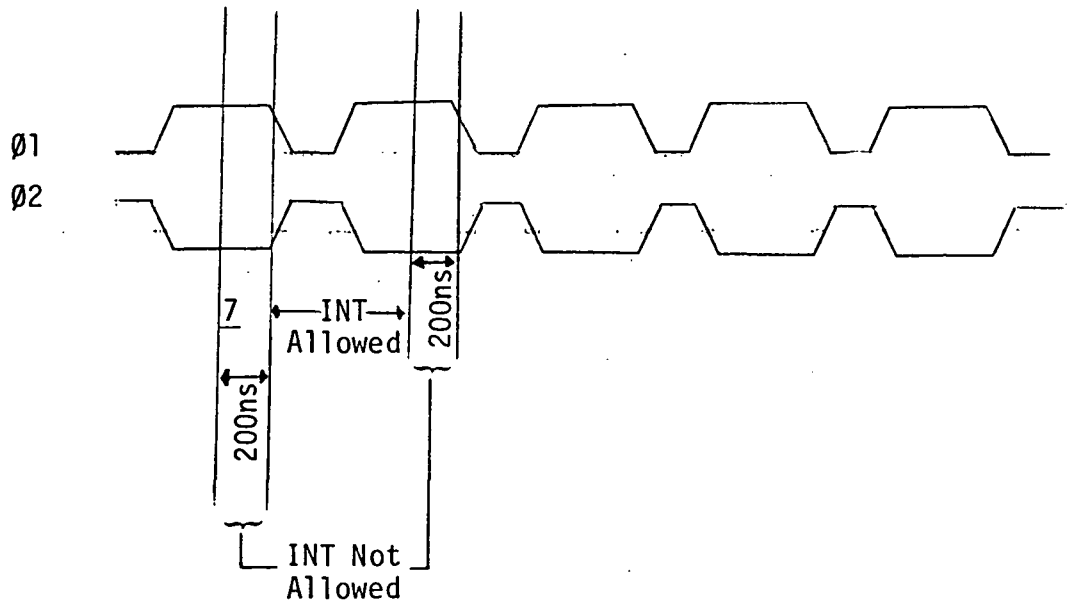
Halt and Interrupt

The Halt instruction and Interrupt feature of the 8008 can be tested together. The Interrupt is verfied first. The critical parameter of the Interrupt is that the interrupt signal to the 8008 cannot be allowed

to occur within 200 ns of the falling edge of Ø1.

The test program should verify that the 8008 will properly respond to an external interrupt which occurs within the allowable timeframe as indicated below. In addition, the Interrupt signal to the 8008 should be applied during all time states of instruction execution.

After this test, the Halt instruction is executed and the ability of the 8008 to respond to an external interrupt is verified over the same time span by executing a series of Halt instructions followed by Interrupts.



C. 2901

The 2901 4-bit bipolar microprocessor slice is not like other microprocessors being only the process portion of the typical microprocessor. Like the typical microprocessor, the 2901 has a data bus, but is not bidirectional. It also provides a register file (16 Word X 4-bits), an Accumulator (4-bits), and an Arithmetic Logic Unit (ALU).

The 2901 does not include an instruction decoder, rather all instructions directly control an operation from an code input. In a typical MPU the instruction code applied on the data bus into the decoder for the complete cycle. The 2901 complete cycle lasts only one clock cycle and if the instruction lines change during the cycle a new operation will occur. Also, the 2901 is not capable of addressing external memory directly, because it does not include an address or program counter. A typical MPU can execute jumps, subroutines, and return from subroutines due to the existance of a stack pointer which the 2901 does not contain.

The architecture of the 2901 can easily be broken up into testable modules that can be controlled and tested by the device pins and its microinstructions (see Figure 2).

The 2901 can be broken up into the following modules:

1. RAM (16 addressable registers) controlled by the "A" address field.
2. RAM (16 addressable registers) controlled by the "B" address field.

3. "Q" Register or Accumulator
4. ALU Source Selector
5. Eight Function ALU
6. Output Data Selector
7. RAM Shift
8. "Q" Shift

Examination of the microinstruction control shows that the 2901 has a 9-bit microinstruction. This microinstruction is divided into three groups: ALU source control, ALU function control, and destination control. The ALU source control controls from what data path the data will be applied into the ALU (Table 5). The ALU function controls what function the ALU will perform. For example, R field + S field, R field or S field, etc., (Table 6). The destination control routes the output of the ALU (or RAM) to different destinations within the 2901. These destinations include the RAM register stack, the "Q" register accumulator, both the RAM and "Q" register or the RAM directly out of the device (Table 7). The microinstruction controls thus route and/or manipulate data through the device.

RAM Addressable Register Test

The RAM Address Register should be divided up into four unique portions structured to test (1) the RAM using the "A" address stored through the output by passing the ALU, (2) the RAM using the "A" address outputted through the ALU, (3) the RAM using the "B" address

MICRO CODE				ALU SOURCE OPERANDS	
I_2	I_1	I_0	Octal Code	R	S
L	L	L	0	A	Q
L	L	H	1	A	B
L	H	L	2	0	Q
L	H	H	3	0	B
H	L	L	4	0	A
H	L	H	5	D	A
H	H	L	6	D	Q
H	H	H	7	D	0

TABLE 5: ALU Source

MICRO CODE				ALU Function	Symbol
I_5	I_4	I_3	Octal Code		
L	L	L	0	R Plus S	R + S
L	L	H	1	S Minus R	S - R
L	H	L	2	R Minus S	R - S
L	H	H	3	R OR S	R $\vee$ S
H	L	L	4	R AND S	R $\wedge$ S
H	L	H	5	$\overline{R}$ AND S	$\overline{R}$ $\wedge$ S
H	H	L	6	R EX-OR S	R $\veebar$ S
H	H	H	6	R EX-NOR S	$\overline{R \veebar S}$

TABLE 6: ALU Function Control

MICRO CODE				RAM FUNCTION		Q-REGISTER FUNCTION		Y OUTPUT	RAM SHIFTER		Q SHIFTER	
I_8	I_7	I_6	OCTAL CODE	SHIFT	LOAD	SHIFT	LOAD		RAM_0	RAM_3	Q_0	Q_3
L	L	L	0	X	None	None	F⟶Q	F	X	X	X	X
L	L	H	1	X	None	X	None	F	X	X	X	X
L	H	L	2	None	F⟶B	X	None	A	X	X	X	X
L	H	H	3	None	F⟶B	X	None	F	X	X	X	X
H	L	L	4	Down	F/2⟶B	Down	Q/2⟶Q	F	F_0	IN_3	Q_0	IN_3
H	L	H	5	Down	F/2⟶B	X	None	F	F_0	IN_3	Q_0	X
H	H	L	6	Up	2F⟶B	Up	2Q⟶Q	F	IN_0	F_3	IN_0	Q_3
H	H	H	7	Up	2F⟶B	X	None	F	IN_0	F_3	X	Q_3

X=Don't care. Electrically, the shift pin is a TTL input internally connected to a three-state output which is in the high-impedance state.
B=Register Address by B inputs.
Up is toward MSB.
Down is toward LSB.

TABLE 7: ALU Destination Control

outputted through the ALU, and (4) the right/left shift operation of the RAM.

To test the RAM using the "A" address outputted bypassing the ALU, the following is recommended. The object of the test is to run a GALPAT pattern on the RAM using all combinations from 0 to 15 for the test pattern and the compliment of this as the background pattern. Since the RAM can only be written into a location addressed by the "B" address, care has to be taken to address only the test location when writing into the RAM. When a location is being tested or read, the "B" address should be different then the "A" address. The easiest solution to this would be to compliment the "B" address relative to the "A" address when reading a test cell. The setup to run this test would be to set the ALU source operand to octal code 7 (D,Ø) when writing into the RAN and octal code 4 (Ø,A) when reading out a location. The ALU function is used during this test to route the data on the data input pin to the RAM. This should be programmed for a recommended function, octal code 3 (R OR S), as this will be used in a later test. The destination control should be programmed for octal code 2 which selects the RAM "A" data port to the output, bypassing the ALU. The clock pins should be held in a high state. Throughout this test the only pins that will be sampled will be the "Y" ouput pins. Once this test setup has been executed a GALPAT pattern should be performed using all test patterns of 0 to 15 and background patterns of 15 to 0. What the GALPAT does is to write a background pattern then write a test pattern. The test pattern is then read, a background

pattern location read, then the test location again, then another background location, test location, etc., until all background locations have been read. Then the test pattern is moved and the process repeated until all locations have been used as a test pattern. The test pattern is then incremented and the background pattern decremented. This process is then continued until all pattern combinations have been tested (see Figure 17 for illustration).

Performing this test will verify that all data combinations can be written into and read out of with every data combination.

The second test on the RAM is to check the RAM addressed by the "A" address field but checking the data output path through the ALU. The same test should be run as previously described with only one change in the microinstruction. This change would be to modify the destination control to an octal code 3. This modifies the output path from the RAM "A" address output to the ALU output. This would then check if the RAM "A" address path through the ALU is functional with all data sequences.

The next test on the RAM would be similar to the second test, but the "B" address and output path is checked. The changes to the second test would be to have addressing to the RAM entirely controlled by the "B" address field. During this test it is recommended that the "A" address field be the compliment of the "B" address. This would cause the worst interaction between the RAM addressing. The remaining difference would be to modify the ALU source operand to select octal code

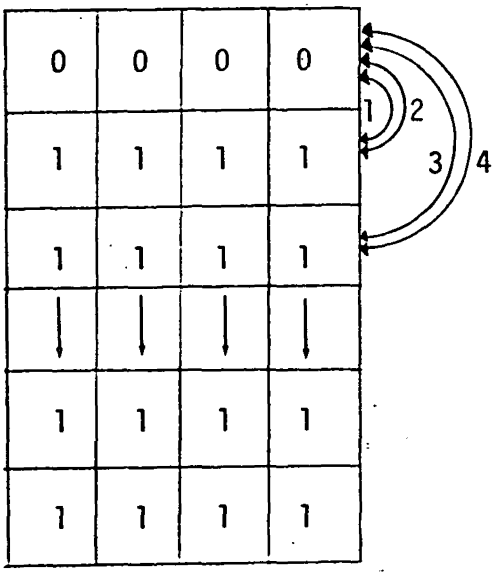


FIGURE 19: GALPAT Read Example

3 when performing a read of the RAM. This will select a source of Ø,B, thus enabling the "B" output of the RAM to pass through the RAM.

The final test to be run on the RAM is verification of the right/left shift operation. The recommended test will only describe a shift operation from the left as to test the right shift all that would be required is to input data from the right input and test the left output. The object of this test is to verify that all data combinations from 0 to 15 can be shifted through the RAM. Also, since latches are noted to be sensitive to noise, and the 4-bit output of the RAM uses a latch, the test will also recommend how to check for this. The recommended test sequence is as follows:

The test should start out by loading a 0 into location 0, and a 15 into location 15 of the RAM. The purpose of this is that one location will be used to shift data input and the other location will be used as a background test pattern. The microinstruction for the ALU source operation when writing the initial patterns should be an octal 7 which selects the data bus. All other times during the test an octal code 4 should be selected which selects "A" output latch for input to the ALU. (Note: An octal code of 3 should be selected when checking the "B" output latch.) The ALU function should be selected for an octal code 3 (R OR S) so that the output latch can be tested throughout the test. The advantage of using the R OR S function is that the ouput of the ALU will be the same as the output latch. The microinstruction for the destination control should be selected to octal code 4 which will execute

a left shift. (Octal codes 5, 6, and 7 should also be tested using this same test. When selecting codes 6 and 7, the input and output shift pins should be complimented when codes 4 and 5 are selected.) The "A" and "B" address fields should be exactly the same throughout this test. This allows an easy modification to the ALU source operand to check the "B" output latch as described earlier. Last of all, the final setup should be to produce a clock pulse each time there is a requirement to write to the RAM, but not during a test cycle when the background location is being addressed.

The test on the right/left shift will verify that (1) the shift operation will occur, (2) this shift operation can shift all combinations of 1's and 0's, (3) the output latches will hold data, and (4) the shift operation can be accomplished using any RAM address. Following the initial loading of the test and background patterns one bit of the shift pattern (101000111100101) is shifted into the RAM and the shift and Y outputs checked. Then the background address is addressed but no clock is produced and the outputs again checked. This will verify that the output latch will hold data. The next bit is now shifted in and verified and the background location addressed and data verified. This process continues until all bits have been shifted into the RAM. Then the testword and background address are incremented and decremented, respectively, and the above test repeated. This will continue until all RAM locations have been used for the test location and background location. Upon completion of the first pass, the background pattern is decremented until the initial pattern has

gone from 15 to 0. This will check for a sensitivity in the RAM output latches. This test is then repeated for both right and left shift operations on the RAM.

The next test performed is on the "Q" register. There should be two tests on the "Q" register. First, a test that will load the register with all combinations from 0 to 15 and follow each load with the compliment of the previous load. Second, a test to check the right/left shift operation on the register.

The first test should start by loading a "0" into the register and testing. Next a "15" should be loaded and tested, then 1, 14, 2, 13, ..., until a 0 and 15 are again reloaded. This test verifies that any number can be loaded into register and that all data tranactions are checked.

The second test checks the right/left shift of the "Q" register (ALU Destination Control, octal codes 4 and 6). To check these operations an initial value should be loaded into the register and checked. Destination control octal code 4 is selected and a pattern (1010000111100101) is shifted into the register. After each bit shift the register data is checked. Then an octal code is selected on the destination control and the other shift operation checked as previously described.

The next test will test the ALU source operands. This test verifies

that all ALU sources can be selected and that all data combinations can pass through the selector. The test should start by first loading RAM locations 0, 5, 10, and 15 with data equal to the address. The "Q" register is intially loaded with a 0. Using the ALU function "R OR S" (octal code 3), and a destination control which loads neither the RAM nor the "Q" register (octal code 1), the sequence of ALU source operands shown in Table 8 should be tested. During this test the "A" and "B" address will equal the data being selected by the source operand.

The ALU functions and flags should be tested next, since all other sections of the devices have now been verified as operational. First, all locations in the RAM are loaded with a data pattern equal to its address. Then data values of 0, 5, 10, and 15 and RAM values of 0, 5, 10, and 15 and CN values of 0 and 1 in all combinations are used to test each of the eight possible ALU functions. In all cases, R is the data bus and S is the "A" output from the RAM (ALU source operand, octal code 5). First, the R & S function (octal code 0) is tested. The basic sequence is as shown in Table 9.

This sequence is then repeated for each of the other ALU functions.

Function	Octal Code
S - R	1
R - S	2
R OR S	3

R	S
A = 1010	Q = 0000
A = 0101	Q = 0000
A = 1010	Q = 0000
(Load Q with 1010)	
A = 0000	Q = 1010
(Load Q with 0101)	
A = 0000	Q = 0101
(Load Q with 1010)	
A = 0000	Q = 1010
A = 1010	B = 0000
A = 0101	B = 0000
A = 1010	B = 0000
A = 0000	B = 1010
A = 0000	B = 0101
A = 0000	B = 1010
(Load Q with 1111)	
Ø	Q = 1111
Ø	B = 0000
Ø	B = 1111
Ø	A = 0000
Ø	A = 1111
D = 1010	A = 0000
D = 0101	A = 0000
D = 0101	A = 0000
D = 0000	A = 1010
D = 0000	A = 0101
D = 0000	A = 1010
(Load Q with 0000)	
D = 1111	Q = 0000
D = 0000	Q = 0000
(Load Q with 1111)	
D = 0000	Q = 1111

TABLE 8: ALU Source Operands

CN	R = A	S = D
0	0000	0000
1	0000	0000
0	1111	0000
1	1111	0000
0	0101	0000
1	0101	0000
0	1010	0000
1	1010	0000
0	0000	1111
1	0000	1111
0	1111	1111
1	1111	1111
0	0101	1111
1	0101	1111
0	1010	1111
1	1010	1111
0	0000	0101
1	0000	1010
0	1111	0101
1	1111	0101
0	0101	0101
1	0101	0101
0	1010	0101
1	1010	0101
0	0000	1010
1	0000	1010
0	1111	1010
1	1111	1010
0	0101	1010
1	0101	1010
0	1010	1010
1	1010	1010

TABLE 9: ALU Function Sequence

Function	Octal Code
R AND S	4
$\overline{R}$ AND S	5
R XOR S	6
R SNOR S	7

The last test on the device is to check to see if the output enable/disable will cause the output to go to tristate. This is accomplished by inputting a 0, 5, 10, and 15 into the Data bus and outputting it through the ALU (R OR S function) to the Y output. After each data pattern is on the Data bus the "Y" output is checked wtih the output enable. Then the outputs are disabled and the outputs checked for tristate.

D. 6800

The 6800 microprocessor unit is divided into the basic modules as listed below:

1. Program Counter
2. Stack Pointer
3. Index Registers
4. Accumulators A and B
5. Arithmetic Logic Unit
6. Timing and Control Logic
7. Interrupt Capability

For thorough testing of the 6800, the functional test sequence should thoroughly exercise each module independent of all other modules with the specific instructions applicable to that module. In addition, sufficient data patterns are used to verify proper operation of each module. An interactive type test is also performed to ensure that execution of an instruction on one module will not cause destruction of data in a different module or an otherwise malfunction of the device.

In determination of the instruction sequence, the possible discovery of instruction and/or data pattern sensitivities was not considered.

Program Counter

The Program Counter (PC) test consists of resetting the PC to 0

and then incrementing the PC through its entire range. Results of this test may be verified after each increment or after the PC has reached full value. Benefits of this test are proof that the device is basically operational, there are no stuck-at-one stuck-at-zero defects in the PC and the address bus drivers are capable of driving a logic 0 or logic 1 in any combination of bits present on the address bus.

Operation of the device during this test is as follows:

1. Reset the device.
2. Verify the reset address vectors of $FFFE_{16}$ and $FFFF_{16}$.
3. Input an instruction that will cause the PC to increment by 1.
4. Continue operation of this instruction until the PC equals $FFFF_{16}$.
5. Execute the instruction one more time to verify the overflow characteristic of the program counter.

Stack Pointer

Operational Modes:

1. Load
2. Store
3. Increment
4. Decrement
5. Transfer +1 to Index Register

6. Receiver -1 from Index Register
7. Output Data on Address Bus for:
    a. Push, Pull Data
    b. Store Device Status in Stack
    c. Pull Device Status from Stack

Stack pointer contents are available on the data bus and also the address bus during instruction execution. Accordingly the test approach is defined to verify both conditions of output. The method of defining the test approach follows that of all modules, i.e., start with instruction sequences designed to verify basic module operation, increasing the complexity of instructions for total testing of the particular module. The transfer of SP contents to the index register and transfer of index register contents to the stack pointer require verification of the index register's functionality, and will therefore be defined in the index register section of this description.

Load/Store, Data Bus

To initiate testing of the stack pointer, a load instruction is executed followed by a store instruction to output the SP contents on the data bus.

Stack Pointer Instructions

LDS Immediate, Direct, Index, Extended

STS	Direct, Index, Extended	
INS		Implied
DES		Implied
TXS		Implied
TSX		Implied

Several data patterns should be chosen such that all bits of the stack pointer have been loaded to both a logic 1 and 0. In addition, all different operational codes of the load stack pointer/store stack pointer instruction are executed at this time. This instruction sequence is defined as illustrated in Table 10.

Benefits of this test are that the stack pointer is identified as an addressable register, is capable of being loaded to several values, each bit of the stack pointer is capable of being a logic 1 or logic 0 and that each bit of the data bus is capable of driving a logic 1 or logic 0.

Increment/Decrement

Execution of this test requires initial loading of the SP to 0000_{16}, the incrementing the SP from 0000_{16} to $FFFF_{16}$ using the increment stack pointer instruction.

For detailed error analysis, the contents of the SP should be outputted to the data bus after each increment. This method may prove

INSTRUCTION	ADDRESS MODE	DATA PATTERN
Load Stack Pointer	Immediate	0000_{16}
Store Stack Pointer	Direct	-
Load Stack Pointer	Direct	$FFFF_{16}$
Store Stack Pointer	Index	-
Load Stack Pointer	Index	$AAAA_{16}$
Store Stack Pointer	Extended	-
Load Stack Pointer	Extended	5555_{16}
Store Stack Pointer	Extended	-

TABLE 10: Stack Pointer Load Routine

not feasible due to test system capability and in that case the increment stack pointer instruction would be repeated 16,384 times, and the SP contents then read. The increment stack pointer instruction is then executed one more time and the SP contents outputted to verify the overflow characteristic.

The Decrement Test is similar to the previous test with the exception of initially loading the stack pointer to $FFFF_{16}$, using the decrement stack pointer instruction and executing the decrement instruction on additional time after the SP is equal to 0 to verify the underflow characteristic.

In either of the above tests, the choice of which stack pointer store instruction to use is arbitrary and left to the discretion of the test engineer.

Address Bus Output (Push/Pull)

The Push and Pull instructions of the 6800 will cause the contents of the stack pointer to appear on the address bus and also increment or decrement the contents of this register.

Verification of this mode is performed by resetting the 6800 (getting a starting address of 0000_{16} to the PC) and execution of repeated PUL instructions. During instruction execution, the address is read

during all four to verify that the following information is present:

Cycle 1:	Program Counter
Cycle 2:	Program Counter +1
Cycle 3:	Stack Pointer
Cycle 4:	Stack Pointer +1

The PUL instruction is repeatedly executed until both PC and SP are equal to $FFFF_{16}$.

The PSH instruction is now executed in a similar manner, again verifying the address bus during all four clock cycles as follows:

Cycle 1:	Program Counter
Cycle 2:	Program Counter +1
Cycle 3:	Stack Pointer
Cycle 4:	Stack Pointer +1

This sequence is repeated until the SP is equal to 0.

Index Register (X)

Operational modes:

1.	Local	Load Immediate,	Direct, Index, Extended
2.	Store	Store	Direct, Index, Extended
3.	Increment	Increment	
4.	Decrement	Decrement	

INSTRUCTION	ADDRESS MODE	DATA PATTERN
Load Index Register	Immediate	0000_{16}
Store Index Register	Direct	-
Load Index Register	Direct	$FFFF_{16}$
Store Index Register	Index	-
Load Index Register	Index	$AAAA_{16}$
Store Index Register	Extended	-
Load Index Register	Extended	5555_{16}
Store Index Register	Extended	-

TABLE 11: Index Register Load Routine

5. Transfer to Stack Pointer TXS
6. Receive from Stack Pointer TSX

The Index Register is identical in size (16-bits X 1) and similar in operation to the Stack Pointer. Therefore, the test plan defined for this module closely parallels that of the Stack Pointer.

Load/Store

The Index Register is loaded with several data patterns, storing the register contents after each load to verify proper load operation. All different OP codes of the load and store instruction should be used to verify proper operation. The instruction sequence is defined as illustrated in Table 11.

Increment

Execution of this test requires initial loading of the Index Register to 0000_{16}, repeating execution of the increment Index Register (INX) instruction to increment the X register from 0000_{16} to $FFFF_{16}$.

As in the Stack Pointer test, the contents of the X register should be stored in the data bus after every increment. If not feasible, the increment instruction should be repeated continuously and the X register contents outputted when equal to $FFFF_{16}$. The increment instruction should then be executed one more time and the contents of the Index Register stored to verify the overflow characteristic.

Decrement

The Decrement test is similar to the previous test with the exception of initially loading the Index Register to $FFFF_{16}$, using the decrement Index Register instruction. When the X register is equal to 0000_{16}, the decrement instruction should be executed one more time and the register contents stored to verify the underflow characteristic.

Stack Pointer and Index Register Transfers

Transfers of the Stack Pointer and Index Register are limited to transferring the Stack Pointer contents +1 to the Index Register or the Index Register contents -1 to the Stack Pointer. The two instructions which define those operations are TSX and TXS respectively.

The test sequence to verify this sequence takes advantage of the functionality of these registers proven by previous tests.

Both registers are initially loaded to 0. An instruction sequence which increments the SP executes a TSX instruction and stores the Index Register contents is repeatedly executed until the Index Register is equal to $FFFF_{16}$.

This procedure is now repeated in a reverse fashion by executing a decrement Index Register, TXS, instruction followed by a read of the Stack Pointer. This instruction sequence is repeated until the Stack

Pointer is equal to 0.

Accumulators A and B

Accumulators A and B are two general purpose 8-bit registers used to store operands and results for ALU operations. The instruction set for each accumulator is similar with one or two exceptions. At this point, the definition of the different modules of the 6800 are open to different philosophies as to where one module ends and another module begins. For example, controversy may arise as to whether a logical OR instruction is an accumulator instruction or an ALU instruction.

This situation illustrates the problem of two different modules being involved in the execution of an instruction. The operation of the logical OR instruction of the contents of the Accumulator A (ACCA) with a byte of memory involves the input of a byte of memory, input of ACCA and the byte of memory to the ALU, execution of the logical OR between the two and transferring this results back to ACCA. Here two different modules are involved in the instruction execution and the question is to which module group the instruction belongs. For the purposes of clarity, this type of instruction will be attributed to the ALU module. In a more general sense, where more than one module is involved in the execution of an instruction, the instruction will be classified as belonging to the module which performs the basic operation intended by the instruction.

Accumulator A and B, Load/Store

As in the Stack Pointer and Index Register test, the initial phase of the accumulator test consists of executing a load and store accumulator routine, using all applicable operation codes in conjunction with numerous data patterns. The specific instruction sequence is defined as illustrated in Table 12. Note that the contents of the accumulator not involved in a series of instructions is stored on data bus to verify no interaction of the two accumulators.

Increment/Decrement

Accumulator A is loaded to all 0's and the increment Accumulator A instruction is executed followed by a store Accumulator A instruction. This process is continued until ACCA is equal to FF_{16}. The decrement Accumulator A instruction is now executed followed by a store ACCA instruction. This instruction sequence is repeated until ACCA is equal to 0.

The above process is repeated on Accumulator B substituting the appropriate Accumulator B instructions.

Transfer ACCA to ACCB, ACCB to ACCA

This test is designed to verify the internal transfer of accumulator to accumulator by using previously verified instructions.

INSTRUCTION	DATA PATTERN	INSTRUCTION	DATA PATTERN
Load A	FF_{16}	Load A	$4Ø_{16}$
Store A	$--_{16}$	Store A	$--_{16}$
Load B	FF_{16}	Load B	$4Ø_{16}$
Store B	$--_{16}$	Store B	$--_{16}$
Load A	55_{16}	Load A	$8Ø_{16}$
Store A	$--_{16}$	Store A	$--_{16}$
Load B	55_{16}	Load B	$8Ø_{16}$
Store B	$--_{16}$	Store B	$--_{16}$
Load A	AA_{16}	Load A	FE_{16}
Store A	$--_{16}$	Store A	$--_{16}$
Load B	AA_{16}	Load B	FE_{16}
Store B	$--_{16}$	Store B	$--_{16}$
Load A	$ØØ_{16}$	Load A	FD_{16}
Store A	$--_{16}$	Store A	$--_{16}$
Load B	$ØØ_{16}$	Load B	FD_{16}
Store B	$--_{16}$	Store B	$--_{16}$
Load A	$Ø1_{16}$	Load A	FB_{16}
Store A	$--_{16}$	Store A	$--_{16}$
Load B	$Ø1_{16}$	Load B	FB_{16}
Store B	$--_{16}$	Store B	$--_{16}$
Load A	$Ø2_{16}$	Load A	$F7_{16}$
Store A	$--_{16}$	Store A	$--_{16}$
Load B	$Ø2_{16}$	Load B	$F7_{16}$
Store B	$--_{16}$	Store B	$--_{16}$
Load A	$Ø4_{16}$	Load A	EF_{16}
Store A	$--_{16}$	Store A	$--_{16}$
Load B	$Ø4_{16}$	Load B	EF_{16}
Store B	$--_{16}$	Store B	$--_{16}$
Load A	$Ø8_{16}$	Load A	DF_{16}
Store A	$--_{16}$	Store A	$--_{16}$
Load B	$Ø8_{16}$	Load B	DF_{16}
Store B	$--_{16}$	Store B	$--_{16}$
Load A	$1Ø_{16}$	Load A	BF_{16}
Store A	$--_{16}$	Store A	$--_{16}$
Load B	$1Ø_{16}$	Load B	BF_{16}
Store B	$--_{16}$	Store B	$--_{16}$
Load A	$2Ø_{16}$	Load A	$7F_{16}$
Store A	$--_{16}$	Store A	$--_{16}$
Load B	$2Ø_{16}$	Load B	$7F_{16}$
Store B	$--_{16}$	Store B	$--_{16}$

TABLE 12: Accumulator Load Routine

Both accumulators are initially loaded to all 0's. An increment ACCA is executed followed by a transfer ACCA to ACCB, clear ACCA and then store both accumulators. Now, an increment ACCB is executed, followed by a transfer ACCB to ACCA, clear ACCB and store both accumulators. This sequence is repeated until ACCA is equal to X'FF.

Shift/Rotate Capability

The Accumulator registers of the 6800 are equipped with five modes of shift and/or rotate instructions. To properly verify the operation of these instructions, each is executed with several data patterns designed to represent worst case. Also, included in the execution of the shift and rotate instructions is verification of the Condition Code register, in particular the Carry Bit (C).

The test routine for the shift and rotate instructions initializes the MPU to a 0 state and then executes all five instructions on each accumulator. The recommended data patterns for each instruction is:

FF_{16}

AA_{16}

55_{16}

$Ø1_{16}$

$ØØ_{16}$

Each instruction is executed a total of eight times in order to

shift or rotate the data pattern through the entire accumulator. The contents of the accumulator being tested should be stored after each execution of the shift or rotate instruction. Also the contents of the Condition Code register should be stored after each eight executions of the instruction being used.

Arithmetic Logic Unit (ALU)

The function of the ALU is to perform addition, subtraction, and logical operations (OR, AND, Exclusive OR, 1's complement and 2's complement). Arithmetic comparisons can also be performed to set or reset bits of the Condition Codes register (CCR) which are testable for use in condition branch instructions.

Proper verification of the ALU includes execution and verification of all associated instructions in conjunction with worst case data patterns to verify that the ALU can add, subtract, recognize a carry, half carry, positive number, negative number and 2's complement overflow. As the CCR is an intergral portion of the ALU, its contents should be verified after execution of each instruction.

As in previous situations, the actual order of the instruction and data sequence should be structured such that, when possible, only instructions that have been previously verified are used for verification of unused instructions. The actual data patterns must be chosen such that the desired results will be generated.

Timing and Control Logic

Timing and control logic verification includes testing proper generation of the Valid Memory Address (VMA), Bus A available (BA), and Read/Write control signals (R/W). The control signals BA, VMA, and R/W are generated according to the decode of each instruction with 30 different possible combinations. Therefore, each instruction must be verified as producing the proper response of these signals.

Interrupt Capability

The 6800 microprocessor unit has been designed to offer two priority levels of hardware interrupt capability, the $\overline{IRQ}$ (maskable) and $\overline{NMI}$ (non-maskable) interrupts, $\overline{NMI}$ having priority of $\overline{IRQ}$.

Upon detection of an interrupt, the 6800 will enter the interrupt state at the end of the instruction being executed or after the completion of next instruction, depending upon what clock cycle of the present instruction execution the interrupt has occurred.

The "I" bit of the Condition Codes register has been designated as the mask bit for the $\overline{IRQ}$ interrupt. If an $\overline{IRQ}$ occurs and the "I" bit is set, the interrupt is ignored. If not, the interrupt state is entered.

The objectives of this test can now be stated as verification of

the following conditions.

1. Proper 6800 response to an $\overline{IRQ}$ interrupt by testing the data bus for storage of internal register contents, address bus for Stack Pointer address generation during the above storage and the address bus for generation of the $\overline{IRQ}$ address interrupt vector.
2. The "I" bit is set as a results of an $\overline{IRQ}$ interrupt.
3. That the 6800 will not respond to an $\overline{IRQ}$ interrupt when the "I" bit of the CCR is set.
4. Proper response to an $\overline{NMI}$ interrupt when the "I" bit is set and reset.
5. Priority of the $\overline{NMI}$ interrupt over $\overline{IRQ}$ by causing both signals to indicate interrupts simultaneously.

A third mode of interrupt is under software control by means of the SWI (Software Interrupt Instruction). Execution of this instruction is not hardware related and will therefore be executed whenever it occurs in the user program. This instruction is verified by testing the data and address bus for proper storage of internal 6800 register contents and the generation of the SWI address interrupt vector.

Execution of the WAI (Wait for Interrupt Instruction) stores all internal register in the stack and then places the 6800 in an inactive wait state. The device will remain in this state until either an $\overline{IRQ}$ or $\overline{NMI}$ interrupt occurs. This instruction is verified by first observing the data and address bus during internal register content storage

and second that an $\overline{IRQ}$ and $\overline{NMI}$ interrupt will be allowed to respond as previously described for these signals.

The $\overline{IRQ}$ and $\overline{NMI}$ signals are asynchronous and as such should be tested for interrupt generating capability by causing the interrupts to occur within several timeframes. First each interrupt should occur such that the recognition routine starts after completion of the present instruction being executed at the time of interrupt and second, after completion of the next instruction at the time of interrupt.

Instruction Decode Test

The Instruction Decode test verifies proper execution of all jump, branch, and subroutine instructions.

The major aspect of the jump instruction is to test for proper address generation in response to the two addressing modes of this instruction.

Testing of the branch instructions requires execution of each instruction and testing that (1) the branch address is generated, if the branch condition is true, and (2) that the branch does not occur, if the associated condition is false.

Subroutine instructions tests are required to verify that (1) the Stack Pointer address occurs on the address bus simultaneously with the

return address on the data bus, (2) the correct subroutine address is generated, and (3) that the return from subroutine generates the Stack Pointer address on the address bus for the purpose of pulling the return address from stack.

E. 1802

The 1802 microprocessor unit is a static 8-bit device employing CMOS technology. The device provides the following internal architecture (see Figure 11).

1. 16-bit by 16-bit Register Array
2. 8-bit Arithmetic Logic Unit (ALU)
3. 8-bit Accumulator (D)
4. Two 4-bit Instruction Registers (I and N)
5. A 4-bit Register P used to specify which of the 16-bit Registers in (1) is the present program counter.
6. A 4-bit Auxiliary Register (X)
7. An 8-bit Temporary Register (T)
8. A 1-bit Register (Q)

Examination of the instruction set of the 1802 reveals that the major data path to and from the internal register array is through the D register. Therefore, this module of the 1802 is of extreme importance and the test program will exercise this module fully as an initial starting point. Next, the uniqueness and functionality of the 16-bit by 16-bit register array will be proven. Arithmetic and Logical instructions will be tested next followed by the Branch and Skip instructions.

A unique feature of the 1802 is a built-in DMA feature which uses an internal register as a counter for the number of bytes transferred to or from memory. This feature is evaluated for both the DMA in and

DMA out modes of operation. The Interrupt feature is verified for proper operation and also tested for its masking capability.

D Register

The importance of the D register is its function of being the path by which to load or store contents of the scratch pad register array via the data bus and as a working register of the arithmetic logic unit. The initial phase of the test on this register is to ensure the ability to load worst case data patterns in the D register and also store the same.

Execution of this test consists of a series of load instructions to walk a 1 through a field of 0's and a 0 through a field of 1's, each load instruction being followed by a store to verify the load operation.

Register Array

The purpose of the register array is to provide a program counter, 16-bit vectored interrupt address storage, DMA address counter, and general purpose scratch pad registers. The initial test on this module consists of a series of instructions to verify that each register can be loaded to worst case data patterns and that each register can be accessed for the retrival of this information. All input and storage of data patterns to the register array will take place through the D

register. An important point is that at all times one of the 16 registers is being utilized as a program counter, as determined by the value in the 4-bit P register. Upon initial start-up and reset of the 1802, the P register is reset to 0, making R(0) the current program counter. Therefore registers R(1) through R(15) are tested first and then the SET P instruction must be executed to change the register being used as the program counter. Register R(0) is then tested in the same manner as the others. Due to the use of R(0) as a program counter during this exercise, R(0) should be stored through the D register at the completion of this test to check that it has been incrementing during the execution of the test. Then a test sequence which loads and stores worst case data patterns can be executed. The actual test sequence of loading and storing data patterns in the register array should use different data such that the uniqueness of each register is proven.

The next portion of the Register Array test will verify operation of the increment and decrement instructions, INC and DEC.

The procedure is to verify that each of the 16 registers of the register array can increment and decrement throughout the entire range of 0 to 2^{15} -1. Also to be verified is the over and underflow characteristics of each register. Registers R(1) through R(15) are to be tested first with R(0) acting as the program counter. Then R(0) is tested with R(1) as the program counter. The test procedure is as

follows:

1. Reset device.
2. Load registers R(2) through R(15), each with a distinct data pattern.
3. Load register R(1) with 0 and using the increment N instruction, cause this register to increment from 0 to 2^{15} -1. Then execute the increment instruction an additional time to cause R(1) to overflow to 0. Each increment instruction should be followed by PUT low register N and PUT high register N instructions to verify the increment.
4. At the completion of step 3, all other registers should be stored on the data bus to verify that no destructive interaction has occurred.
5. The decrement register N instruction is now executed to cause register R(1) to decrement from 0 to 2^{15} -1, and then to 0. Again, each decrement instruction is followed by a PUT low register N and PUT high register N instruction to verify each decrement.
6. Registers R(2) through R(15) are now read onto the data bus to verify no destructive interaction.
7. This process is repeated until registers R(1) through R(15) have been tested.
8. A SET P instruction is executed to change the current program counter from R(0) to R(1).
9. Register R(0) is stored on the data bus and its present contents verified.

10. R(0) is loaded to 0 and the same procedure is followed for verification as described above.

X Register

The purpose of the X register is to hold a four bit code used to designate one of the 16 registers of the register array for use in certain load and store instructions. Upon initial reset of the MPU, this register is reset to 0 and then may be loaded to another value by the SET X instruction. Proper verification of the operation of this register is to reset the MPU, and execute a load via X or store via X instruction. The value which will appear on the address bus will be the contents of register R(0) which is also the current contents of the program counter as a reset will clear the P register to 0.

At this point the SET X instruction is executed to designate R(1) and the load via X or store via X instruction executed. This process is repeated until all registers have been designated by the X register. It is important to note that all registers should be loaded to different values in order to prove that the R(X) register is actually present on the address bus.

P Register

The P register is used to hold a four bit code used to designate

which of the 16 registers of the register array is the current program counter. The verification of the register operation is performed in a similar manner to that of the X register.

The device is reset, which should clear the P register and register R(0) to 0.

The initial portion of this program after reset should load the register array such that each register contains a different value. By doing this, each register can be uniquely identified as it is gated to the address bus. After verification of R(0) as the program counter the SET P instruction should be executed to change the current program counter from R(0) to R(1) and the address bus monitored. All remaining values of the P register are verified in the same manner.

Q Register

The Q register is a 1-bit register which can be set or reset under program control. The Q register bit is also cleared after an initial clear is performed. Also, the status of this bit can be tested by several of the branch instructions. However, this portion of the Q register test will not utilize the branch instruction as a part of the test.

The 1802 is initially cleared and the Q bit tested for the logic 0 state. The SET Q instruction is executed and then reset, the state of

the Q bit being tested after each operation. This procedure can be repeated several times to ensure proper operation.

Arithmetic Logic Unit

The test of the arithmetic logic unit is divided into two sections, the logical operations and the arithmetic operations. Also all addressing modes included in this portion of the instruction test are verified together with the operation of the DF flag.

Logical Instruction Test

The purpose of this test is to verify that all logical instructions are operational and that worst case data patterns have no effect on functionality of the device.

For the instructions of OR, Exclusive OR, and AND, worst case data patterns are defined as those patterns that cause each bit in the result to be either set or reset according to the instruction being tested. Examples are illustrated in Figures 18, 19, and 20.

Initially the OR instruction is executed with the data patterns specified. The D register is loaded, the OR instruction executed and the D register stored on the data bus to verify the results. This test is executed twice. The first time the OR instruction is used and the second time the OR IMMEDIATE instruction is used.

Pattern 1	Byte 1	1 0 1 0 1 0 1 0
	Byte 2	0 1 0 1 0 1 0 1
	Result	1 1 1 1 1 1 1 1
Pattern 2	Byte 1	0 1 0 1 0 1 0 1
	Byte 2	1 0 1 0 1 0 1 0
	Result	1 1 1 1 1 1 1 1
Pattern 3	Byte 1	1 1 1 1 1 1 1 1
	Byte 2	0 0 0 0 0 0 0 0
	Result	1 1 1 1 1 1 1 1
Pattern 4	Byte 1	0 0 0 0 0 0 0 0
	Byte 2	1 1 1 1 1 1 1 1
	Result	1 1 1 1 1 1 1 1
Pattern 5	Byte 1	0 0 0 0 0 0 0 0
	Byte 2	0 0 0 0 0 0 0 0
	Result	0 0 0 0 0 0 0 0

FIGURE 18: 1802--OR Data Pattern

Pattern 1	Byte 1	1 0 1 0 1 0 1 0
	Byte 2	0 1 0 1 0 1 0 1
	Result	1 1 1 1 1 1 1 1
Pattern 2	Byte 1	0 1 0 1 0 1 0 1
	Byte 2	1 0 1 0 1 0 1 0
	Result	1 1 1 1 1 1 1 1
Pattern 3	Byte 1	1 1 1 1 1 1 1 1
	Byte 2	1 1 1 1 1 1 1 1
	Result	0 0 0 0 0 0 0 0
Pattern 4	Byte 1	1 1 1 1 1 1 1 1
	Byte 2	0 0 0 0 0 0 0 0
	Result	1 1 1 1 1 1 1 1
Pattern 5	Byte 1	0 0 0 0 0 0 0 0
	Byte 2	1 1 1 1 1 1 1 1
	Result	1 1 1 1 1 1 1 1
Pattern 6	Byte 1	0 0 0 0 0 0 0 0
	Byte 2	0 0 0 0 0 0 0 0
	Result	0 0 0 0 0 0 0 0
Pattern 7	Byte 1	1 0 1 0 1 0 1 0
	Byte 2	0 0 0 0 0 0 0 0
	Result	1 0 1 0 1 0 1 0
Pattern 8	Byte 1	0 1 0 1 0 1 0 1
	Byte 2	0 0 0 0 0 0 0 0
	Result	0 1 0 1 0 1 0 1
Pattern 9	Byte 1	0 0 0 0 0 0 0 0
	Byte 2	1 0 1 0 1 0 1 0
	Result	1 0 1 0 1 0 1 0
Pattern 10	Byte 1	0 0 0 0 0 0 0 0
	Byte 2	0 1 0 1 0 1 0 1
	Result	0 1 0 1 0 1 0 1

FIGURE 19: 1802--Exclusive OR Data Pattern

Pattern 1	Byte 1	1 1 1 1 1 1 1 1
	Byte 2	1 1 1 1 1 1 1 1
	Result	1 1 1 1 1 1 1 1
Pattern 2	Byte 1	1 1 1 1 1 1 1 1
	Byte 2	0 0 0 0 0 0 0 0
	Result	1 1 1 1 1 1 1 1
Pattern 3	Byte 1	1 0 1 0 1 0 1 0
	Byte 2	0 1 0 1 0 1 0 1
	Result	1 1 1 1 1 1 1 1
Pattern 4	Byte 1	0 1 0 1 0 1 0 1
	Byte 2	1 0 1 0 1 0 1 0
	Result	1 1 1 1 1 1 1 1
Pattern 5	Byte 1	0 0 0 0 0 0 0 0
	Byte 2	1 1 1 1 1 1 1 1
	Result	1 1 1 1 1 1 1 1
Pattern 6	Byte 1	0 0 0 0 0 0 0 0
	Byte 2	0 0 0 0 0 0 0 0
	Result	0 0 0 0 0 0 0 0

FIGURE: 20: 1802--AND Data Pattern

The Exclusive OR, EXCLUSIVE or IMMEDIATE, AND and AND IMMEDIATE instructions are executed in the same manner.

The four shift instructions are verified using the same philosophy for worst case data patterns as for the OR and AND instructions. One of the functions of the DF bit will be used and therefore requires verification.

The procedure for verification consists of loading the D register with a test pattern, executing the particular shift instruction eight times, storing the contents of the D register after each instruction execution.

Verification of the proper operation of the DF bit can only be made by designing a test program such that the DF bit is left to an expected known state. This state is then used as a starting point for the next data pattern. For example, if the completion of a shift instruction has put the DF bit to a logic 1, the next shift instruction to be executed would be one that shifted the DF bit to either the least or most significant bit of the D register.

For the shift instructions, the following data patterns can be used as initial values:

Shift Right:	55_{16}, AA_{16}, FF_{16}, $8Ø_{16}$, $ØØ_{16}$
Shift Right with Carry:	55_{16}, AA_{16}, FF_{16}, $8Ø_{16}$, $Ø1_{16}$, $ØØ_{16}$

Shift Left:	55_{16}, AA_{16}, FF_{16}, $8Ø_{16}$, $Ø1_{16}$
Shift Left with Carry:	55_{16}, AA_{16}, FF_{16}, $8Ø_{16}$, $Ø1_{16}$, $ØØ_{16}$

Arithmetic Operations

The object of this portion of the test program on the ALU is to verify that the ALU can add, subtract, with and without a carry or borrow, respectively, detect an overflow or underflow condition via the DF bit, and that the register and immediate addressing modes are functional.

Suggested data patterns for the arithmetic instructions are as follows:

Add, Add Immediate:	FF_{16} to FF_{16}, 55_{16} to 55_{16}, AA_{16} to AA_{16}, $ØØ_{16}$ to $ØØ_{16}$, FØ to $ØF_{16}$
Add with Carry, Add with Carry Immediate	OD_{16} to CC_{16}, FF_{16} to FF_{16}, $FØ_{16}$ to $8Ø_{16}$
Subtract, Subtract Immediate	FF_{16} from FF_{16}, 55_{16} from AA_{16}, AA_{16} from 55_{16}
Subtract with Borrow, Subtract with Borrow Immediate	FF_{16} from $Ø1_{16}$, $ØF_{16}$ from $ØF_{16}$, $8Ø_{16}$ from 99_{16}, $6Ø_{16}$ from $7Ø_{16}$

Subtract Memory, FF_{16} from FF_{16}, 55_{16} from AA_{16}, AA_{16} from
Subtract Memory Immediate 55_{16}

Subtract Memory with Borrow, FF_{16} from $\emptyset 1_{16}$, $\emptyset F_{16}$ from $\emptyset F_{16}$,
Subtract Memory with Borrow Immediate $8\emptyset_{16}$ from 99_{16}, $6\emptyset_{16}$ from $7\emptyset_{16}$

Two methods exist for verifying proper operation of the DF bit during execution of these instructions. The first is to follow each add or subtract instruction by an add with carry or subtract with borrow. This second add or subtract instruction will verify the proper DF bit operation if the results are what is expected as a result of the instruction execution.

The second is to execute a shift right with carry or shift left with carry to put the value of DF into the MSB or LSB of the D register respectively. The contents of the D register are now read and the MSB or LSB verified to reflect the expected state of the DF bit. This is the preferred method for several reasons. First, if a failure occurs using the first method, the cause of the failure could be that the ALU did not detect the original overflow or could not execute the add or subtract with carry. As the shift instructions have previously been verified, this mode of verification pinpoints the cause of failure.

Branch and Skip Instructions (Long & Short)

The branch and skip instructions are verified by causing the condition tested by the particular instruction to occur and then executing

the associated branch or skip instruction. The address bus is tested for generation of the expected branch address. Alternately, the opposite condition is verified by executing the necessary branch or skip instruction when the branch condition does not exist and verifying that the branch or skip does not execute. The conditions tested for in the branch and skip are the following:

Short Branch if:	D = Ø
	D ≠ Ø
	DF = 1
	DF = Ø
	Q = 1
	Q = Ø
	EF1 = 1
	EF1 = Ø
	EF2 = 1
	EF2 = Ø
	EF3 = 1
	EF3 = Ø
	EF4 = 1
	EF4 = Ø
	Always, Never

The short branch and long branch are similar with the exception that the long branch provides an absolute branch address, while the short branch provides an address which is 0 to +255 locations from the address containing the short branch instruction. The conditions tested

for the long branch instruction is limited to the states of the D register, DF bit, and Q bit. Specifically these are:

Long Branch if:	D = Ø
	D ≠ Ø
	DF = 1
	DF ≠ 1
	Q = 1
	Q ≠ 1
	Always, Never

The skip instructions are similar to the branch instructions except no branch address is required. The conditions tested are as follows:

Short Skip:	Never
Long Skip:	Always
Long Skip if:	D = Ø
	D ≠ Ø
	DF = 1
	DF = Ø
	Q = 1
	Q = Ø
	IE = 1

Interrupt

The response of the 1802 to an asychronous Interrupt is tested by causing the Interrupt input to become active and verifying that the following states occur:

1. The instruction in process at the time of interrupt is completed.
2. The next machine cycle is a normal fetch except the address gated to the address bus is from register R(1).
3. The X register has been set to 2_{10}.
4. The state codes indicate Interrupt recognition.
5. The IE enable bit has been reset by causing the Interrupt input to indicate additional Interrupts and verifying that they are ignored.
6. The values of registers X and P have been saved in the T register. (This can be accomplished by execution of a MARK isntruction).
7. The Interrupt mode of operation is asynchronous by repeating the test in every clock cycle of instruction execution.

$\overline{\text{DMA-In-Out}}$

The $\overline{\text{DMA-In-Out}}$ features are tested in a manner similar to that of the Interrupt with all expected activities verified.

$\overline{\text{DMA In}}$

1. $\overline{\text{DMA-IN}}$ is caused to become active.

2. At the completion of the present instruction, verify the state codes indicate $\overline{\text{DMA-In}}$, register R(0) is gated to the address bus and $\overline{\text{MWR}}$ is active.
3. Item 2 is repeated for as long as $\overline{\text{DMA-In}}$ remains active, with register R(0) being incremented after each transfer.
4. Normal program execution is resumed when $\overline{\text{DMA-In}}$ becomes in-active.

$\overline{\text{DMA-Out}}$

1. $\overline{\text{DMA-Out}}$ is caused to become active.
2. At the completion of the present instruction execution, the state codes indicate $\overline{\text{DMA-Out}}$, $\overline{\text{MRD}}$ is active, and register R(0) is gated to the address bus.
3. Item 2 is repeated for as long as $\overline{\text{DMA-Out}}$ is active.
4. Normal program execution is resumed when $\overline{\text{DMA-Out}}$ becomes in-active.

At this point, the priority of the previous tests should be verified such that a DMA and Interrupt request occur simultaneously. The order of priority is $\overline{\text{DMA-In}}$ first, $\overline{\text{DMA-Out}}$ second, and Interrupt last.

Input/Output Transfers

The test to verify the input and output instruction capability of the 1802 is performed separately for proper operation. Each instruction should be executed with all possible combinations of I/O device selections, testing for proper access of the least three significant bits on output pins N0, N1, and N2, and the contents of register R(X) being

VI. DC TEST REQUIREMENTS

Although the major portion of this report has been devoted to testing of the functional characteristics of microprocessor units, the importance of DC testing should not be de-emphasized. As with other semiconductor devices, microprocessor units malfunction as a results of DC characteristics being out of specification. Therefore, it is recommended that any complete test on a microprocessor unit include verification of the manufacturers specified DC characteristics.

The commonly specified DC parameters are input and output voltage levels, input and output currents and leakages, tristate leakage currents, power supply voltages and power supply currents. Proper verification and/or measurement of each parameter should be performed, simulating the necessary condition for accurate test execution. Refer to Attachment I for DC specifications of each device.

VII. SURVEY SUMMARY

A. List of Companies Interviewed

Advanced Micro Devices, Sunnyvale, California

American Micro Systems, Incorporated, Cupertino, California

Boeing Aerospace, Seattle, Washington

Burroughs Corporation, Pasadena, California

Chrysler Corporation, Hunstville, Alabama

Fairchild Systems & Technology, San Jose, California

General Electric Company, Pittsfield, Massachusetts

Hewlett Packard, Palo Alto, California

Hughes Aircraft Corporation, Culver City, California

Intel Corporation, Santa Clara, California

Motorola, Austin, Texas

Motorola, Phoenix, Arizona

National Semiconductor, Santa Clara, California

RCA, Sommerville, New Jersey

Rockwell International, Anaheim, California

Tektronics, Beaverton, Oregon

Texas Instruments, Houston, Texas

B. REVIEW OF PRESENT MICROPROCESSOR TEST TECHNIQUES QUESTIONNAIRE

I. TEST EQUIPMENT

A. Tester (Which Device on Which Tester)

B. Clock Speed of Tester

C. Burn-in Equipment

1. Type Used

2. Static

3. Dynamic

4. What Type

II. DC TEST (PRODUCTION)

A. Parameters Tested

1. What DC Parameters Are Tested

2. Are Voltage Measurements Done DC Static or Functional

3. IF DC, How Long Is Sample

4. IF AC, Is VOH and VOL Measured One Pass or Two Pass

B. Execution Time (Delete Overhead)

C. Overhead Time

D. Percentage of Total Test Program

E. Differences Between Wafer and Final Package DC Tests

F. Type of Failures Observed

III. FUNCTIONAL TESTS (PRODUCTION)

A. Test Pattern

1. Method of Generation

2. Instruction Sequence (What Do They Test For)
    a. Modular
    b. Other
3. Gold Device
    a. As A Comparison Test
    b. As A Learn Method
    c. Self Diagnostic (Board Test)
4. Pattern Length
5. Pattern Sensitivity
6. Frequency of Testing Device Output(s)
    a. Each Cycle
    b. End of Operation
    c. Other

B. Functional Test Conditions
1. Device Timing
2. AC Parameters
    a. Rise/Fall Times
    b. Minimum Pulses
    c. Access Times
3. Execution Time (Delete Overhead)
4. Overhead Time
5. Error Analysis
    a. Why Device Failed
    b. What Instruction
    c. What Data Pattern
    d. What Pin(s)

C. Percentage of Total Test Program

D. Types of Failures Discovered

E. Differences Between Wafer and Final Package Functional Test Programs

IV. CHARACTERIZATION EFFORTS

A. Parameters Characterized

1. Functional and AC

2. DC

B. Temperature Conditions

C. Burn-in Conditions

D. Form of Characterization Data Log

1. Histogram

2. Shmoo Plot

3. Other

E. Number of Devices Characterized

F. Department Responsible for Characterization

V. PRODUCTION TESTING

A. Temperature Conditions

1. Hold, Cold, Ambient

2. If Not Done, Why

B. Burn-in Conditions

1. What Temperature

2. AC or Static

3. What Loads

4. If Not Done, Why

C. Data Logging

1. Bin Classification

2. Hardcopy

a. What Is Obtained

D. Location Performed

E. Percentage of Devices Screened

F. Department Responsible for Production Testing

G. 38510 Specification--Yes/No (If Yes, Who Wrote It)

H. Types of Failures

VI. WHAT TYPE OF PROBLEMS ARE YOU FINDING

VII. RECOMMENDATIONS FOR USER TESTING OF MICROPROCESSORS

C. QUESTIONNAIRE RESPONSES

I. TEST EQUIPMENT

A. Tester

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Fairchild Sentry II	2	1		1	
2. Fairchild Sentry 600	2	2			
3. Macrodata MD-154					1
4. Tektronics S-3260	3	1		1	
5. Teradyne J277	1				
6. Teradyne J283				1	1
7. Teradyne J293				1	
8. In-House System			1		1

B. Burn-In Equipment

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Blue M	1		1	1	
2. In-house Design	5	1		3	
3. Commercially Available	1	1			
4. Not Being Performed					1

II. DC TESTS (PRODUCTION)

A. Parameters Tested

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. All Data Sheet Parameters	5	2	1	3	1
2. All Data Sheet Parameters Plus Several Unspecified Parameters		1			

B. Voltage Measurements, Static or Dynamic

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Dynamic	4	2			
2. Static				4	1
3. Clocked Very Slow (Considered Static)	1		1		
4. Static Where Possible Dynamic Otherwise	1	1			

C. Length of Sample Time, If Voltage Measurements are Static

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. 5 ms	1				
2. 10 ms	1	1	1	3	1
3. Dependent On Parameter				1	
4. Don't Know	2				

D. VOL and VOH Measurements

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. AC Measurement--Made In One Pass Using Differential Voltage Comparators.	4	2	2		1
2. AC Measurement--Made In Two Passes.		1		4	
3. AC Measurement--Made In Separate Passes.	2				
4. DC Measurement--Made In Static Mode.				1	

E. DC Test Execution Time

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. 1/4 Seconds Total Test Time, Breakdown Not Available	1				
2. 2 Seconds Total Test Time, Breakdown Not Available	1		1		
3. 3 Seconds Total Test Time, Breakdown Not Available		1			
4. 3.5 Seconds Total Test Time, Breakdown Not Available				1	
5. 5 Seconds Total Test Time, Breakdown Not Available		1		1	
6. Up to 9 Seconds Total Test Time, Breakdown Not Available					1
7. 10 Seconds Total Test Time, Breakdown Not Available		1			
8. 20 Seconds Total Test Time, Breakdown Not Available	1			1	
9. 60 Seconds Total Test Time, Breakdown Not Available	2				
10. All tests performed are engineering type tests, not production or incoming inspection oriented.	1			1	
11. Full test program not written to date, time undeterminable.	1				

F. Overhead Time, DC Test Program

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. 10 Seconds For Hardcopy Printout.				1	
2. 1%		1			
3. Undeterminable	6	2	2	3	1

G. Percentage of Total Test Program

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. 0%			1		
2. 4%				1	
3. 10%	1				
4. 20%	2		1		
5. 20 - 30%		1			
6. 70 - 80%	1			1	
7. Undeterminable	2	2		2	1

H. Differences Between Wafer and Final Package DC Tests

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Wafer level tests include a 25 V Stress Test which is not done at final package.	1				
2. Wafer level DC tests are closely monitored for indications of yield relating to process parameters. Final Package is strictly Go/NoGo.		1			
3. Wafer level tests are performed with wider guardbands.			1	1	
4. None.	1			1	1
5. Undeterminable.	1	2			
6. Not Applicable.	3			2	

I. Types of Failures

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Normal Process Related Failures.	2	2		3	
2. Leakage Current, Temperature Failures.	2	1			1
3. Undeterminable	1			1	
4. No Comment	1	1	1		

III. FUNCTIONAL TEST PROGRAM, PRODUCTION

A. Test Pattern

1. Method of Generation

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Functional Computer Simulation.	4	2		1	1
2. Manual, Line by Line.	1				
3. Manual, Line by Line, Generation in Tester Assembly Language.				2	
4. Half Simulation (Learn Mode).	1	1		1	
5. Gold Device (DUT operates in parallel to known good device.).			1		

2. Basis For the Order of uP Instruction Test Sequence

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Exercise every node. Verify operation of every instruction within specified timing requirement. Exercise adjacent nodes in Modular Approach. Satisfy large user requirements.	1		1		
2. Use a Modular Approach to verify device operation. Also utilizes test engineer's experience to generate an interactive type test.	1				
3. Modular Approach using worst case instruction and data pattern sequence.	3				
4. Modular Approach designed to represent worst case operation.		2		3	
5. Test pattern developed by device designer to represent worst case operation.	1	1		1	
6. Identify all data paths, all instruction operations.					1

3. Pattern Length

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. 1K				1	
2. 2K	1	3			
3. 2K Clock Cycles	1				
4. 5K				1	
5. 8K				1	
6. 12K	1				
7. 16K	1				
8. 7500					1
9. Program Incomplete To Date	1			1	
10. No Comment	1			1	

4. Pattern Sensitivity

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. RAM section sensitive to a CHECKERBOARD Pattern. Results are based upon a sample space of five devices.				1	
2. ALU				1	
3. None	6	3	1	2	1

5. Frequency of Testing Device Outputs

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Each Cycle, Each Pin	5	3		2	1
2. When Determinable Data Is Expected to Be Present.	1				
3. Not Every Pin, Every Clock Cycle. The Status of a Pin is Tested Based Upon the Test Engineers Judgement.	1		1		
4. Several Instructions are Executed, Pins of Interest Are Tested.				1	

B. FUNCTIONAL TEST CONDITIONS

1. Device Timing, Frequency

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. 1 MHz		1			
2. 2 MHz	1				
3. 3 MHz				1	
4. 4 MHz	1			1	
5. 100 MHz				1	
6. 500 MHz				1	
7. Minimum and Maxiumum Cycle Time.	1				
8. Maximum Cycle Time, Minimum Cycle Time, Each Extreme Tested With Guardband.	1	2	2		
9. Maximum	2				

2. AC Test Parameters

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Rise and Fall Times, Minimum Pulse Width, Access Times.	2	1		1	
2. Minimum Pulse Width's, Access Times.	4	2	1	3	1

3. Functional Test Execution Times and Overhead

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. 2 Seconds Total Test Time 1/2 Second Overhead.		1			
2. 3 Seconds Total Test Time Overhead Undeterminable.		1			
3. 3 Seconds Total Test Time 1 Second Overhead.				1	
4. 3.5 Seconds Total Test Time Overhead Undeterminable.				1	
5. 4 Seconds Total Test Time 1 Second Overhead.	1				
6. 5 Seconds Total Test Time Overhead Undeterminable.	1			1	
7. 10 Seconds Total Test Time Overhead Undeterminable.					1
8. 60 Seconds Total Test Time Overhead Undeterminable.	1				
9. 63 Seconds Total Test Time 60 Seconds Overhead.	1				
10. Program Not Completed to Date.	1	1			
11. No Comment.	1		1		

4. Error Analysis Information Available

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Results indicate what test, what uP instruction, what data pattern, what pin(s), although this information is not used in Go/NoGo testing.	5				
2. Only Pass/Fail status.	1				
3. Results indicate what test failed, what uP instruction, what data pattern, and the pin(s) involved.		2		1	
4. Results indicate the uP instruction involved in the failure but not expected data output or what pin(s) involved. The failing test is indicated.		1			
5. Results indicate what test failed and the data pattern involved. The instructions involved can be determined with a manual.				1	
6. Off-line analysis, the failing pin is not displayed.				1	
7. Only RAM section test results indicate what data pattern, input code and failing pin(s) status.				1	
8. The capability for indicating failing test, uP instruction, data pattern, and pin(s) involved exists although it is not used in the Go/NoGo situation.			1		
9. Parametric test portion indicates the test failed, functional portion indicates uP instruction which failed.					1

5. Types of Failures Discovered

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Normal, no pattern sensitivity found.	1				
2. Timing, logic error, temperature related failures.	1				
3. Parts are slow, do not meet timing specifications.	1				
4. Majority of failures are totally inoperative.	1				
5. Package devices are mainly functional failures.	1				
6. Normal type failures of DC and functional.		1			
7. Normal process related failures.				1	
8. Mostly functional failures.		1			
9. In DC mode, leakage current is the predominant failure mode. Most failures are parts that fail within first 15 instructions.					1
10. Have only tested small amount information inaccurate.				1	
11. Have not completed in-coming inspection program to date.	1				
12. Information not available.		1	1	1	

6. Differences Between Wafer and Final Package Functional Tests

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. No Multiple Probes on Wafer.	1				
2. Less Functional Tests at Wafer, to Identify Working Parts, do Speed Classification at Final Package.	2				
3. Final Package Tests Include More Extensive Timing and Voltage Corners To Classify Parts.		2			
4. Test Pattern Is the Same. Timing and Voltage Corners Are More Extensive to Classify Parts.				1	
5. Do No Perform Wafer Probe Except Under Special Circumstances. Test Would Be Different But Details Not Available.				1	
6. None.	1		1		1
7. Not Applicable.	2	1		1	

IV. Characterization Efforts

A. Parameters Characterized

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. All AC & DC Parameters.	1		1	3	1
2. All AC & DC Parameters Except Rise & Fall Times.	4	2		1	
3. All AC & DC Parameters Plus Additional Parameters Related to Process Control.	1				
4. All AC & DC Parameters, Except Rise & Fall Times, Data Patterns and Instruction Sequences.		1			

B. Temperature & Burn-in Conditions

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. -55°C to 125°C temperature, 6-7°C increments, life test evaluation performed in lieu of burn-in.	1		1		
2. At present ambient, expect to go to 80°C case temperature, life testing.	1				
3. Military: -55°C, -40°C, 0°C, 25°C, 100°C, 125°C. Commercial 0-70°C. Burn-in at 150°C 40 hours minimum, Ø1, Ø2 clocked, loaded outputs.	1				
4. -55°C, -30°C, Ambient, 85°C, 125°C, life testing at 125°C, 1000-2000 hours.	1				
5. 0°C & 70°C, will go to 85°C, possibly higher, no burn-in.	1				
6. 70°C, no burn-in.	1				1
7. Ambient, no burn-in.		1			
8. 0°C to 70°C, guard banded life test.		1			
9. -55°C to +125°C, burn in at 125°C, 48 hours, outputs loaded, Ø1,Ø2, clocked.		1			
10. -55, 30°C, Ambient, 85°C, +125°C, life test at 125°C, dynamic, 1000-2000 hours.				1	
11. -55, 0°C, 25°C, 70°C, +125°C life test at 125°C, dynamic, 5000 hours.				1	
12. 0°C, 70°C, 125°C, burn-in is static, power supplied, no pattern applied, outputs loaded.				1	
13. Temperature is -65°C to +200°C. Perform burn-in only if contract specifies. Have capability to perform all burn-in and environmental tests.				1	

C. Form of Characterization Data Log

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Statistical analysis, curves, extensive use of shmoo plots, occasionally log to disc or mag tape for off-line evaluation.	1				
2. Shmoo plots, cumulative and individual.	2				
3. Histograms, shmoo plots and statistical analysis.	2	3			
4. Number charts, might go to histograms in future. Are not performing extensive characterization to date.	1				
5. Tabular output and statistical analysis.				1	
6. Statistical analysis, curves, and extensive use of shmoo plots.				1	
7. Graphical pictures, histograms, statistical analysis, and some shmoo plots.				1	
8. Statistical analysis, histograms, and shmoo plots.			1		
9. Summary.				1	
10. Shmoo Plots.					1

D. Number of Devices Characterized

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. 6				1	
2. 10	1	1			
3. 12 - 24	1				
4. 20				1	
5. 40				1	
6. 50	1				
7. 100	1				
8. 500	1		1		
9. 1400		1			
10. Information Not Available	1	1		1	1

E. Department Responsible for Characterization

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Product Engineering	2	1	1		
2. Design Engineering	1				
3. Production & Design Engineering	1				
4. Manufacturing Engineering				1	
5. Electronic Design	1				
6. Components & Evaluation Department	1	1		1	
7. Operations Department				1	
8. Advanced Device Technology Department				1	
9. Production Test Group					1

F. Method of Processing Characterization Data

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Automatically	5	3	1	4	1
2. Manually	1				

V. PRODUCTION TESTING

A. Temperature Conditions

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Ambient	1	2		2	1
2. -55°C, +125°C Ambient	1			1	
3. 0-70°C, 125°C, Ambient	1				
4. 70°C		1			
5. Commercial parts 70°C, Military, Cold, Ambient, Hot	1		1		
6. Initial at 70°C, plan to reduce to Ambient.	2				
7. Wafer at Ambient, final package per MD STD 883, 5004, 5005, Class C.				1	

B. Burn-in Conditions

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. 125°C, dynamic, outputs loaded, performed as part of QA sampling, not normal portion of production test.		2			
2. 125°C, dynamic, outputs loaded.	1			1	
3. At customers request only, 125°C, static, outputs loaded.	3				
4. At customers request 125°C to 160°C, dynamic, outputs loaded.				1	
5. Performed as part of QA sampling, static mode, no pattern applied, outputs loaded, 125°C.				1	
6. Only performed if contract specifies. Have capability to do full military temperature range and dynamic type burn-in.				1	
7. Military only, 150°C, 40 Hours minimum, Ø1, Ø2, clocked, outputs loaded.	1		1		
8. No burn-in.	1				1
9. None at present.		1			

C. Datalog Format and Hardcopy

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. 6 Bins, hardcopy of Bin distribution obtained.	1				
2. Go/NoGo Testing, hard-copy of Bin count.	1			3	
3. Bin classification, hardcopy of Bin count.	4	3	1		1
4. Bin classification: Pass, Fail, DC, Fail Functional, no hardcopy obtained.				1	

D. Percentage of Devices Screened

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
100%	6	3	1	4	1

E. Department Responsible for Production Testing

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Production Operations	1		1		
2. Production Control	1				
3. Production Testing	1				1
4. Incoming Inspection	2	1		1	
5. Product Engineering	1	1			
6. Manufacturing Engineering		1			
7. Bipolar Microprocessor Department				1	
8. Quality Assurance				1	
9. Operations Department				1	

F. Use of 38510 Specification

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Are testing to what the 8080, 38510 is anticipated to contain.	1				
2. Are testing to in-house version of 38510, specification for 8080, written by Quality Assurance Department.	1				
3. None--an in-house specification is used which parallels a class C military specification.	1				
4. Are using 38510 slash sheet.		1			
5. Use in-house version of 38510.				1	
6. Will generate in-house version.					1
7. None	3	2		1	

G. Types of Functional Failures

RESPONSE	DEVICE				
	8080	6800	8008	2901	1802
1. Totally inoperative parts, functionally.	1				
2. Normal process and packaged related failures.	1			1	
3. Package devices are predominantly functional failures.	1	1			
4. DC tests are predominantly leakage failures; functional failures are parts that wholly inoperative.					1
5. No data available.	2	2		2	
7. No comment.	1		1	1	

VI. Types of Problems Encountered in Testing Microprocessors

The problems being encountered in the testing of microprocessors do not reflect upon an individual microprocessor but rather the concept of testing a central processing unit intergrated on one LSI chip. The following is a summary of all comments received.

The range of problems encountered in testing microprocessors is best presented by establishing a categorical list derived from both manufacturer's and user's. These problems are:

1. None
2. Time Involved
3. Money Expenditures
4. Knowledge of Device
5. User Understanding of Device Operation and Application
6. Appreciation of Total Efforts
7. Test Equipment
8. Accurate Technical, Information on Device

Item 1

Four (4) interviewers stated that no one area of testing presented unusual problems or problems considered to approach the limits of present test technology.

Item 2

The time factor encompasses all aspects of testing microprocessors. This includes test time, preparation of the total test program (production and characterization) and analysis of test results.

Item 3

The amount of monies involved in testing includes capital expenditures for equipment and program development.

Item 4 & 8

Before the device can properly be tested complete knowledge of the microprocessor is essential. Users are of the opinion that this effort is hindered by a lack of adequate technical information concerning device operation including accurate timing and instruction operations.

Item 5 & 6

User's who test microprocessors for outside companies and in-house departments are finding that those people responsible for management of these tasks do not appreciate the total effort of testing. The device application is often not fully stated, nor, is the complexity of the test hardware and software requirements understood.

Item 7

Only one interviewer indicated that presently available test equipment posed a problem in testing. This comment concerned the speed of test equipment with respect to test time. However, the time involved in preparing test programs as viewed from an ease of program development standpoint and the actual test time due to test system overhead requirements can also be considered a valid criticism under this heading.

Item 8

All user's except one stated that existing technical information about specific device operation is not sufficient. Additional information is needed which will accurately define total device operation in terms of timing and instruction execution.

VII. Recommendations for User Testing

Recommendations for user testing were found to touch upon just about every aspect of testing, ranging from determining the extent of testing required to tips on test program structure.

Collating all the comments gathered results in the following summary.

First, determine the nature of the MPU testing problem from consideration of such factors as application, reliability requirements, and money available.

The results of this study should then indicate the capabilities of the test system to be used including hardware/software trade-offs, ease of use, DC and AC test capabilities and provision of test result analysis. Test equipment possibilities also include the end product system or a uP development system, in addition to the option of designing a system in-house. Another alternative is to not buy test equipment but use a testing laboratory instead.

Overall test philosophy should be defined as early in the process as possible with the key objective of being as thorough as possible within the confines of times, money, and manpower available.

The actual test program should retain the objective of thoroughness

by functional verification of each module of the device followed by an interactive type test to insure that there is no module-to-module destruction of data. If possible, the test scheme should be designed such that modifications are easily installed at some later time.

Be prepared for the time and money expenditures that will be necessary for the design and implementation of a thorough testing plan. Additional considerations to be included are resources for providing facilities for the test equipment and personnel to operate and maintain these items.

VIII. DETECTED PROBLEMS

The following instruction sequence sensitivities, module weaknesses, or failures were either described during the study conducted and/or detected by Macrodata Corporation during its characterization of the device prior to this contract. All problems described that were verified by Macrodata have been reported to the manufacturer(s). In all cases, parts manufactured after reporting the problem areas did not exhibit these characteristics.

Device		Problem	Verified By Macrodata
8080	1.	It was detected on some devices that the Ø1 clock cross couples noise that exceeded the threshold level on the HLDA input 1 ns.	Yes
	2.	When running a test similar to the one described for the program counter, certain devices would fail to respond to a reset pulse every time.	Yes
	3.	When performing a test similar to the test described for the register array, some devices showed a sensitivity to H⟶B and H⟶D transfers when the 5 MSB's, in the data sequence were all 1's.	Yes
	4.	Not all manufactured parts operate exactly alike. One 8080 will not execute a program	Yes

Device	Problem	Verified By Macrodata
	instruction identical to that of another manufacturers part. The main differences lies in the execution of arithmetic instructions in that status flag operation is not always identical from one manufacturer to another. This difference between parts produced by different manufacturers still is present.	
8008	1. It has been detected that if an instruction sequence which causes the stack registers to perform push and pop operation is repeated multiple times, the device fails to operate. This was not detected on any other area within the device.	Yes
	2. Some of the fail devices operated correctly for a short period of time, but after a period of time the devices would fail to operate. This period of time was around 5 to 10 seconds. Once the device failed, it would not ever become operational again, even if power was removed and reapplied. The device became a total failure.	No

Device		Problem	Verified By Macrodata
2901	1.	The "A" output latch for the RAM would change state on some devices that were operated at a slightly elevated temperature and voltage (still within specification). The problem was detected by shifting a binary pattern of 111Ø into the RAM and holding this pattern and then addressing another location with a pattern of 1111, but not clocking this pattern into the latch. The parts that exhibited this sensitivity showed that the output latch value changed from a 111Ø pattern to a 1111 pattern.	Yes
	2.	Some parts that have been tested showed a sensitivity to a CHECKERBOARD pattern on the RAM.	No
	3.	Not all parts will operate at their rated speed. Newer versions of these devices do not exhibit this problem.	Yes
	4.	It has been reported that the ALU section has shown some kind of sensitivity to either data, instruction, or a combination of the two. This sequence was not defined, so this failure could not be verified.	No

Device	Problem	Verified By Macrodata
6800 & 1802	No problems were reported on these parts other than normal manufacturing process problems, which were detected by the manufacturers. User's reported no extensive testing on either of these devices, therefore, no errors were reported by them.	-

IX. TEST EQUIPMENT

DC Requirement

1. Voltage/Current Force Function

Voltage Force Range = 0 to ± 15 Volts

It is recommended that there should be two ranges within this total range.

Accuracy > .7% of Full Scale

Current Force Range = 500 pa to 50 ma

This range should be divided into at least three ranges.

Accuracy > .7% of Full Scale

2. Voltage/Current Measurement

Voltage Measurement = +15 Volts to -5 Volts

Recommended at least two ranges.

Accuracy > .3% Full Scale

Current Measurement = 500 pa to 300 ma

Recommended ranges: 2 uA Full Scale, 20 uA, 200 uA, 2 ma, 20 ma, 300 ma.

Accuracy - 0.5% Full Scale

Power Supplies

Three Device Bias Supplies Plus Ground

Voltage Range = ± 15 Volts

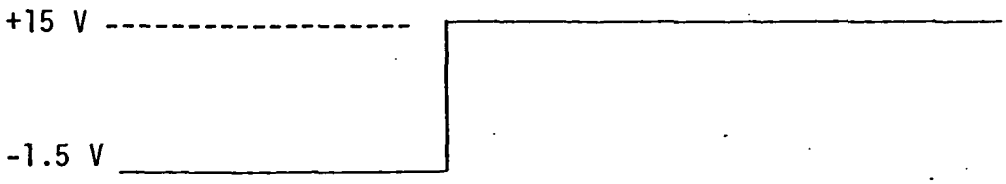
Current Range = 300 ma, Minimum

Accuracy = 0.2% of Set Voltage for Testing

AC Voltage Requirements

Logical input voltage swing: +15 Volts to -1.5 Volts, Maximum

This voltage should be variable in 10 mV increments throughout the range.



Logical output voltage detection: +15 Volts to -1.5 Volts, Maximum

This voltage should be variable in 10 mV increments throughout the range.

It is recommended that the output sampling circuit be able to detect both a VOL and VOH voltage simultaneously. This will allow for a functional test measurement in one pass.

Timing Requirements

It should be noted that all timing edges produced for input or output timing are required to be synchronized to one master clock generator.

Input

Clock Frequency = 10 MHz to DC.

Minimum Clock Pulse Width = 20 ns.

Clocks required = 2 Minimum - Device Clocks
1 Minimum - Data Bus Clock
1 Minimum - Data Bus I/O Control
2 Minimum - Control Signals for Setup and Hold Time Measurements

Timing edge should be capable of being variable in 1 ns increments.

Output

Minimum of one output comparator strobe with both edges variable in 1 ns increments. Strobe positioning should vary over the complete clock input cycle.

With only one comparator strobe it will be required to make more than one functional test on the devices. This is required to verify all output timing of the particular devices.

Tester Configuration

To generate the basic patterns to test the microprocessor that have been previously described the following tester would be required. Figure 21 illustrates the block diagram of a basic tester.

Mass storage, such as disc, or extended RAM or shift register memory is used to store total test patterns.

A high-speed storage media consisting of high-speed RAM or shift register memory (minimum of 1K deep X 48 wide) is used to hold portions of total test pattern. Overlay of this memory is required from the mass storage medium.

The Pattern Control and Sequence control logic allows repetition of the same test pattern or series of patterns to reduce total test pattern size and enable performance of tests that would otherwise not be possible due

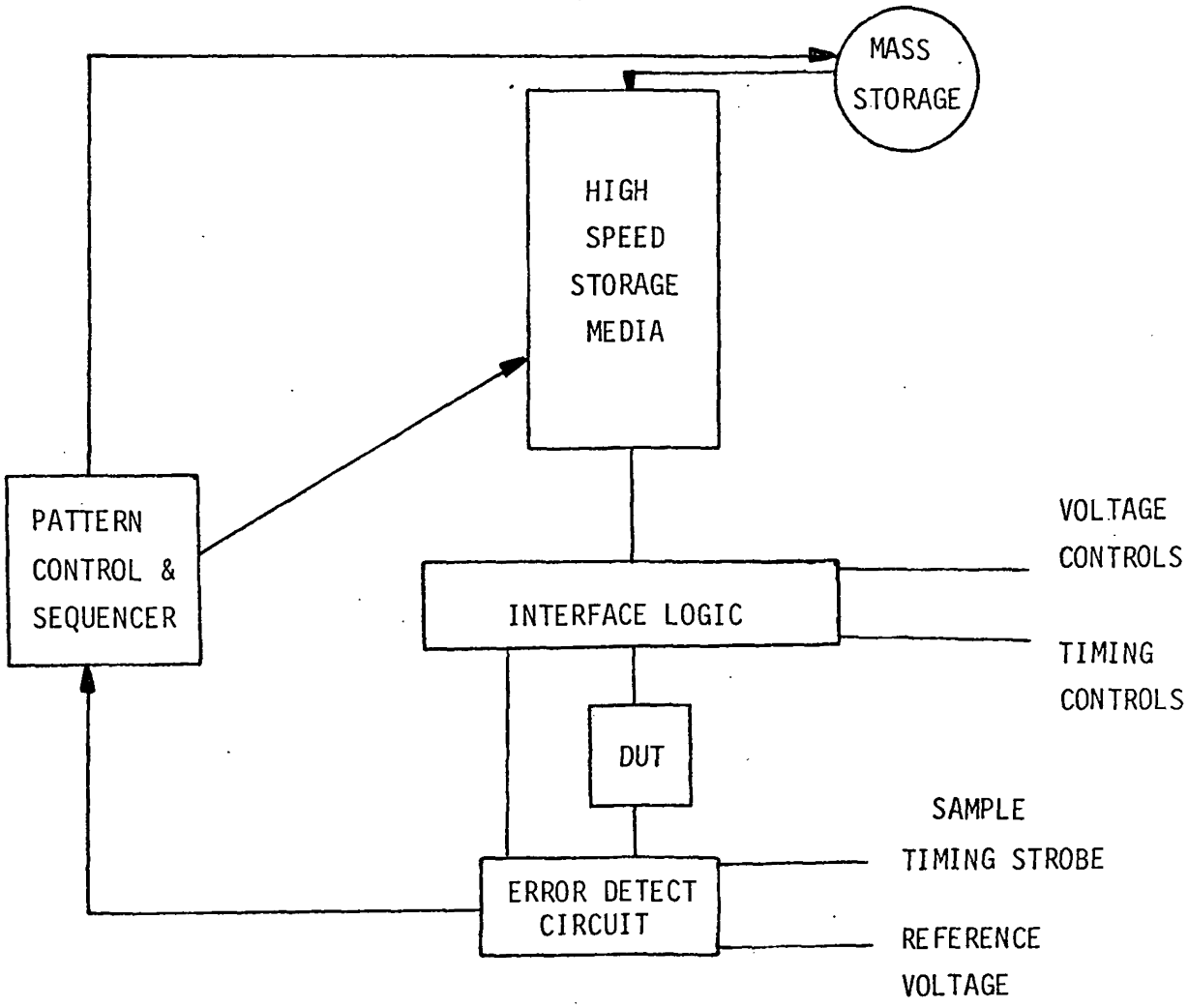


Figure 21: Basic Tester Block Diagram

to size of test pattern. The capability to allow real time error interrupt is also provided.

The Interface logic provides the necessary formatting of signals to DUT including voltage and timing levels and signal format in addition to the capability of holding a test pattern on the device while the high-speed storage media is overlayed by the mass memory.

The Error Detect circuit compares the output of the device under test with the previously stored output response pattern, alerting the pattern control and sequencer module of error conditions.

This basic system can be developed by a company whose testing requirements necessitate the use of such a system. Because of the design variations for the different microprocessors, a project of this type can become costly and time consuming. Since at least three test equipment manufacturers produce equipment that can perform the described test, it is recommended that a company consider purchasing this type of system from a commercially available supplier. Companies that produce this type of equipment are:

Fairchild Systems	Macrodata Corporation	Tektronic Systems
San Jose, CA	Woodland Hills, CA	Beaverton, OR

Software Requirements

The software requirements for the test system described in the preceding

section can be itemized as follows:

1. Disc Operating System
2. Test System Executive Program
3. Microprocessor Test Pattern Development Program

Discussion

In order to efficiently store the large quantities of test data necessary to execute the tests previously described, a magnetic disc based computer system was chosen. The use of the disc requires a sophisticated computer program to control storing and retrieval of information to and from the disc. Programs of this type are available from the manufacturer of the computer chosen for use and can be incorporated in the total system software by the manufacturer of the test system. The user also has the option of designing his own program. The test system Executive Program is a custom computer program designed and developed specifically by the manufacturer of the test equipment. The elements of such a program are many. First, a test system language must be developed to allow the user to easily develop test programs for a wide variety of devices. Additional necessary programs are Editor, List, and Assembly programs.

An Editor program is one which allows the user to modify existing programs in source language. List programs output the entire source or object code contents of a test program to a peripheral medium (line printer or video terminal) to allow examination of the contents of a program. The

Assembly Program converts the source statements of a test program to a binary object code which is understandable by the computer. An Executive Program must be able to initiate to the different modules of the test system for storage and retrieval of information in addition to controlling their activities.

The Microprocessor Test Pattern Development Program is designed to simplify the development of the test patterns previously described. The most desirable and accurate form of this program would be one which would completely simulate the microprocessor from an input string consisting of mnemonics and data patterns. The total output of such a program is a clock cycle by clock cycle definition of all input and output pins of the microprocessor in response to the defined input instructions. This program also includes test system control data such as when to input microprocessor instructions and data patterns and also when to test the output pins. If the test system cannot accomodate testing of all device outputs simultaneously, several versions of the test pattern are necessary to completely verify each device pin.

X. QUALIFICATION TEST VERSUS SCREENING TEST

The test required for qualification of the device should include:

1. All functional and DC tests as previously described.

2. Each functional module test should be verified over the complete voltage operating range of the device. The best means to do this is to generate shmoo plots which plot voltage (VDD, VBB, etc.) versus timing parameters and also voltage versus voltage.

3. All AC timings specified in the manufacturers data sheet should be verified. Again, the best method is shmooing voltage versus each individual timing parameter and other voltages. Voltage should be varied over the complete specification range.

4. Qualification should consist of testing the device over the manufacturers full temperature range. Recommended temperatures are +125, +70, +25, Ø, and -55°C. All manufacturers do not perform this test except for normal AQL sampling.

5. An extensive burn-in program should be performed since manufacturers only perform this upon specific request from a customer. A recommended burn-in program should be at least 160 hours at 125°C with elevated voltages. Also, random dynamic signals should be applied continuously during the burn-in cycle. This is only a recommendation since further

performed on the subject of microprocessor burn-in procedures.

6. All DC specifications on the device should also be varied over the complete voltage range of the device. Again, using the shmoo plot method is recommended.

A 100% screening test should be based on the results from the qualification test. This test should include:

1. Testing each functional module as previously described at upper and lower power supply limits and all combinations.

2. Testing all manufactures DC parameters.

3. If the device is to be used over the complete military temperature range, the test should be performed at +125, +25, and -70°C.

4. A burn-in conditioning should be conducted as previously recommended.

5. Only critical timing should be verified in order to reduce test time. These should include minimum clock pulse width, clock frequency, and access time.

ATTACHMENT 1

8080A FUNCTIONAL PIN DEFINITION

The following describes the function of all of the 8080A I/O pins. Several of the descriptions refer to internal timing periods.

A_{15}-A_0 (output three-state)
ADDRESS BUS; the address bus provides the address to memory (up to 64K 8-bit words) or denotes the I/O device number for up to 256 input and 256 output devices. A_0 is the least significant address bit.

D_7-D_0 (input/output three-state)
DATA BUS; the data bus provides bi-directional communication between the CPU, memory, and I/O devices for instructions and data transfers. Also, during the first clock cycle of each machine cycle, the 8080A outputs a status word on the data bus that describes the current machine cycle. D_0 is the least significant bit.

SYNC (output)
SYNCHRONIZING SIGNAL; the SYNC pin provides a signal to indicate the beginning of each machine cycle.

DBIN (output)
DATA BUS IN; the DBIN signal indicates to external circuits that the data bus is in the input mode. This signal should be used to enable the gating of data onto the 8080A data bus from memory or I/O.

READY (input)
READY; the READY signal indicates to the 8080A that valid memory or input data is available on the 8080A data bus. This signal is used to synchronize the CPU with slower memory or I/O devices. If after sending an address out the 8080A does not receive a READY input, the 8080A will enter a WAIT state for as long as the READY line is low. READY can also be used to single step the CPU.

WAIT (output)
WAIT; the WAIT signal acknowledges that the CPU is in a WAIT state.

$\overline{WR}$ (output)
WRITE; the $\overline{WR}$ signal is used for memory WRITE or I/O output control. The data on the data bus is stable while the $\overline{WR}$ signal is active low ($\overline{WR}$ = 0).

HOLD (input)
HOLD; the HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080A address and data bus as soon as the 8080A has completed its use of these buses for the current machine cycle. It is recognized under the following conditions:

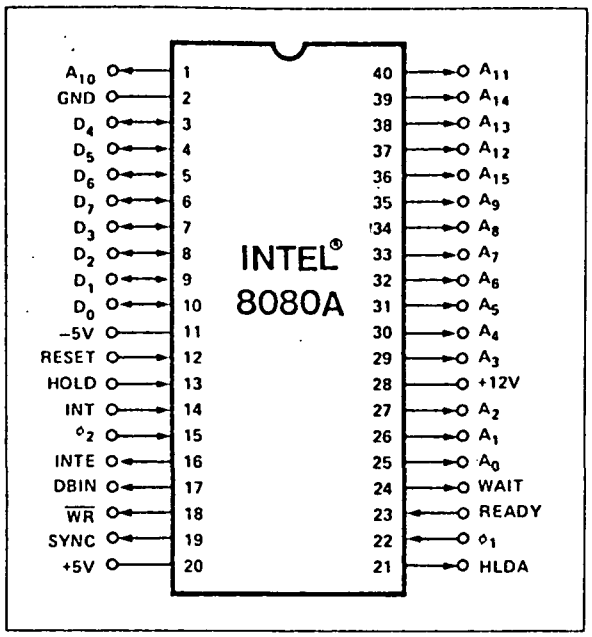
- the CPU is in the HALT state.
- the CPU is in the T2 or TW state and the READY signal is active.

As a result of entering the HOLD state the CPU ADDRESS BUS (A_{15}-A_0) and DATA BUS (D_7-D_0) will be in their high impedance state. The CPU acknowledges its state with the HOLD ACKNOWLEDGE (HLDA) pin.

HLDA (output)
HOLD ACKNOWLEDGE; the HLDA signal appears in response to the HOLD signal and indicates that the data and address bus will go to the high impedance state. The HLDA signal begins at:

- T3 for READ memory or input.
- The Clock Period following T3 for WRITE memory or OUTPUT operation.

In either case, the HLDA signal appears after the rising edge of ϕ_1 and high impedance occurs after the rising edge of ϕ_2.



Pin Configuration

INTE (output)
INTERRUPT ENABLE; indicates the content of the internal interrupt enable flip/flop. This flip/flop may be set or reset by the Enable and Disable Interrupt instructions and inhibits interrupts from being accepted by the CPU when it is reset. It is automatically reset (disabling further interrupts) at time T1 of the instruction fetch cycle (M1) when an interrupt is accepted and is also reset by the RESET signal.

INT (input)
INTERRUPT REQUEST; the CPU recognizes an interrupt request on this line at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the Interrupt Enable flip/flop is reset it will not honor the request.

RESET (input)[1]
RESET; while the RESET signal is activated, the content of the program counter is cleared. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip/flops are also reset. Note that the flags, accumulator, stack pointer, and registers are not cleared.

V_{SS}	Ground Reference.
V_{DD}	+12 ± 5% Volts.
V_{CC}	+5 ± 5% Volts.
V_{BB}	-5 ±5% Volts (substrate bias).
ϕ_1, ϕ_2	2 externally supplied clock phases. (non TTL compatible)

8080A FUNCTIONAL PIN DEFINITION

The following describes the function of all of the 8080A I/O pins. Several of the descriptions refer to internal timing periods.

A_{15}-A_0 (output three-state)
ADDRESS BUS; the address bus provides the address to memory (up to 64K 8-bit words) or denotes the I/O device number for up to 256 input and 256 output devices. A_0 is the least significant address bit.

D_7-D_0 (input/output three-state)
DATA BUS; the data bus provides bi-directional communication between the CPU, memory, and I/O devices for instructions and data transfers. Also, during the first clock cycle of each machine cycle, the 8080A outputs a status word on the data bus that describes the current machine cycle. D_0 is the least significant bit.

SYNC (output)
SYNCHRONIZING SIGNAL; the SYNC pin provides a signal to indicate the beginning of each machine cycle.

DBIN (output)
DATA BUS IN; the DBIN signal indicates to external circuits that the data bus is in the input mode. This signal should be used to enable the gating of data onto the 8080A data bus from memory or I/O.

READY (input)
READY; the READY signal indicates to the 8080A that valid memory or input data is available on the 8080A data bus. This signal is used to synchronize the CPU with slower memory or I/O devices. If after sending an address out the 8080A does not receive a READY input, the 8080A will enter a WAIT state for as long as the READY line is low. READY can also be used to single step the CPU.

WAIT (output)
WAIT; the WAIT signal acknowledges that the CPU is in a WAIT state.

$\overline{WR}$ (output)
WRITE; the $\overline{WR}$ signal is used for memory WRITE or I/O output control. The data on the data bus is stable while the $\overline{WR}$ signal is active low ($\overline{WR}$ = 0).

HOLD (input)
HOLD; the HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080A address and data bus as soon as the 8080A has completed its use of these buses for the current machine cycle. It is recognized under the following conditions:

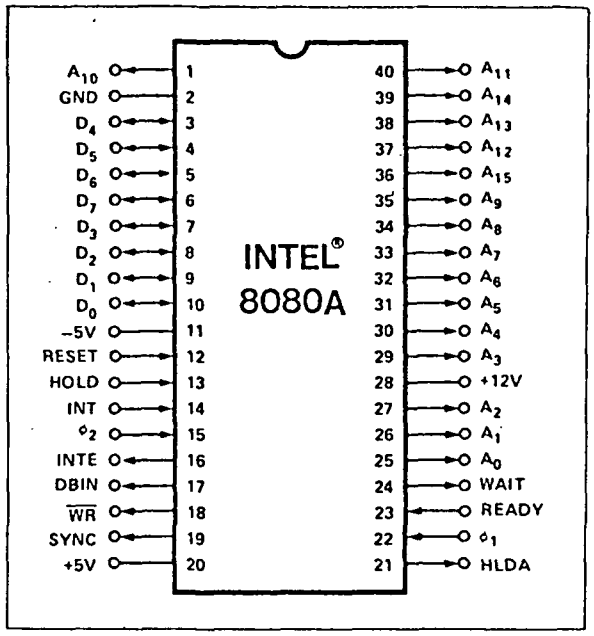
- the CPU is in the HALT state.
- the CPU is in the T2 or TW state and the READY signal is active.

As a result of entering the HOLD state the CPU ADDRESS BUS (A_{15}-A_0) and DATA BUS (D_7-D_0) will be in their high impedance state. The CPU acknowledges its state with the HOLD ACKNOWLEDGE (HLDA) pin.

HLDA (output)
HOLD ACKNOWLEDGE; the HLDA signal appears in response to the HOLD signal and indicates that the data and address bus will go to the high impedance state. The HLDA signal begins at:

- T3 for READ memory or input.
- The Clock Period following T3 for WRITE memory or OUTPUT operation.

In either case, the HLDA signal appears after the rising edge of ϕ_1 and high impedance occurs after the rising edge of ϕ_2.



Pin Configuration

INTE (output)
INTERRUPT ENABLE; indicates the content of the internal interrupt enable flip/flop. This flip/flop may be set or reset by the Enable and Disable Interrupt instructions and inhibits interrupts from being accepted by the CPU when it is reset. It is automatically reset (disabling further interrupts) at time T1 of the instruction fetch cycle (M1) when an interrupt is accepted and is also reset by the RESET signal.

INT (input)
INTERRUPT REQUEST; the CPU recognizes an interrupt request on this line at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the Interrupt Enable flip/flop is reset it will not honor the request.

RESET (input)[1]
RESET; while the RESET signal is activated, the content of the program counter is cleared. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip/flops are also reset. Note that the flags, accumulator, stack pointer, and registers are not cleared.

V_{SS}	Ground Reference.
V_{DD}	+12 ± 5% Volts.
V_{CC}	+5 ± 5% Volts.
V_{BB}	-5 ±5% Volts (substrate bias).
ϕ_1, ϕ_2	2 externally supplied clock phases. (non TTL compatible)

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	0°C to +70° C
Storage Temperature	-65°C to +150°C
All Input or Output Voltages With Respect to V_{BB}	-0.3V to +20V
V_{CC}, V_{DD} and V_{SS} With Respect to V_{BB}	-0.3V to +20V
Power Dissipation	1.5W

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, Unless Otherwise Noted.

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	V_{SS}-1		V_{SS}+0.8	V	
V_{IHC}	Clock Input High Voltage	9.0		V_{DD}+1	V	
V_{IL}	Input Low Voltage	V_{SS}-1		V_{SS}+0.8	V	
V_{IH}	Input High Voltage	3.3		V_{CC}+1	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 1.9mA on all outputs, I_{OH} = -150μA.
V_{OH}	Output High Voltage	3.7			V	
$I_{DD(AV)}$	Avg. Power Supply Current (V_{DD})		40	70	mA	Operation T_{CY} = .48 μsec
$I_{CC(AV)}$	Avg. Power Supply Current (V_{CC})		60	80	mA	
$I_{BB(AV)}$	Avg. Power Supply Current (V_{BB})		.01	1	mA	
I_{IL}	Input Leakage			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{CL}	Clock Leakage			±10	μA	$V_{SS} \leq V_{CLOCK} \leq V_{DD}$
I_{DL} [2]	Data Bus Leakage in Input Mode			-100 -2.0	μA mA	$V_{SS} \leq V_{IN} \leq V_{SS}$ +0.8V V_{SS} +0.8V $\leq V_{IN} \leq V_{CC}$
I_{FL}	Address and Data Bus Leakage During HOLD			+10 -100	μA	$V_{ADDR/DATA} = V_{CC}$ $V_{ADDR/DATA} = V_{SS}$ + 0.45V

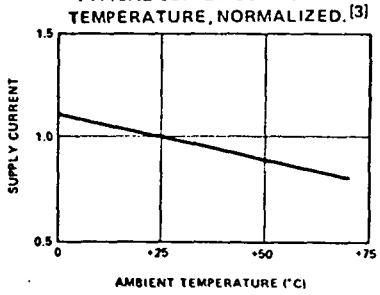
CAPACITANCE

T_A = 25°C $V_{CC} = V_{DD} = V_{SS}$ = 0V, V_{BB} = -5V

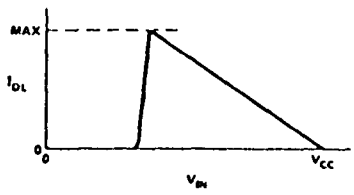
Symbol	Parameter	Typ.	Max.	Unit	Test Condition
C_ϕ	Clock Capacitance	17	25	pf	f_c = 1 MHz
C_{IN}	Input Capacitance	6	10	pf	Unmeasured Pins
C_{OUT}	Output Capacitance	10	20	pf	Returned to V_{SS}

NOTES:
1. The RESET signal must be active for a minimum of 3 clock cycles.
2. When DBIN is high and $V_{IN} > V_{IH}$ an internal active pull up will be switched onto the Data Bus.
3. ΔI supply / ΔT_A = -0.45%/°C.

TYPICAL SUPPLY CURRENT VS. TEMPERATURE, NORMALIZED.[3]



DATA BUS CHARACTERISTIC DURING DBIN



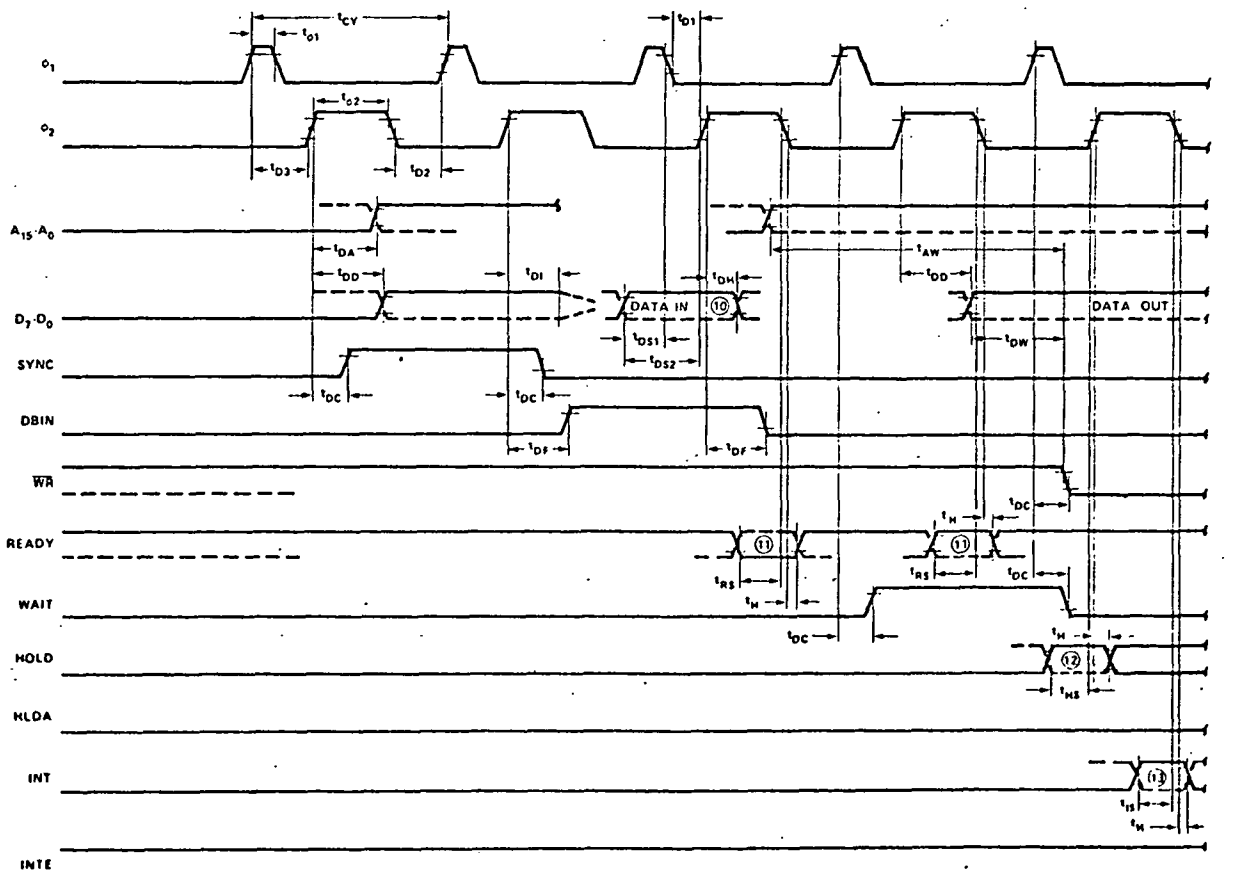
A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, Unless Otherwise Noted

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{CY}[3]	Clock Period	0.48	2.0	μsec	
t_r, t_f	Clock Rise and Fall Time	0	50	nsec	
$t_{\phi 1}$	ϕ_1 Pulse Width	60		nsec	
$t_{\phi 2}$	ϕ_2 Pulse Width	220		nsec	
t_{D1}	Delay ϕ_1 to ϕ_2	0		nsec	
t_{D2}	Delay ϕ_2 to ϕ_1	70		nsec	
t_{D3}	Delay ϕ_1 to ϕ_2 Leading Edges	80		nsec	
t_{DA}[2]	Address Output Delay From ϕ_2		200	nsec	C_L = 100pf
t_{DD}[2]	Data Output Delay From ϕ_2		220	nsec	C_L = 100pf
t_{DC}[2]	Signal Output Delay From ϕ_1, or ϕ_2 (SYNC, $\overline{WR}$,WAIT,HLDA)		120	nsec	C_L = 50pf
t_{DF}[2]	DBIN Delay From ϕ_2	25	140	nsec	C_L = 50pf
t_{DI}[1]	Delay for Input Bus to Enter Input Mode		t_{DF}	nsec	
t_{DS1}	Data Setup Time During ϕ_1 and DBIN	30		nsec	

TIMING WAVEFORMS [14]

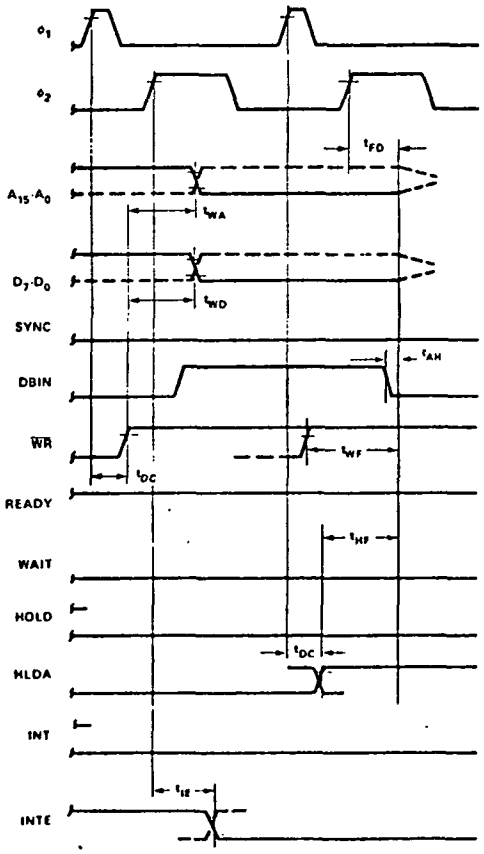
(Note: Timing measurements are made at the following reference voltages: CLOCK "1" = 8.0V "0" = 1.0V; INPUTS "1" = 3.3V, "0" = 0.8V; OUTPUTS "1" = 2.0V, "0" = 0.8V.)



A.C. CHARACTERISTICS (Continued)

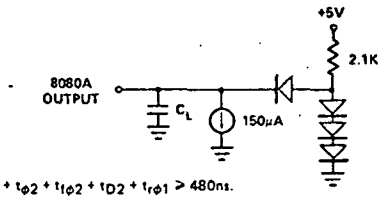
T_A = 0°C to 70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, Unless Otherwise Noted

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{DS2}	Data Setup Time to ϕ_2 During DBIN	150		nsec	
t_{DH} [1]	Data Hold Time From ϕ_2 During DBIN	[1]		nsec	
t_{IE} [2]	INTE Output Delay From ϕ_2		200	nsec	C_L = 50pf
t_{RS}	READY Setup Time During ϕ_2	120		nsec	
t_{HS}	HOLD Setup Time to ϕ_2	140		nsec	
t_{IS}	INT Setup Time During ϕ_2 (During ϕ_1 in Halt Mode)	120		nsec	
t_H	Hold Time From ϕ_2 (READY, INT, HOLD)	0		nsec	
t_{FD}	Delay to Float During Hold (Address and Data Bus)		120	nsec	
t_{AW} [2]	Address Stable Prior to $\overline{WR}$	[5]		nsec	C_L = 100pf: Address, Data C_L = 50pf: $\overline{WR}$, HLDA, DBIN
t_{DW} [2]	Output Data Stable Prior to $\overline{WR}$	[6]		nsec	
t_{WD} [2]	Output Data Stable From $\overline{WR}$	[7]		nsec	
t_{WA} [2]	Address Stable From $\overline{WR}$	[7]		nsec	
t_{HF} [2]	HLDA to Float Delay	[8]		nsec	
t_{WF} [2]	$\overline{WR}$ to Float Delay	[9]		nsec	
t_{AH} [2]	Address Hold Time After DBIN During HLDA	-20		nsec	



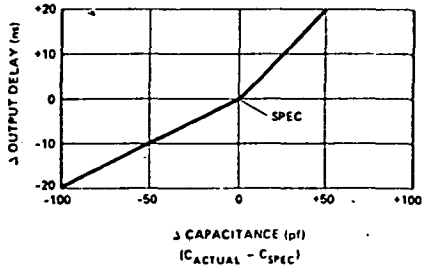
NOTES:

1. Data input should be enabled with DBIN status. No bus conflict can then occur and data hold time is assured. t_{DH} = 50 ns or t_{DF}, whichever is less.
2. Load Circuit.



3. $t_{CY} = t_{D3} + t_{r\phi2} + t_{\phi2} + t_{f\phi2} + t_{D2} + t_{r\phi1} \geq 480$ns.

TYPICAL Δ OUTPUT DELAY VS. Δ CAPACITANCE



4. The following are relevant when interfacing the 8080A to devices having V_{IH} = 3.3V:
   a) Maximum output rise time from .8V to 3.3V = 100ns @ C_L = SPEC.
   b) Output delay when measured to 3.0V = SPEC +60ns @ C_L = SPEC.
   c) If $C_L \neq$ SPEC, add .6ns/pF if $C_L > C_{SPEC}$, subtract .3ns/pF (from modified delay) if $C_L < C_{SPEC}$.
5. $t_{AW} = 2\,t_{CY} - t_{D3} - t_{r\phi2} - 140$nsec.
6. $t_{DW} = t_{CY} - t_{D3} - t_{r\phi2} - 170$nsec.
7. If not HLDA, $t_{WD} = t_{WA} = t_{D3} + t_{r\phi2} + 10$ns. If HLDA, $t_{WD} = t_{WA} = t_{WF}$.
8. $t_{HF} = t_{D3} + t_{r\phi2} - 50$ns.
9. $t_{WF} = t_{D3} + t_{r\phi2} - 10$ns
10. Data in must be stable for this period during DBIN · T_3. Both t_{DS1} and t_{DS2} must be satisfied.
11. Ready signal must be stable for this period during T_2 or T_W. (Must be externally synchronized.)
12. Hold signal must be stable for this period during T_2 or T_W when entering hold mode, and during T_3, T_4, T_5 and T_{WH} when in hold mode. (External synchronization is not required.)
13. Interrupt signal must be stable during this period of the last clock cycle of any instruction in order to be recognized on the following instruction. (External synchronization is not required.)
14. *This timing diagram shows timing relationships only; it does not represent any specific machine cycle*

INSTRUCTION SET

The accumulator group instructions include arithmetic and logical operators with direct, indirect, and immediate addressing modes.

Move, load, and store instruction groups provide the ability to move either 8 or 16 bits of data between memory, the six working registers and the accumulator using direct, indirect, and immediate addressing modes.

The ability to branch to different portions of the program is provided with jump, jump conditional, and computed jumps. Also the ability to call to and return from subroutines is provided both conditionally and unconditionally. The RESTART (or single byte call instruction) is useful for interrupt vector operation.

Double precision operators such as stack manipulation and double add instructions extend both the arithmetic and interrupt handling capability of the 8080A. The ability to increment and decrement memory, the six general registers and the accumulator is provided as well as extended increment and decrement instructions to operate on the register pairs and stack pointer. Further capability is provided by the ability to rotate the accumulator left or right through or around the carry bit.

Input and output may be accomplished using memory addresses as I/O ports or the directly addressed I/O provided for in the 8080A instruction set.

The following special instruction group completes the 8080A instruction set: the NOP instruction, HALT to stop processor execution and the DAA instructions provide decimal arithmetic capability. STC allows the carry flag to be directly set, and the CMC instruction allows it to be complemented. CMA complements the contents of the accumulator and XCHG exchanges the contents of two 16-bit register pairs directly.

Data and Instruction Formats

Data in the 8080A is stored in the form of 8-bit binary integers. All data transfers to the system data bus will be in the same format.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

DATA WORD

The program instructions may be one, two, or three bytes in length. Multiple byte instructions must be stored in successive words in program memory. The instruction formats then depend on the particular operation executed.

One Byte Instructions

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	OP CODE

TYPICAL INSTRUCTIONS

Register to register, memory reference, arithmetic or logical, rotate, return, push, pop, enable or disable Interrupt instructions

Two Byte Instructions

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	OP CODE
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	OPERAND

Immediate mode or I/O instructions

Three Byte Instructions

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	OP CODE
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	LOW ADDRESS OR OPERAND 1
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	HIGH ADDRESS OR OPERAND 2

Jump, call or direct load and store instructions

For the 8080A a logic "1" is defined as a high level and a logic "0" is defined as a low level.

INSTRUCTION SET

Summary of Processor Instructions

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
MOV r1, r2	Move register to register	0	1	D	D	D	S	S	S	5
MOV M, r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r, M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No operation	0	0	0	0	0	0	0	0	4

NOTES: 1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

8008/8008-1

EIGHT-BIT MICROPROCESSOR

- Instruction Cycle Time — 12.5 μs with 8008-1 or 20 μs with 8008
- Directly addresses 16K x 8 bits of memory (RAM, ROM, or S.R.)
- Interrupt Capability
- 48 Instructions, Data Oriented
- Address stack contains eight 14-bit registers (including program counter) which permit nesting of subroutines up to seven levels

The 8008 is a single chip MOS 8-bit parallel central processor unit for the MCS-8 microcomputer system.

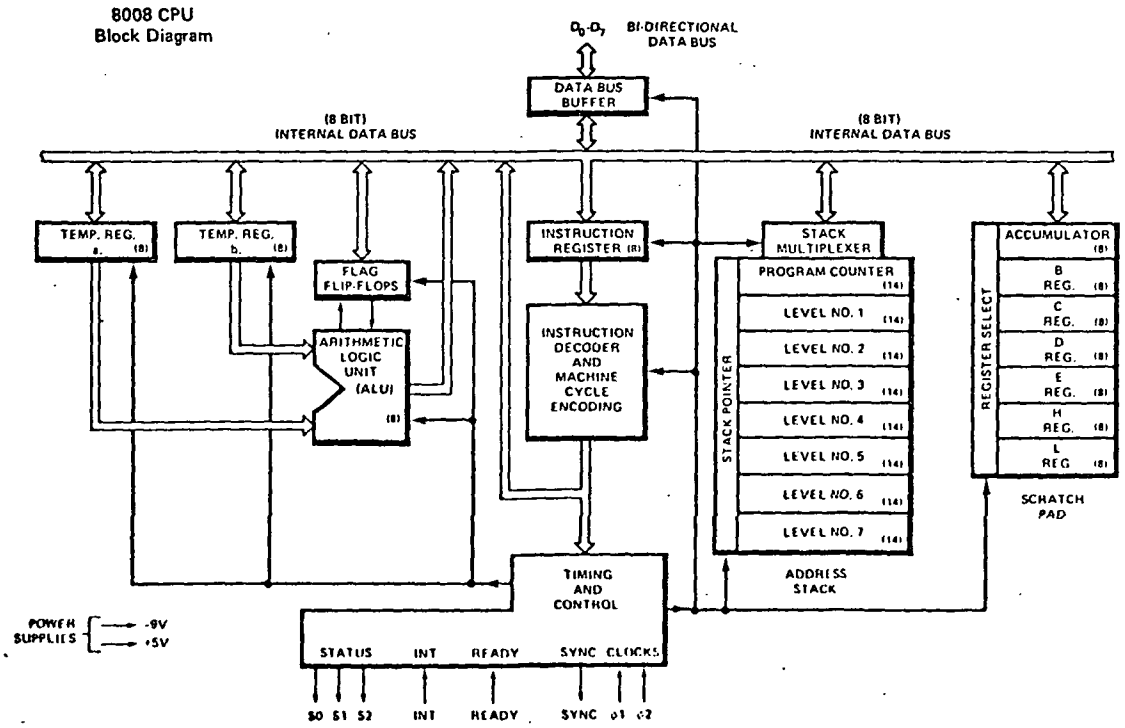
This CPU contains six 8-bit data registers, an 8-bit accumulator, two 8-bit temporary registers, four flag bits (carry, zero, sign, parity), and an 8-bit parallel binary arithmetic unit which implements addition, subtraction, and logical operations. A memory stack containing a 14-bit program counter and seven 14-bit words is used internally to store program and subroutine addresses. The 14-bit address permits the direct addressing of 16K words of memory (any mix of RAM, ROM or S.R.).

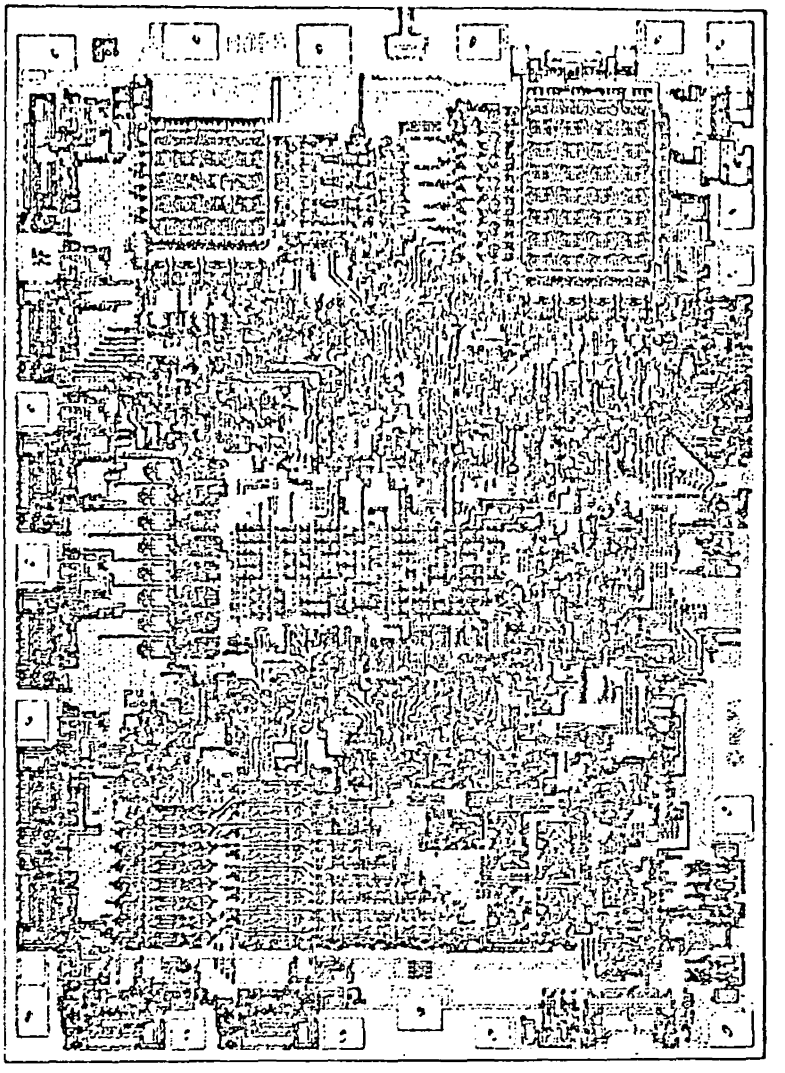
The instruction set of the 8008 consists of 48 instructions including data manipulation, binary arithmetic, and jump to subroutine.

The normal program flow of the 8008 may be interrupted through the use of the INTERRUPT control line. This allows the servicing of slow I/O peripheral devices while also executing the main program.

The READY command line synchronizes the 8008 to the memory cycle allowing any type or speed of semiconductor memory to be used.

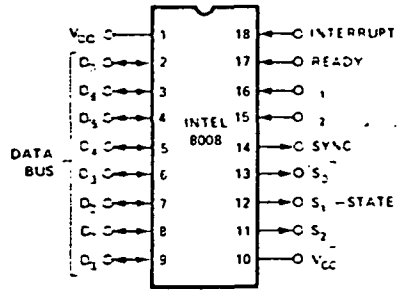
8008 CPU
Block Diagram





8008 PHOTOMICROGRAPH

8008 FUNCTIONAL PIN DESCRIPTION



D_0-D_7

BI-DIRECTIONAL DATA BUS. All address and data communication between the processor and the program memory, data memory, and I/O devices occurs on these 8 lines. Cycle control information is also available.

INT

INTERRUPT input. A logic "1" level at this input causes the processor to enter the INTERRUPT mode.

READY

READY input. This command line is used to synchronize the 8008 to the memory cycle allowing any speed memory to be used.

SYNC

SYNC output. Synchronization signal generated by the processor. It indicates the beginning of a machine cycle.

ϕ_1, ϕ_2

Two phase clock inputs.

S_0, S_1, S_2

MACHINE STATE OUTPUTS. The processor controls the use of the data bus and determines whether it will be sending or receiving data. State signals S_0, S_1, and S_2, along with SYNC inform the peripheral circuitry of the state of the processor.

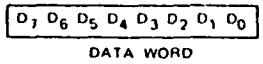
V_{CC} +5V ±5%

V_{DD} -9V ±5%

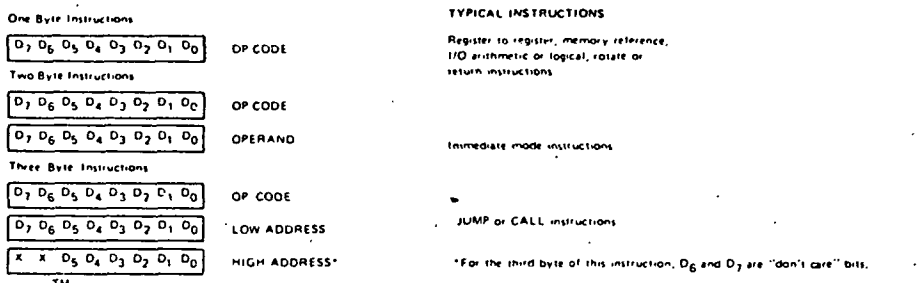
BASIC INSTRUCTION SET

Data and Instruction Formats

Data in the 8008 is stored in the form of 8 bit binary integers. All data transfers to the system data bus will be in the same format.



The program instructions may be one, two, or three bytes in length. Multiple byte instructions must be stored in successive words in program memory. The instruction formats then depend on the particular operation executed.



For the MCS-8™ a logic "1" is defined as a high level and a logic "0" is defined as a low level.

Index Register Instructions

The load instructions do not affect the flag flip-flops. The increment and decrement instructions affect all flip-flops except the carry.

MNEMONIC	MINIMUM STATES REQUIRED	INSTRUCTION CODE $D_7 D_6$	$D_5 D_4 D_3$	$D_2 D_1 D_0$	DESCRIPTION OF OPERATION
(1) MOV r_1, r_2	(5)	1 1	D D D	S S S	Load index register r_1 with the content of index register r_2.
(2) MOV r, M	(8)	1 1	D D D	1 1 1	Load index register r with the content of memory register M.
MOV M, r	(7)	1 1	1 1 1	S S S	Load memory register M with the content of index register r.
(3) MVI r	(8)	0 0 B B	D D D B B B	1 1 0 B B B	Load index register r with data B . . . B.
MVI M	(9)	0 0 B B	1 1 1 B B B	1 1 0 B B B	Load memory register M with data B . . . B.
INR r	(5)	0 0	D D D	0 0 0	Increment the content of index register r (r ≠ A).
DCR r	(5)	0 0	D D D	0 0 1	Decrement the content of index register r (r ≠ A).

Accumulator Group Instructions

The result of the ALU instructions affect all of the flag flip-flops. The rotate instructions affect only the carry flip-flop.

ADD r	(5)	1 0	0 0 0	S S S	Add the content of index register r, memory register M, or data B . . . B to the accumulator. An overflow (carry) sets the carry flip-flop.
ADD M	(8)	1 0	0 0 0	1 1 1	
ADI	(8)	0 0 B B	0 0 0 B B B	1 0 0 B B B	
ADC r	(5)	1 0	0 0 1	S S S	Add the content of index register r, memory register M, or data B . . . B from the accumulator with carry. An overflow (carry) sets the carry flip-flop.
ADC M	(8)	1 0	0 0 1	1 1 1	
ACI	(8)	0 0 B B	0 0 1 B B B	1 0 0 B B B	
SUB r	(5)	1 0	0 1 0	S S S	Subtract the content of index register r, memory register M, or data B . . . B from the accumulator. An underflow (borrow) sets the carry flip-flop.
SUB M	(8)	1 0	0 1 0	1 1 1	
SUI	(8)	0 0 B B	0 1 0 B B B	1 0 0 B B B	
SBB r	(5)	1 0	0 1 1	S S S	Subtract the content of index register r, memory register M, or data B . . . B from the accumulator with borrow. An underflow (borrow) sets the carry flip-flop.
SBB M	(8)	1 0	0 1 1	1 1 1	
SBI	(8)	0 0 B B	0 1 1 B B B	1 0 0 B B B	

BASIC INSTRUCTION SET

MNEMONIC	MINIMUM STATES REQUIRED	INSTRUCTION CODE D_7D_6	$D_5D_4D_3$	$D_2D_1D_0$	DESCRIPTION OF OPERATION
ANA r	(5)	1 0	1 0 0	S S S	Compute the logical AND of the content of index register r, memory register M, or data B . . . B with the accumulator.
ANA M	(8)	1 0	1 0 0	1 1 1	
ANI	(8)	0 0	1 0 0	1 0 0	
		B B	B B B	B B B	
XRA r	(5)	1 0	1 0 1	S S S	Compute the EXCLUSIVE OR of the content of index register r, memory register M, or data B . . . B with the accumulator.
XRA M	(8)	1 0	1 0 1	1 1 1	
XRI	(8)	0 0	1 0 1	1 0 0	
		B B	B B B	B B B	
ORA r	(5)	1 0	1 1 0	S S S	Compute the INCLUSIVE OR of the content of index register r, memory register m, or data B . . . B with the accumulator.
ORA M	(8)	1 0	1 1 0	1 1 1	
ORI	(8)	0 0	1 1 0	1 0 0	
		B B	B B B	B B B	
CMP r	(5)	1 0	1 1 1	S S S	Compare the content of index register r, memory register M, or data B . . . B with the accumulator. The content of the accumulator is unchanged.
CMP M	(8)	1 0	1 1 1	1 1 1	
CPI	(8)	0 0	1 1 1	1 0 0	
		B B	B B B	B B B	
RLC	(5)	0 0	0 0 0	0 1 0	Rotate the content of the accumulator left.
RRC	(5)	0 0	0 0 1	0 1 0	Rotate the content of the accumulator right.
RAL	(5)	0 0	0 1 0	0 1 0	Rotate the content of the accumulator left through the carry.
RAR	(5)	0 0	0 1 1	0 1 0	Rotate the content of the accumulator right through the carry.

Program Counter and Stack Control Instructions

(4) JMP	(11)	0 1	X X X	1 0 0	Unconditionally jump to memory address $B_3 \ldots B_3B_2 \ldots B_2$.
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
(5) JNC, JNZ, JP, JPO	(9 or 11)	0 1	0 C_4 C_3	0 0 0	Jump to memory address $B_3 \ldots B_3B_2 \ldots B_2$ if the condition flip-flop is false. Otherwise, execute the next instruction in sequence.
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
JC, JZ JM, JPE	(9 or 11)	0 1	1 C_4 C_3	0 0 0	Jump to memory address $B_3 \ldots B_3B_2 \ldots B_2$ if the condition flip-flop is true. Otherwise, execute the next instruction in sequence.
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
CALL	(11)	0 1	X X X	1 1 0	Unconditionally call the subroutine at memory address $B_3 \ldots$ $B_3B_2 \ldots B_2$. Save the current address (up one level in the stack).
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
CNC, CNZ, CP, CPO	(9 or 11)	0 1	0 C_4 C_3	0 1 0	Call the subroutine at memory address $B_3 \ldots B_3B_2 \ldots B_2$ if the condition flip-flop is false, and save the current address (up one level in the stack.) Otherwise, execute the next instruction in sequence.
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
CC, CZ, CM, CPE	(9 or 11)	0 1	1 C_4 C_3	0 1 0	Call the subroutine at memory address $B_3 \ldots B_3B_2 \ldots B_2$ if the condition flip-flop is true, and save the current address (up one level in the stack). Otherwise, execute the next instruction in sequence.
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
RET	(5)	0 0	X X X	1 1 1	Unconditionally return (down one level in the stack).
RNC, RNZ, RP, RPO	(3 or 5)	0 0	0 C_4 C_3	0 1 1	Return (down one level in the stack) if the condition flip-flop is false. Otherwise, execute the next instruction in sequence.
RC, RZ RM, RPE	(3 or 5)	0 0	1 C_4 C_3	0 1 1	Return (down one level in the stack) if the condition flip-flop is true. Otherwise, execute the next instruction in sequence.
RST	(5)	0 0	A A A	1 0 1	Call the subroutine at memory address AAA000 (up one level in the stack).

Input/Output Instructions

IN	(8)	0 1	0 0 M	M M 1	Read the content of the selected input port (MMM) into the accumulator.
OUT	(6)	0 1	R R M	M M 1	Write the content of the accumulator into the selected output port (RRMMM, RR ≠ 00).

Machine Instruction

HLT	(4)	0 0	0 0 0	0 0 X	Enter the STOPPED state and remain there until interrupted.
	(4)	1 1	1 1 1	1 1 1	

NOTES

(1) SSS = Source Index Register, DDD = Destination Index Register. These registers, r_i, are designated A(accumulator = 000), B(001), C(010), D(011), E(100), H(101), L(110).

(2) Memory registers are addressed by the contents of registers H & L.

(3) Additional bytes of instruction are designated by BBBBBBBB.

(4) X = "Don't Care".

(5) Flag flip-flops are defined by C_4C_3: carry (00 overflow or underflow), zero (01 result is zero), sign (10 MSB of result is "1"), parity (11 parity is even).

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to +70°C
Storage Temperature	−55°C to +150°C
Input Voltages and Supply Voltage With Respect to V_{CC}	+0.5 to −20V
Power Dissipation	1.0 W @ 25°C

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied.

D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5% unless otherwise specified. Logic "1" is defined as the more positive level (V_{IH}, V_{OH}). Logic "0" is defined as the more negative level (V_{IL}, V_{OL}).

SYMBOL	PARAMETER	LIMITS MIN.	LIMITS TYP.	LIMITS MAX.	UNIT	TEST CONDITIONS
I_{DD}	AVERAGE SUPPLY CURRENT-OUTPUTS LOADED*		30	60	mA	T_A = 25°C
I_{LI}	INPUT LEAKAGE CURRENT			10	μA	V_{IN} = 0V
V_{IL}	INPUT LOW VOLTAGE (INCLUDING CLOCKS)	V_{DD}		V_{CC}−4.2	V	
V_{IH}	INPUT HIGH VOLTAGE (INCLUDING CLOCKS)	V_{CC}−1.5		V_{CC}+0.3	V	
V_{OL}	OUTPUT LOW VOLTAGE			0.4	V	I_{OL} = 0.44mA, C_L = 200 pF
V_{OH}	OUTPUT HIGH VOLTAGE	V_{CC}−1.5			V	I_{OH} = 0.2mA

*Measurements are made while the 8008 is executing a typical sequence of instructions. The test load is selected such that at V_{OL} = 0.4V, I_{OL} = 0.44mA on each output.

A.C. CHARACTERISTICS

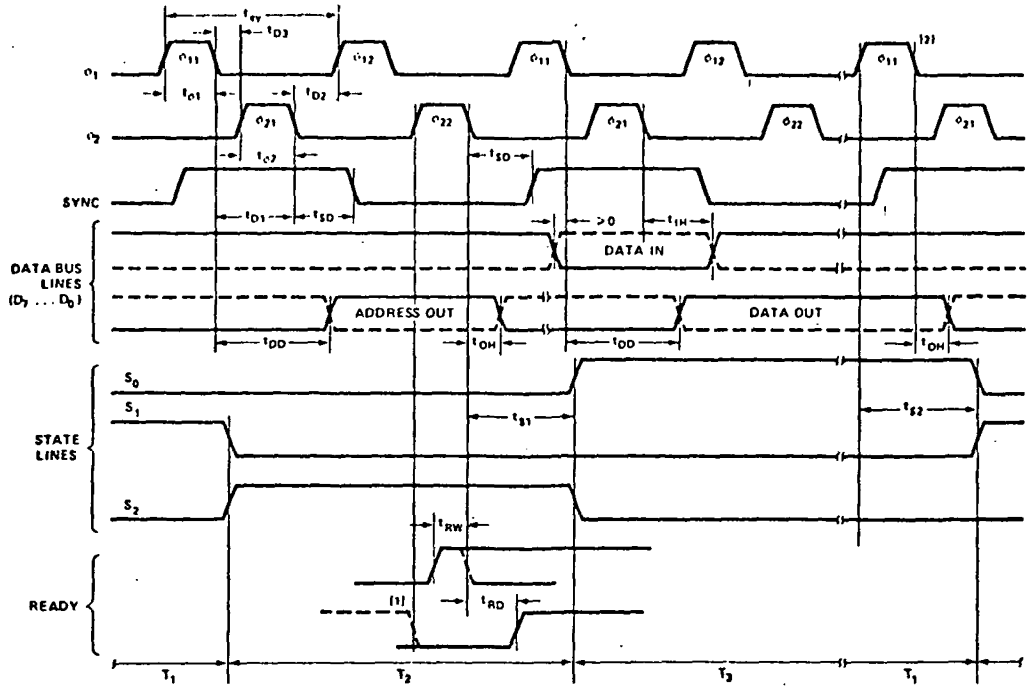
T_A = 0°C to 70°C; V_{CC} = +5V ±5%, V_{DD} = −9V ±5%. All measurements are referenced to 1.5V levels.

SYMBOL	PARAMETER	8008 LIMITS MIN.	8008 LIMITS MAX.	8008-1 LIMITS MIN.	8008-1 LIMITS MAX.	UNIT	TEST CONDITIONS
t_{CY}	CLOCK PERIOD	2	3	1.25	3	μs	t_R,t_F = 50ns
t_R,t_F	CLOCK RISE AND FALL TIMES		50		50	ns	
$t_{\phi 1}$	PULSE WIDTH OF ϕ_1	.70		.35		μs	
$t_{\phi 2}$	PULSE WIDTH OF ϕ_2	.55		.35		μs	
t_{D1}	CLOCK DELAY FROM FALLING EDGE OF ϕ_1 TO FALLING EDGE OF ϕ_2	.90	1.1		1.1	μs	
t_{D2}	CLOCK DELAY FROM ϕ_2 TO ϕ_1	.40		.35		μs	
t_{D3}	CLOCK DELAY FROM ϕ_1 TO ϕ_2	.20		.20		μs	
t_{DD}	DATA OUT DELAY		1.0		1.0	μs	C_L = 100pF
t_{OH}	HOLD TIME FOR DATA BUS OUT	.10		.10		μs	
t_{IH}	HOLD TIME FOR DATA IN	[1]		[1]		μs	
t_{SD}	SYNC OUT DELAY		.70		.70	μs	C_L = 100pF
t_{S1}	STATE OUT DELAY (ALL STATES EXCEPT T1 AND T1I)[2]		1.1		1.1	μs	C_L = 100pF
t_{S2}	STATE OUT DELAY (STATES T1 AND T1I)		1.0		1.0	μs	C_L = 100pF
t_{RW}	PULSE WIDTH OF READY DURING ϕ_{22} TO ENTER T3 STATE	.35		.35		μs	
t_{RD}	READY DELAY TO ENTER WAIT STATE	.20		.20		μs	

[1] t_{IH} MIN ≥ t_{SD}

[2] If the INTERRUPT is not used, all states have the same output delay, t_{S1}.

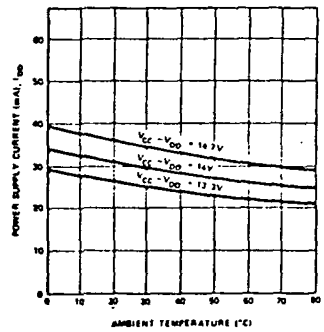
TIMING DIAGRAM



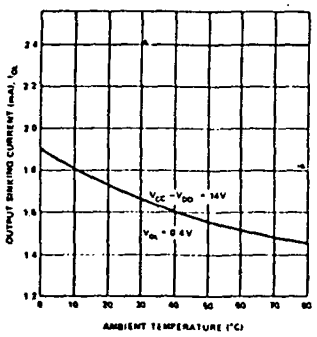
Notes: 1. READY line must be at "0" prior to ϕ_{22} of T_2 to guarantee entry into the WAIT state.
2. INTERRUPT line must not change levels within 200ns (max.) of falling edge of ϕ_1.

TYPICAL D.C. CHARACTERISTICS

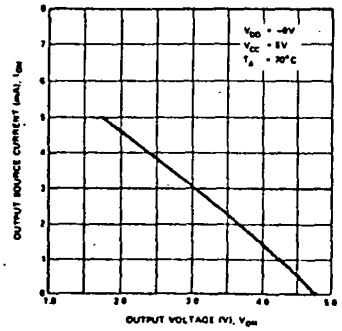
POWER SUPPLY CURRENT VS. TEMPERATURE



OUTPUT SINKING CURRENT VS. TEMPERATURE

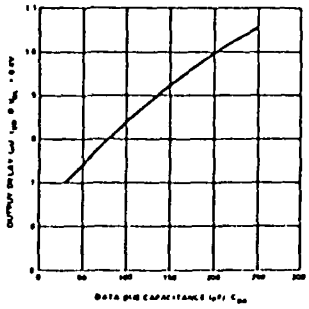


OUTPUT SOURCE CURRENT VS. OUTPUT VOLTAGE



TYPICAL A.C. CHARACTERISTICS

DATA OUT DELAY VS. OUTPUT LOAD CAPACITANCE



CAPACITANCE f = 1MHz; T_A = 25°C; Unmeasured Pins Grounded

SYMBOL	TEST	LIMIT (pF)	
		TYP.	MAX.
C_{IN}	INPUT CAPACITANCE	5	10
C_{D8}	DATA BUS I/O CAPACITANCE	5	10
C_{OUT}	OUTPUT CAPACITANCE	5	10

Am2901

Four-Bit Bipolar Microprocessor Slice

DISTINCTIVE CHARACTERISTICS

- Two-address architecture –
  Independent simultaneous access to two working registers saves machine cycles.
- Eight-function ALU –
  Performs addition, two subtraction operations, and five logic functions on two source operands.
- Flexible data source selection –
  ALU data is selected from five source ports for a total of 203 source operand pairs for every ALU function.
- Left/right shift independent of ALU –
  Add and shift operations take only one cycle.
- Four status flags –
  Carry, overflow, zero, and negative.
- Expandable –
  Connect any number of Am2901's together for longer word lengths.
- Microprogrammable –
  Three groups of three bits each for source operand, ALU function, and destination control.

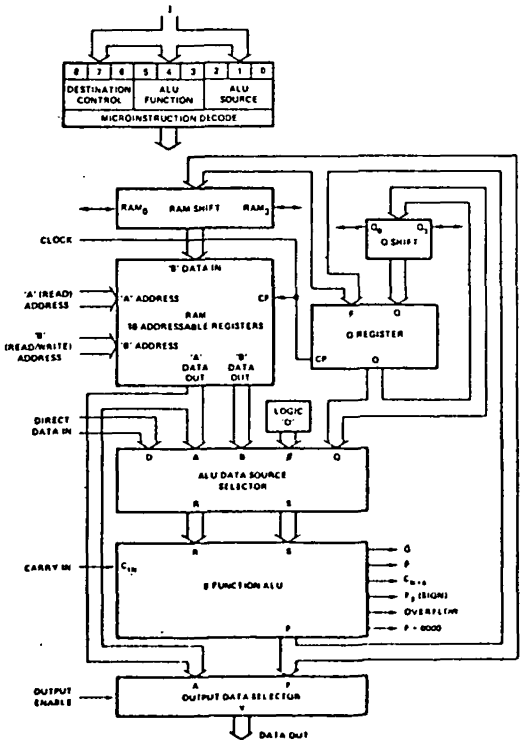
GENERAL DESCRIPTION

The four-bit bipolar microprocessor slice is designed as a high-speed cascadable element intended for use in CPU's, peripheral controllers, programmable microprocessors and numerous other applications. The microinstruction flexibility of the Am2901 will allow efficient emulation of almost any digital computing machine.

The device, as shown in the block diagram below, consists of a 16-word by 4-bit two-port RAM, a high-speed ALU, and the associated shifting, decoding and multiplexing circuitry. The nine-bit microinstruction word is organized into three groups of three bits each and selects the ALU source operands, the ALU function, and the ALU destination register. The microprocessor is cascadable with full look-ahead or with ripple carry, has three-state outputs, and provides various status flag outputs from the ALU. Advanced low-power Schottky processing is used to fabricate this 40-lead LSI chip.

TABLE OF CONTENTS

MICROPROCESSOR SLICE BLOCK DIAGRAM



ARCHITECTURE

A detailed block diagram of the bipolar microprogrammable microprocessor structure is shown in Figure 1. The circuit is a four-bit slice cascadable to any number of bits. Therefore, all data paths within the circuit are four bits wide. The two key elements in the Figure 1 block diagram are the 16-word by 4-bit 2-port RAM and the high-speed ALU.

Data in any of the 16 words of the Random Access Memory (RAM) can be read from the A-port of the RAM as controlled by the 4-bit A address field input. Likewise, data in any of the 16 words of the RAM as defined by the B address field input can be simultaneously read from the B-port of the RAM. The same code can be applied to the A select field and B select field in which case the identical file data will appear at both the RAM A-port and B-port outputs simultaneously.

When enabled by the RAM write enable (RAM EN), new data is always written into the file (word) defined by the B address field of the RAM. The RAM data input field is driven by a 3-input multiplexer. This configuration is used to shift the ALU output data (F) if desired. This three-input multiplexer scheme allows the data to be shifted up one bit position, shifted down one bit position, or not shifted in either direction.

The RAM A-port data outputs and RAM B-port data outputs drive separate 4-bit latches. These latches hold the RAM data while the clock input is LOW. This eliminates any possible race conditions that could occur while new data is being written into the RAM.

The high-speed Arithmetic Logic Unit (ALU) can perform three binary arithmetic and five logic operations on the two 4-bit input words R and S. The R input field is driven from a 2-input multiplexer, while the S input field is driven from a 3-input multiplexer. Both multiplexers also have an inhibit capability; that is, no data is passed. This is equivalent to a "zero" source operand.

Referring to Figure 1, the ALU R-input multiplexer has the RAM A-port and the direct data inputs (D) connected as inputs. Likewise, the ALU S-input multiplexer has the RAM A-port, the RAM B-port and the Q register connected as inputs.

This multiplexer scheme gives the capability of selecting various pairs of the A, B, D, Q and "0" inputs as source operands to the ALU. These five inputs, when taken two at a time, result in ten possible combinations of source operand pairs. These combinations include AB, AD, AQ, A0, BD, BQ, B0, DQ, D0 and Q0. It is apparent that AD, AQ and A0 are somewhat redundant with BD, BQ and B0 in that if the A address and B address are the same, the identical function results. Thus, there are only seven completely non-redundant source operand pairs for the ALU. The Am2901 microprocessor implements eight of these pairs. The microinstruction inputs used to select the ALU source operands are the I_0, I_1, and I_2 inputs. The definition of I_0, I_1, and I_2 for the eight source operand combinations are as shown in Figure 2. Also shown is the octal code for each selection.

The two source operands not fully described as yet are the D input and Q input. The D input is the four-bit wide direct data field input. This port is used to insert all data into the working registers inside the device. Likewise, this input can be used in the ALU to modify any of the internal data files. The Q register is a separate 4-bit file intended primarily for multiplication and division routines but it can also be used as an accumulator or holding register for some applications.

The ALU itself is a high-speed arithmetic/logic operator capable of performing three binary arithmetic and five logic functions. The I_3, I_4, and I_5 microinstruction inputs are used to select the ALU function. The definition of these inputs is shown in Figure 3. The octal code is also shown for reference. The normal technique for cascading the ALU of several devices is in a look-ahead carry mode. Carry generate, $\overline{G}$, and carry propagate, $\overline{P}$, are outputs of the device for use with a carry-look-ahead-generator such as the Am2902 ('182). A carry-out, C_{n+4}, is also generated and is available as an output for use as the carry flag in a status register. Both carry-in (C_n) and carry-out (C_{n+4}) are active HIGH.

The ALU has three other status-oriented outputs. These are F_3, F = 0, and overflow (OVR). The F_3 output is the most significant (sign) bit of the ALU and can be used to determine positive or negative results without enabling the three-state data outputs. F_3 is non-inverted with respect to the sign bit output Y_3. The F = 0 output is used for zero detect. It is an open-collector output and can be wire OR'ed between microprocessor slices. F = 0 is HIGH when all F outputs are LOW. The overflow output (OVR) is used to flag arithmetic operations that exceed the available two's complement number range. The overflow output (OVR) is HIGH when overflow exists. That is, when C_{n+3} and C_{n+4} are not the same polarity.

The ALU data output is routed to several destinations. It can be a data output of the device and it can also be stored in the RAM or the Q register. Eight possible combinations of ALU destination functions are available as defined by the I_6, I_7, and I_8 microinstruction inputs. These combinations are shown in Figure 4.

The four-bit data output field (Y) features three-state outputs and can be directly bus organized. An output control ($\overline{OE}$) is used to enable the three-state outputs. When $\overline{OE}$ is HIGH, the Y outputs are in the high-impedance state.

A two-input multiplexer is also used at the data output such that either the A-port of the RAM or the ALU outputs (F) are selected at the device Y outputs. This selection is controlled by the I_6, I_7, and I_8 microinstruction inputs. Refer to Figure 4 for the selected output for each microinstruction code combination.

As was discussed previously, the RAM inputs are driven from a three-input multiplexer. This allows the ALU outputs to be entered non-shifted, shifted up one position (X2) or shifted down one position (÷2). The shifter has two ports; one is labeled RAM_0 and the other is labeled RAM_3. Both of these ports consist of a buffer-driver with a three-state output and an input to the multiplexer. Thus, in the shift up mode, the RAM_3 buffer is enabled and the RAM_0 multiplexer input is enabled. Likewise, in the shift down mode, the RAM_0 buffer and RAM_3 input are enabled. In the no-shift mode, both buffers are in the high-impedance state and the multiplexer inputs are not selected. This shifter is controlled from the I_6, I_7 and I_8 microinstruction inputs as defined in Figure 4.

Similarly, the Q register is driven from a 3-input multiplexer. In the no-shift mode, the multiplexer enters the ALU data into the Q register. In either the shift-up or shift-down mode, the multiplexer selects the Q register data appropriately shifted up or down. The Q shifter also has two ports; one is labeled Q_0 and the other is Q_3. The operation of these two ports is similar to the RAM shifter and is also controlled from I_6, I_7, and I_8 as shown in Figure 4.

The clock input to the Am2901 controls the RAM, the Q register, and the A and B data latches. When enabled, data is clocked into the Q register on the LOW-to-HIGH transition of the clock. When the clock input is HIGH, the A and B latches are open and will pass whatever data is present at the RAM outputs. When the clock input is LOW, the latches are closed and will retain the last data entered. If the RAM-EN is enabled, new data will be written into the RAM file (word) defined by the B address field when the clock input is LOW.

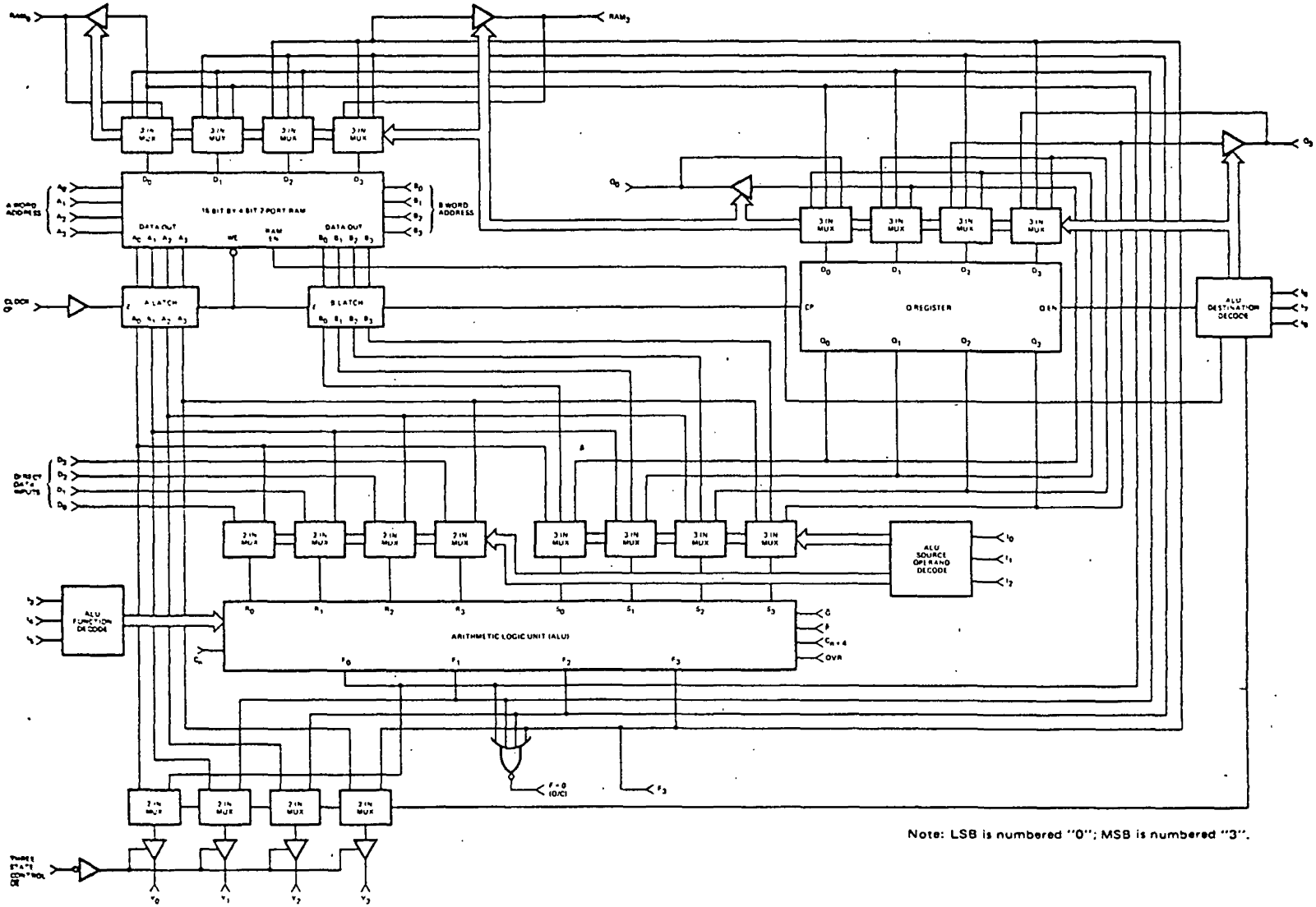


Figure 1. Detailed Am2901 Microprocessor Block Diagram.

MICRO CODE				ALU SOURCE OPERANDS	
I_2	I_1	I_0	Octal Code	R	S
L	L	L	0	A	Q
L	L	H	1	A	B
L	H	L	2	O	Q
L	H	H	3	O	B
H	L	L	4	O	A
H	L	H	5	D	A
H	H	L	6	D	Q
H	H	H	7	D	O

Figure 2. ALU Source Operand Control.

MICRO CODE				ALU Function	Symbol
I_8	I_4	I_3	Octal Code		
L	L	L	0	R Plus S	R + S
L	L	H	1	S Minus R	S − R
L	H	L	2	R Minus S	R − S
L	H	H	3	R OR S	R ∨ S
H	L	L	4	R AND S	R ∧ S
H	L	H	5	$\overline{R}$ AND S	$\overline{R}$ ∧ S
H	H	L	6	R EX-OR S	R ∀ S
H	H	H	7	R EX-NOR S	$\overline{R \forall S}$

Figure 3. ALU Function Control.

MICRO CODE				RAM FUNCTION		Q-REG. FUNCTION		Y OUTPUT	RAM SHIFTER		Q SHIFTER	
I_8	I_7	I_6	Octal Code	Shift	Load	Shift	Load		RAM_0	RAM_3	Q_0	Q_3
L	L	L	0	X	NONE	NONE	F→Q	F	X	X	X	X
L	L	H	1	X	NONE	X	NONE	F	X	X	X	X
L	H	L	2	NONE	F→B	X	NONE	A	X	X	X	X
L	H	H	3	NONE	F→B	X	NONE	F	X	X	X	X
H	L	L	4	DOWN	F/2→B	DOWN	Q/2→Q	F	F_0	IN_3	Q_0	IN_3
H	L	H	5	DOWN	F/2→B	X	NONE	F	F_0	IN_3	Q_0	X
H	H	L	6	UP	2F→B	UP	2Q→Q	F	IN_0	F_3	IN_0	Q_3
H	H	H	7	UP	2F→B	X	NONE	F	IN_0	F_3	X	Q_3

X= Don't care. Electrically, the shift pin is a TTL input internally connected to a three-state output which is in the high-impedance state.
B= Register Addressed by B inputs.
Up is toward MSB, Down is toward LSB.

Figure 4. ALU Destination Control.

OCTAL I_{210} / I_{543} ALU Source / ALU Function	0	1	2	3	4	5	6	7
	A, Q	A, B	O, Q	O, B	O, A	D, A	D, Q	D, O
0 C_n = L R Plus S	A+Q	A+B	Q	B	A	D+A	D+Q	D
0 C_n = H R Plus S	A+Q+1	A+B+1	Q+1	B+1	A+1	D+A+1	D+Q+1	D+1
1 C_n = L S Minus R	Q−A−1	B−A−1	Q−1	B−1	A−1	A−D−1	Q−D−1	−D−1
1 C_n = H S Minus R	Q−A	B−A	Q	B	A	A−D	Q−D	−D
2 C_n = L R Minus S	A−Q−1	A−B−1	−Q−1	−B−1	−A−1	D−A−1	D−Q−1	D−1
2 C_n = H R Minus S	A−Q	A−B	−Q	−B	−A	D−A	D−Q	D
3 R OR S	A∨Q	A∨B	Q	B	A	D∨A	D∨Q	D
4 R AND S	A∧Q	A∧B	0	0	0	D∧A	D∧Q	0
5 $\overline{R}$ AND S	$\overline{A}$∧Q	$\overline{A}$∧B	Q	B	A	$\overline{D}$∧A	$\overline{D}$∧Q	0
6 R EX-OR S	A∀Q	A∀B	Q	B	A	D∀A	D∀Q	D
7 R EX-NOR S	$\overline{A \forall Q}$	$\overline{A \forall B}$	$\overline{Q}$	$\overline{B}$	$\overline{A}$	$\overline{D \forall A}$	$\overline{D \forall Q}$	$\overline{D}$

+ = Plus; − = Minus; ∨ = OR; ∧ = AND; ∀ = EX-OR

Figure 5. Source Operand and ALU Function Matrix.

SOURCE OPERANDS AND ALU FUNCTIONS

There are eight source operand pairs available to the ALU as selected by the I_0, I_1, and I_2 instruction inputs. The ALU can perform eight functions; five logic and three arithmetic. The I_3, I_4, and I_5 instruction inputs control this function selection. The carry input, C_n, also affects the ALU results when in the arithmetic mode. The C_n input has no effect in the logic mode. When I_0 through I_5 and C_n are viewed together, the matrix of Figure 5 results. This matrix fully defines the ALU/source operand function for each state.

The ALU functions can also be examined on a "task" basis, i.e., add, subtract, AND, OR, etc. In the arithmetic mode, the carry will affect the function performed while in the logic mode, the carry will have no bearing on the ALU output. Figure 6 defines the various logic operations that the Am2901 can perform and Figure 7 shows the arithmetic functions of the device. Both carry-in LOW (C_n = 0) and carry-in HIGH (C_n = 1) are defined in these operations.

Octal I_{543}, I_{210}	Group	Function
4 0	AND	A∧Q
4 1		A∧B
4 5		D∧A
4 6		D∧Q
3 0	OR	A∨Q
3 1		A∨B
3 5		D∨A
3 6		D∨Q
6 0	EX-OR	A∀Q
6 1		A∀B
6 5		D∀A
6 6		D∀Q
7 0	EX-NOR	$\overline{A \forall Q}$
7 1		$\overline{A \forall B}$
7 5		$\overline{D \forall A}$
7 6		$\overline{D \forall Q}$
7 2	INVERT	$\overline{Q}$
7 3		$\overline{B}$
7 4		$\overline{A}$
7 7		$\overline{D}$
6 2	PASS	Q
6 3		B
6 4		A
6 7		D
3 2	PASS	Q
3 3		B
3 4		A
3 7		D
4 2	"ZERO"	0
4 3		0
4 4		0
4 7		0
5 0	MASK	$\overline{A}$∧Q
5 1		$\overline{A}$∧B
5 5		$\overline{D}$∧A
5 6		$\overline{D}$∧Q

Figure 6. ALU Logic Mode Functions. (C_n Irrelevant)

Octal	C_n = 0 (Low)		C_n = 1 (High)	
I_{543}, I_{210}	Group	Function	Group	Function
0 0	ADD	A+Q	ADD plus one	A+Q+1
0 1		A+B		A+B+1
0 5		D+A		D+A+1
0 6		D+Q		D+Q+1
0 2	PASS	Q	Increment	Q+1
0 3		B		B+1
0 4		A		A+1
0 7		D		D+1
1 2	Decrement	Q–1	PASS	Q
1 3		B–1		B
1 4		A–1		A
2 7		D–1		D
2 2	1's Comp.	–Q–1	2's Comp. (Negate)	–Q
2 3		–B–1		–B
2 4		–A–1		–A
1 7		–D–1		–D
1 0	Subtract (1's Comp)	Q–A–1	Subtract (2's Comp)	Q–A
1 1		B–A–1		B–A
1 5		A–D–1		A–D
1 6		Q–D–1		Q–D
2 0		A–Q–1		A–Q
2 1		A–B–1		A–B
2 5		D–A–1		D–A
2 6		D–Q–1		D–Q

Figure 7. ALU Arithmetic Mode Functions.

LOGIC FUNCTIONS FOR G, P, C_{n+4}, AND OVR

The four signals G, P, C_{n+4}, and OVR are designed to indicate carry and overflow conditions when the Am2901 is in the add or subtract mode. The table below indicates the logic equations for these four signals for each of the eight ALU functions. The R and S inputs are the two inputs selected according to Figure 2.

Definitions (+ = OR)

$P_0 = R_0 + S_0$ $\quad G_0 = R_0S_0$

$P_1 = R_1 + S_1$ $\quad G_1 = R_1S_1$

$P_2 = R_2 + S_2$ $\quad G_2 = R_2S_2$

$P_3 = R_3 + S_3$ $\quad G_3 = R_3S_3$

$$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_n$$

$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_n$$

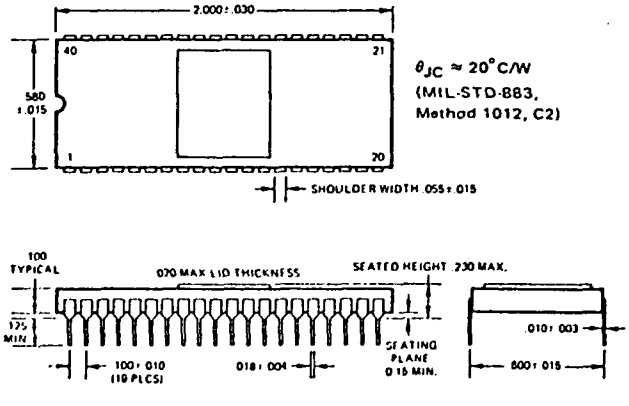
I_{543}	Function	$\overline{P}$	$\overline{G}$	C_{n+4}	OVR
0	R + S	$\overline{P_3P_2P_1P_0}$	$\overline{G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0}$	C_4	$C_3 \forall C_4$
1	S − R	← Same as R + S equations, but substitute $\overline{R_i}$ for R_i in definitions →			
2	R − S	← Same as R + S equations, but substitute $\overline{S_i}$ for S_i in definitions →			
3	R ∨ S	LOW	$P_3P_2P_1P_0$	$\overline{P_3P_2P_1P_0} + C_n$	$\overline{P_3P_2P_1P_0} + C_n$
4	R ∧ S	LOW	$\overline{G_3 + G_2 + G_1 + G_0}$	$G_3 + G_2 + G_1 + G_0 + C_n$	$G_3 + G_2 + G_1 + G_0 + C_n$
5	$\overline{R}$ ∧ S	LOW	← Same as R ∧ S equations, but substitute $\overline{R_i}$ for R_i in definitions →		
6	R ∀ S	← Same as $\overline{R \forall S}$, but substitute $\overline{R_i}$ for R_i in definitions →			
7	$\overline{R \forall S}$	$G_3 + G_2 + G_1 + G_0$	$G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$	$\overline{G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1P_0 (G_0 + \overline{C_n})}$	See note

Note: $[\overline{P}_2 + \overline{G}_2\overline{P}_1 + \overline{G}_2\overline{G}_1\overline{P}_0 + \overline{G}_2\overline{G}_1\overline{G}_0C_n] \forall [\overline{P}_3 + \overline{G}_3\overline{P}_2 + \overline{G}_3\overline{G}_2\overline{P}_1 + \overline{G}_3\overline{G}_2\overline{G}_1\overline{P}_0 + \overline{G}_3\overline{G}_2\overline{G}_1\overline{G}_0C_n]$ + = OR

Figure 8.

PACKAGE OUTLINE

40-Pin Ceramic (Side Brazed)



42-Pin Flat Package

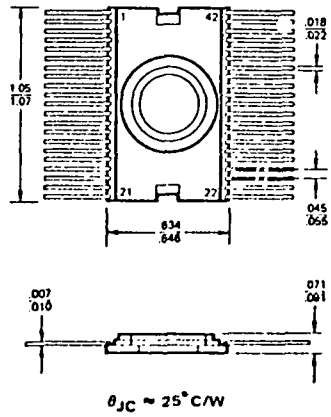
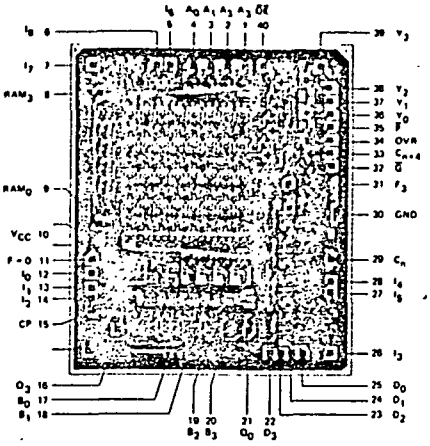


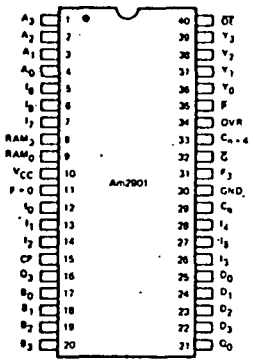
Figure 9.

METALLIZATION AND PAD LAYOUT



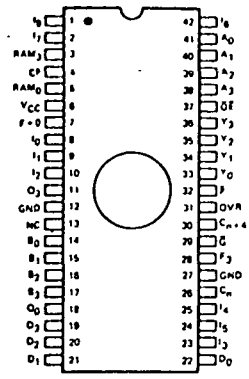
DIE SIZE 0.167" x 0.198"

CONNECTION DIAGRAM
Top View
DIP



Note: Pin 1 is marked for orientation.

FLAT PACK CONNECTIONS
Top View



Note: Both grounds (pins 12 and 27) must be connected.

Figure 10.

PIN DEFINITIONS

A_{0-3} The four address inputs to the register stack used to select one register whose contents are displayed through the A-port.

B_{0-3} The four address inputs to the register stack used to select one register whose contents are displayed through the B-port and into which new data can be written when the clock goes LOW.

I_{0-8} The nine instruction control lines to the Am2901, used to determine what data sources will be applied to the ALU (I_{012}), what function the ALU will perform (I_{345}), and what data is to be deposited in the Q-register or the register stack (I_{678}).

Q_3 RAM_3 A shift line at the MSB of the Q register (Q_3) and the register stack (RAM_3). Electrically these lines are three-state outputs connected to TTL inputs internal to the Am2901. When the destination code on I_{678} indicates an up shift (octal 6 or 7) the three-state outputs are enabled and the MSB of the Q register is available on the Q_3 pin and the MSB of the ALU output is available on the RAM_3 pin. Otherwise, the three-state outputs are OFF (high-impedance) and the pins are electrically LS-TTL inputs. When the destination code calls for a down shift, the pins are used as the data inputs to the MSB of the Q register (octal 4) and RAM (octal 4 or 5).

Q_0 RAM_0 Shift lines like Q_3 and RAM_3, but at the LSB of the Q-register and RAM. These pins are tied to the Q_3 and RAM_3 pins of the adjacent device to transfer data between devices for up and down shifts of the Q register and ALU data.

D_{0-3} Direct data inputs. A four-bit data field which may be selected as one of the ALU data sources for entering data into the Am2901. D_0 is the LSB.

Y_{0-3} The four data outputs of the Am2901. These are three-state output lines. When enabled, they display either the four outputs of the ALU or the data on the A-port of the register stack, as determined by the destination code I_{678}.

$\overline{OE}$ Output Enable. When $\overline{OE}$ is HIGH, the Y outputs are OFF; when $\overline{OE}$ is LOW, the Y outputs are active (HIGH or LOW).

$\overline{P}, \overline{G}$ The carry generate and propagate outputs of the Am2901's ALU. These signals are used with the Am2902 for carry-lookahead. See Figure 8 for the logic equations.

OVR Overflow. This pin is logically the Exclusive-OR of the carry-in and carry-out of the MSB of the ALU. At the most significant end of the word, this pin indicates that the result of an arithmetic two's complement operation has overflowed into the sign-bit. See Figure 8 for logic equation.

F = 0 This is an open collector output which goes HIGH (OFF) if the data on the four ALU outputs F_{0-3} are all LOW. In positive logic, it indicates the result of an ALU operation is zero.

C_n The carry-in to the Am2901's ALU.

C_{n+4} The carry-out of the Am2901's ALU. See Figure 8 for equations.

CP The clock to the Am2901. The Q register and register stack outputs change on the clock LOW-to-HIGH transition. The clock LOW time is internally the write enable to the 16 x 4 RAM which comprises the "master" latches of the register stack. While the clock is LOW, the "slave" latches on the RAM outputs are closed, storing the data previously on the RAM outputs. This allows synchronous master-slave operation of the register stack.

MAXIMUM RATINGS (Above which the useful life may be impaired)

Storage Temperature	−65°C to +150°C
Temperature (Ambient) Under Bias	−55°C to +125°C
Supply Voltage to Ground Potential	−0.5 V to +6.3 V
DC Voltage Applied to Outputs for HIGH Output State	−0.5 V to +V_{CC} max.
DC Input Voltage	−0.5 V to +5.5 V
DC Output Current, Into Outputs	30 mA
DC Input Current	−30 mA to +5.0 mA

OPERATING RANGE

P/N	Ambient Temperature	V_{CC}
Am2901PC, DC	0°C to +70°C	4.75 V to 5.25 V
Am2901DM, FM	−55°C to +125°C	4.50 V to 5.50 V

STANDARD SCREENING

(Conforms to MIL-STD-883 for Class C Parts)

Step	MIL-STD-883 Method	Conditions	Level	
			Am2901PC, DC	Am2901DM, FM
Pre-Seal Visual Inspection	2010	B	100%	100%
Stabilization Bake	1008	C 24-hour 150°C	100%	100%
Temperature Cycle	1010	C −65°C to +150°C 10 cycles	100%	100%
Centrifuge	2001	B 10,000 G	100% *	100%
Fine Leak	1014	A 5×10^{-8} atm-cc/cm³	100% *	100%
Gross Leak	1014	C2 Fluorocarbon	100% *	100%
Electrical Test Subgroups 1 and 7	5004	See below for definitions of subgroups	100%	100%
Insert Additional Screening here for Class B Parts				
Group A Sample Tests	5005	See below for definitions of subgroups		
Subgroup 1			LTPD = 5	LTPD = 5
Subgroup 2			LTPD = 7	LTPD = 7
Subgroup 3			LTPD = 7	LTPD = 7
Subgroup 7			LTPD = 7	LTPD = 7
Subgroup 8			LTPD = 7	LTPD = 7
Subgroup 9			LTPD = 7	LTPD = 7

*Not applicable for Am2901PC

ADDITIONAL SCREENING FOR CLASS B PARTS

Step	MIL-STD-883 Method	Conditions	Level
			Am2901DMB, FMB
Burn-In	1015	D 125°C 160 hours min.	100%
Electrical Test	5004		
Subgroup 1			100%
Subgroup 2			100%
Subgroup 3			100%
Subgroup 7			100%
Subgroup 9			100%
Return to Group A Tests in Standard Screening			

ORDERING INFORMATION

Package Type	Temperature Range	Order Number
Molded DIP	0°C to +70°C	AM2901PC
Hermetic DIP	0°C to +70°C	AM2901DC
Hermetic DIP	−55°C to +125°C	AM2901DM
Hermetic Flat Pack	−55°C to +125°C	AM2901FM
Dice	0°C to +70°C	AM2901XC

GROUP A SUBGROUPS

(as defined in MIL-STD-883, method 5005)

Subgroup	Parameter	Temperature
1	DC	25°C
2	DC	Maximum rated temperature
3	DC	Minimum rated temperature
7	Function	25°C
8	Function	Maximum and minimum rated temperature
9	Switching	25°C
10	Switching	Maximum Rated Temeperature
11	Switching	Minimum Rated Temperature

ELECTRICAL CHARACTERISTICS OVER OPERATING RANGE (Unless Otherwise Noted)

(Group A, Subgroups 1, 2 and 3)

Parameters	Description	Test Conditions (Note 1)			Min.	Typ. (Note 2)	Max.	Units
V_{OH}	Output HIGH Voltage	V_{CC} = MIN., V_{IN} = V_{IH} or V_{IL}	I_{OH} = −1.6mA Y_0, Y_1, Y_2, Y_3		2.4			Volts
			I_{OH} = −1.0mA, C_{n+4}		2.4			
			I_{OH} = −800μA, OVR, $\overline{P}$		2.4			
			I_{OH} = −600μA, F_3		2.4			
			I_{OH} = −600μA $RAM_{0,3}, Q_{0,3}$		2.4			
			I_{OH} = −1.6mA, $\overline{G}$		2.4			
I_{CEX}	Output Leakage Current for F = 0 Output	V_{CC} = MIN., V_{OH} = 5.5V V_{IN} = V_{IH} or V_{IL}					250	μA
V_{OL}	Output LOW Voltage	V_{CC} = MIN., V_{IN} = V_{IH} or V_{IL}	I_{OL} = 16mA $Y_0, Y_1, Y_2, Y_3, \overline{G}$				0.5	Volts
			I_{OL} = 10mA, C_{n+4}, F = 0				0.5	
			I_{OL} = 8.0mA, OVR, $\overline{P}$				0.5	
			I_{OL} = 6.0mA, F_3 $RAM_{0,3}, Q_{0,3}$				0.5	
V_{IH}	Input HIGH Level	Guaranteed input logical HIGH voltage for all inputs			2.0			Volts
V_{IL}	Input LOW Level	Guaranteed input logical LOW voltage for all inputs		Military			0.7	Volts
				Commercial			0.8	
V_I	Input Clamp Voltage	V_{CC} = MIN., I_{IN} = −18mA					−1.5	Volts
I_{IL}	Input LOW Current	V_{CC} = MAX., V_{IN} = 0.5V	Clock, $\overline{OE}$				−0.36	mA
			A_0, A_1, A_2, A_3				−0.36	
			B_0, B_1, B_2, B_3				−0.36	
			D_0, D_1, D_2, D_3				−0.72	
			I_0, I_1, I_2, I_6, I_8				−0.36	
			I_3, I_4, I_5, I_7				−0.72	
			$RAM_{0,3}, Q_{0,3}$ (Note 4)				−0.8	
			C_n				−3.6	
I_{IH}	Input HIGH Current	V_{CC} = MAX., V_{IN} = 2.7V	Clock, $\overline{OE}$				20	μA
			A_0, A_1, A_2, A_3				20	
			B_0, B_1, B_2, B_3				20	
			D_0, D_1, D_2, D_3				40	
			I_0, I_1, I_2, I_6, I_8				20	
			I_3, I_4, I_5, I_7				40	
			$RAM_{0,3}, Q_{0,3}$ (Note 4)				100	
			C_n				200	
I_I	Input HIGH Current	V_{CC} = MAX., V_{IN} = 5.5V					1.0	mA
I_{OZH} I_{OZL}	Off State (High Impedance) Output Current	V_{CC} = MAX.	Y_0, Y_1, Y_2, Y_3	V_O = 2.4V			50	μA
				V_O = 0.5V			−50	
			$RAM_{0,3}, Q_{0,3}$	V_O = 2.4V (Note 4)			100	
				V_O = 0.5V (Note 4)			−800	
I_{OS}	Output Short Circuit Current (Note 3)	V_{CC} = 5.75V V_O = 0.5V	$Y_0, Y_1, Y_2, Y_3, \overline{G}$		−15		−40	mA
			C_{n+4}		−15		−40	
			OVR, $\overline{P}$		−15		−40	
			F_3		−15		−40	
			$RAM_{0,3}, Q_{0,3}$		−15		−40	
I_{CC}	Power Supply Current	V_{CC} = MAX.	Military			185	280	mA
			Commercial			185	280	

Notes: 1. For conditions shown as MIN. or MAX., use the appropriate value specified under Electrical Characteristics for the applicable device type.
2. Typical limits are at V_{CC} = 5.0V, 25°C ambient and maximum loading.
3. Not more than one output should be shorted at a time. Duration of the short circuit test should not exceed one second.
4. These are three state outputs internally connected to TTL inputs. Input characteristics are measured with I_{G78} in a state such that the three state output is OFF.

GUARANTEED OPERATING CONDITIONS OVER TEMPERATURE AND VOLTAGE

Tables I, II, and III below define the timing requirements of the Am2901 in a system. The Am2901 is guaranteed to function correctly over the operating range when used within the delay and set-up time constraints of these tables for the appropriate device type. The tables are divided into three types of parameters; clock characteristics, combinational delays from inputs to outputs, and set-up and hold time requirements. The latter table defines the time prior to the end of the cycle (i.e., clock LOW-to-HIGH transition) that each input must be stable to guarantee that the correct data is written into one of the internal registers.

The performance of the Am2901 within the limits of these tables is guaranteed by the testing defined as "Group A, Subgroup 9" Electrical Testing. For a copy of the tests and limits used for subgroup 9, contact Advanced Micro Devices' Product Marketing.

TABLE I

CYCLE TIME AND CLOCK CHARACTERISTICS

TIME	Am2901DC,PC	Am2901DM,FM
Read-Modify-Write Cycle (time from selection of A, B registers to end of cycle)	105ns	120ns
Maximum Clock Frequency to Shift Q Register (50% duty cycle)	9.5MHz	8.3MHz
Minimum Clock LOW Time	30ns	30ns
Minimum Clock HIGH Time	30ns	30ns
Minimum Clock Period	105ns	120ns

TABLE II

MAXIMUM COMBINATIONAL PROPAGATION DELAYS (all in ns, $C_L \leq 15pF$)

From Input \ To Output	Am2901DC, PC (0°C to +70°C; 5V ±5%)								Am2901DM, FM (−55°C to +125°C; 5V ±10%)							
	Y	F_3	C_{n+4}	$\overline{G}, \overline{P}$	F=0 R_L= 470	OVR	Shift Outputs RAM_0 RAM_3	Shift Outputs Q_0 Q_3	Y	F_3	C_{n+4}	$\overline{G}, \overline{P}$	F=0 R_L= 470	OVR	Shift Outputs RAM_0 RAM_3	Shift Outputs Q_0 Q_3
A, B	110	85	80	80	110	75	110	–	120	95	90	90	120	85	120	–
D (arithmetic mode)	100	70	70	70	100	60	95	–	110	80	75	75	110	65	105	–
D (I = X37) (Note 5)	60	50	–	–	60	–	60	–	65	55	–	–	65	–	65	–
C_n	55	35	30	–	50	40	55	–	60	40	30	–	55	45	60	–
I_{012}	85	65	65	65	80	65	80	–	90	70	70	70	85	70	85	–
I_{345}	70	55	60	60	70	60	65	–	75	60	65	65	75	65	70	–
I_{678}	55	–	–	–	–	–	45	45	60	–	–	–	–	–	50	50
$\overline{OE}$ Enable/Disable	40/25	–	–	–	–	–	–	–	40/25	–	–	–	–	–	–	–
A bypassing ALU (I = 2xx)	60	–	–	–	–	–	–	–	65	–	–	–	–	–	–	-
Clock ⎍ (Note 6)	115	85	100	100	110	95	105	60	125	95	110	110	120	105	115	65

SET-UP AND HOLD TIMES (all in ns) (Note 1)

TABLE III

From Input	Notes	Am2901DC, PC (0°C to +70°C, 5V ±5%)		Am2901DM, FM (−55°C to +125°C, 5V ±10%)	
		Set-Up Time	Hold Time	Set-Up Time	Hold Time
A, B Source	2, 4 3, 5	105 $t_{pw}L$ + 30	0	120 $t_{pw}L$ + 30	0
B Dest.	2, 4	$t_{pw}L$ + 15	0	$t_{pw}L$ +15	0
D (arithmetic mode)		100	0	110	0
D (I = X37) (Note 5)		60	0	65	0
C_n		55	0	60	0
I_{012}		85	0	90	0
I_{345}		70	0	75	0
I_{678}	4	$t_{pw}L$ + 15	0	$t_{pw}L$ + 15	0
$RAM_{0,3}$, $Q_{0,3}$		30	0	30	0

Notes: 1. *See Figure 11 and 12.*
2. If the B address is used as a source operand, allow for the "A, B source" set-up time; if it is used only for the destination address, use the "B dest." set-up time.
3. Where two numbers are shown, both must be met.
4. "$t_{pw}L$" is the clock LOW time.
5. DV0 is the fastest way to load the RAM from the D inputs. This function is obtained with I = 337.
6. Using Q register as source operand in arithmetic mode. Clock is not normally in critical speed path when Q is not a source.

SET-UP AND HOLD TIMES (minimum cycles from each input)

Set-up and hold times are defined relative to the clock LOW-to-HIGH edge. Inputs must be steady at all times from the set-up time prior to the clock until the hold time after the clock. The set-up times allow sufficient time to perform the correct operation on the correct data so that the correct ALU data can be written into one of the registers.

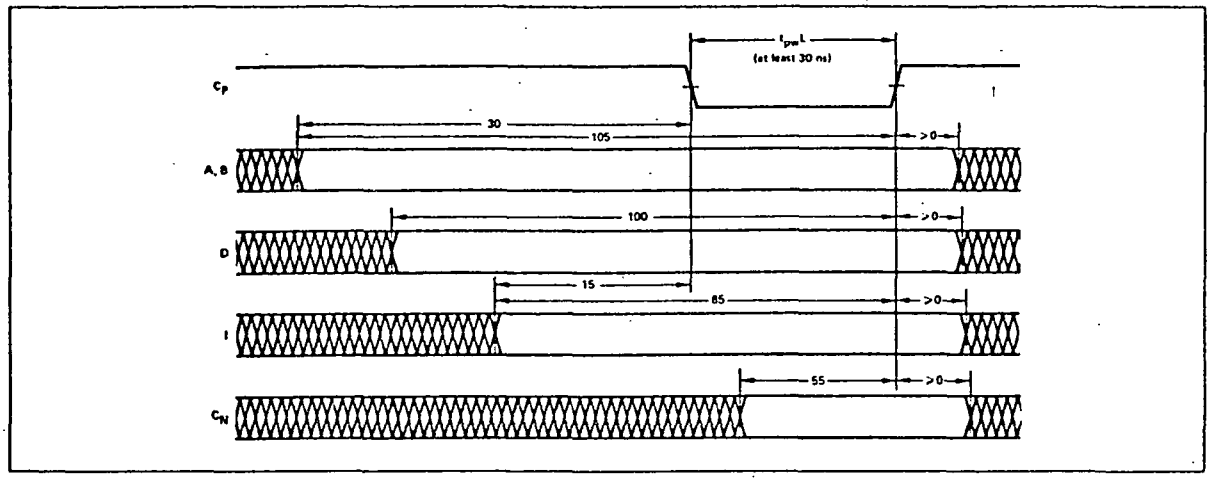
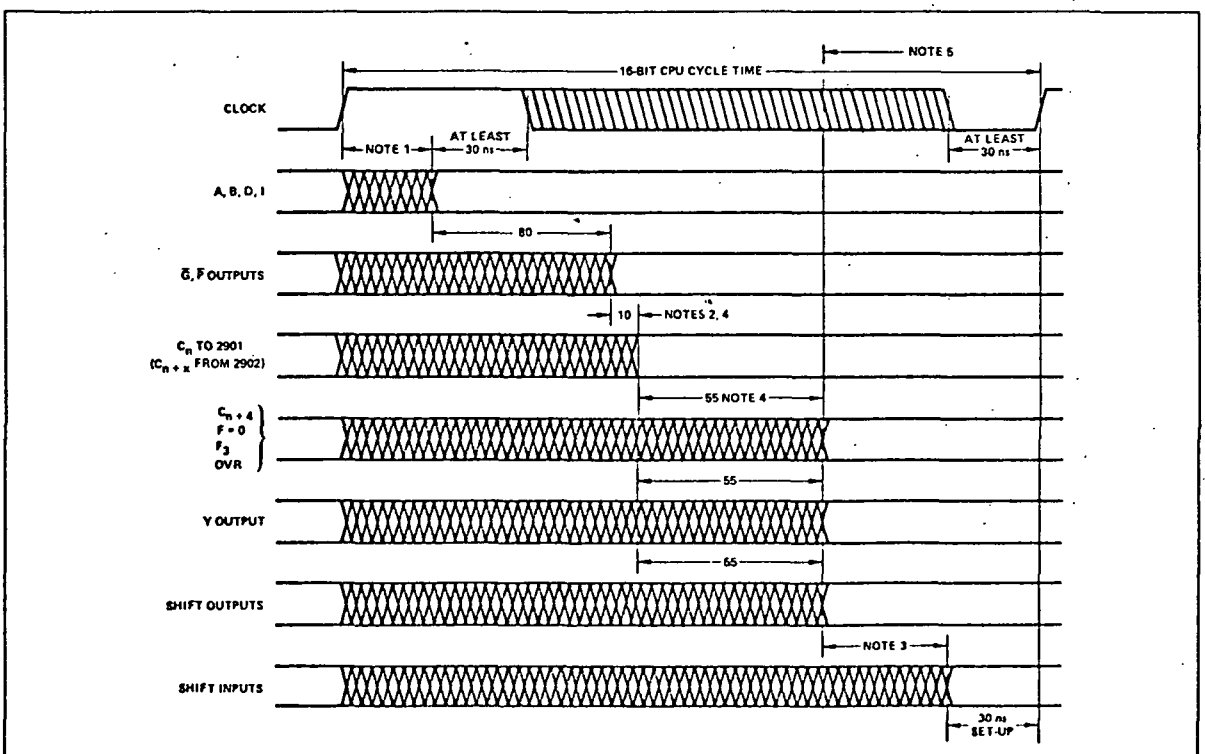


Figure 11. Minimum Cycle Times from Inputs. Numbers Shown are Minimum Data Stable Times for Am2901DC, in ns. See Table III for Detailed Information.



Notes: 1. This delay is the max. t_{pd} of the register containing A, B, D, and I. For the Am2918, use 13ns.
2. 10ns for look-ahead carry. For ripple carry over 16 bits use 2 x (C_n → C_{n+4}), or 60ns.
3. This is the delay associated with the multiplexer between the shift outputs and shift inputs on the Am2901s. See Figure 19.
4. Not applicable for logic operations.
5. Clock rising edge may occur here if add and shift do not occur on same cycle.

Figure 12. Switching Waveforms for 16-Bit System Assuming A, B, D and I are all Driven from Registers with the same Propagation Delay, Clocked by the Am2901 Clock.

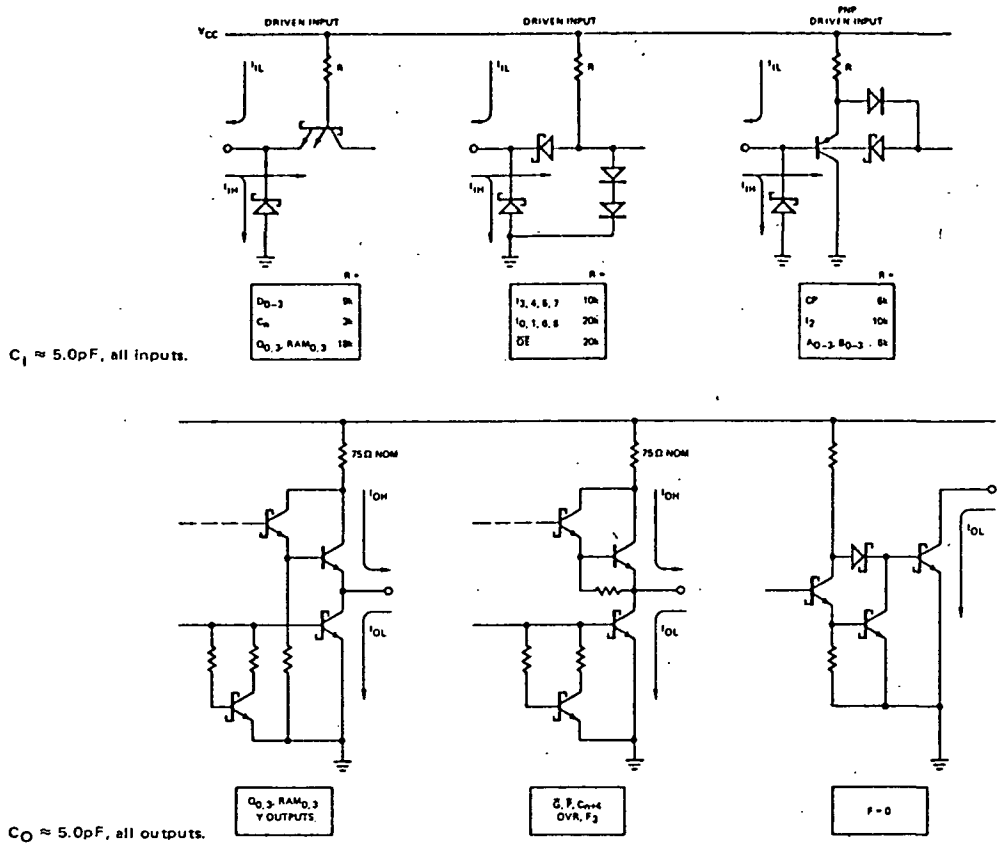


Figure 13. Am2901 Input/Output Current Interface Conditions.

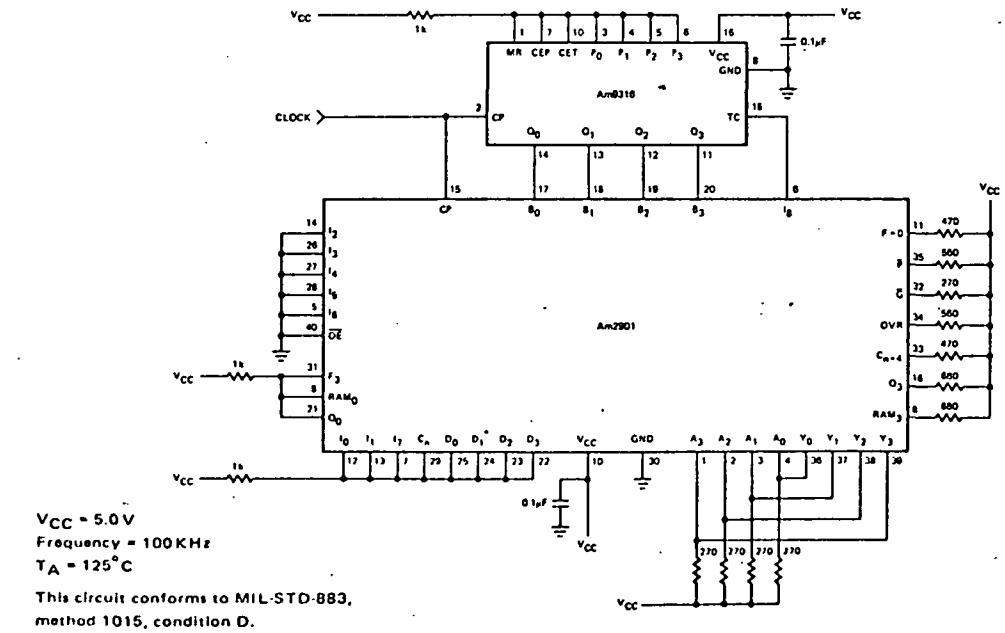


Figure 14. Am2901 Burn-In Circuit.

MC6800
MOS

(N-CHANNEL, SILICON-GATE)

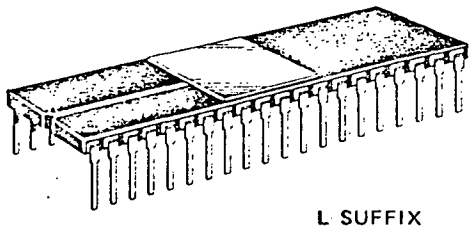
MICROPROCESSOR

MICROPROCESSING UNIT (MPU)

The MC6800 is a monolithic 8-bit microprocessor forming the central control function for Motorola's M6800 family. Compatible with TTL, the MC6800, as with all M6800 system parts, requires only one +5.0-volt power supply, and no external TTL devices for bus interface.

The MC6800 is capable of addressing 65K bytes of memory with its 16-bit address lines. The 8-bit data bus is bidirectional as well as 3-state, making direct memory addressing and multiprocessing applications realizable.

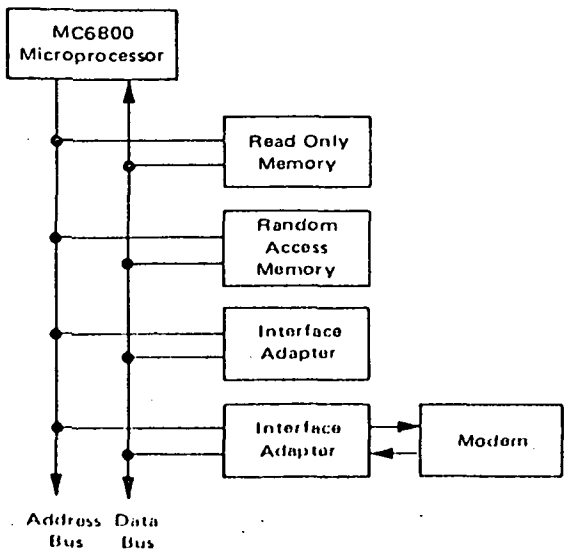
- Eight-Bit Parallel Processing
- Bi-Directional Data Bus
- Sixteen-Bit Address Bus – 65K Bytes of Addressing
- 72 Instructions – Variable Length
- Seven Addressing Modes – Direct, Relative, Immediate, Indexed, Extended, Implied and Accumulator
- Variable Length Stack
- Vectored Restart
- Maskable Interrupt Vector
- Separate Non-Maskable Interrupt – Internal Registers Saved In Stack
- Six Internal Registers – Two Accumulators, Index Register, Program Counter, Stack Pointer and Condition Code Register
- Direct Memory Addressing (DMA) and Multiple Processor Capability
- Clock Rates as High as 1 MHz
- Simple Bus Interface Without TTL
- Halt and Single Instruction Execution Capability



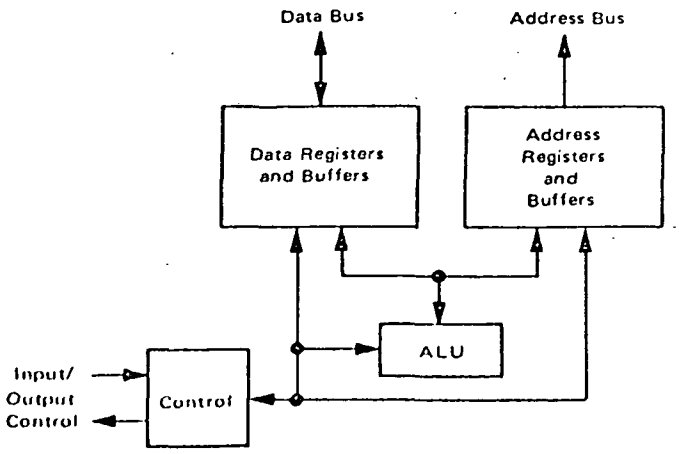
L SUFFIX
CERAMIC PACKAGE
CASE 715

NOT SHOWN: P SUFFIX
PLASTIC PACKAGE
CASE 711

M6800 MICROCOMPUTER FAMILY
BLOCK DIAGRAM



MC6800 MICROPROCESSOR
BLOCK DIAGRAM



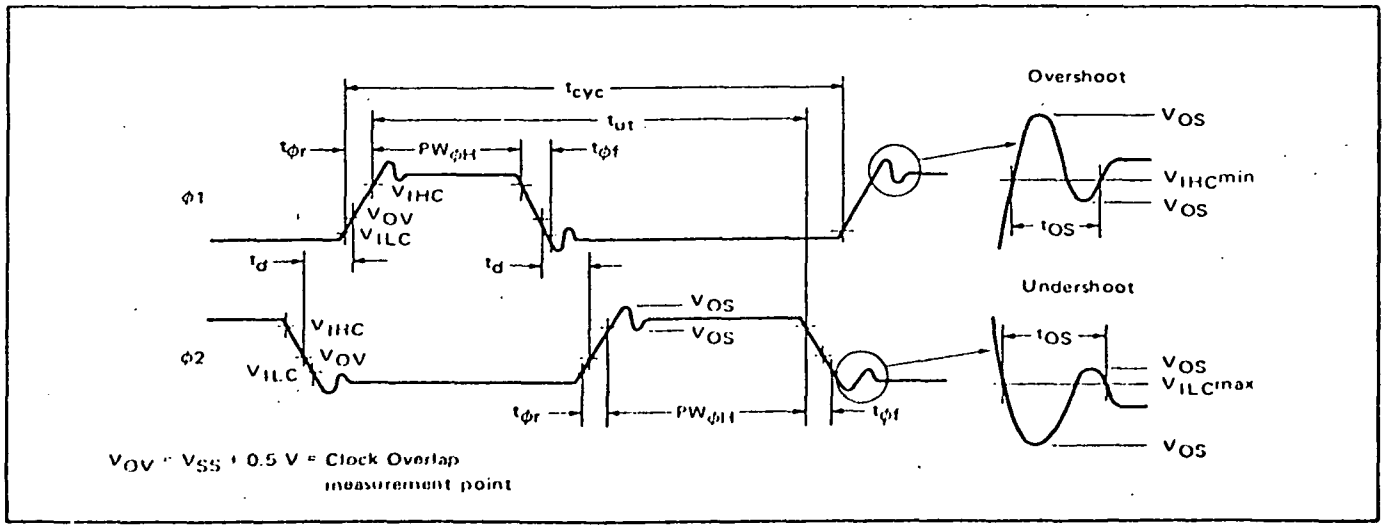
ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 V ± 5%, V_{SS} = 0, T_A = 0 to 70°C unless otherwise noted.)

Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage	Logic	V_{IH}	V_{SS} + 2.0	–	V_{CC}	Vdc
	$\phi1,\phi2$	V_{IHC}	V_{CC} – 0.3	–	V_{CC} + 0.1	
Input Low Voltage	Logic	V_{IL}	V_{SS} – 0.3	–	V_{SS} + 0.8	Vdc
	$\phi1,\phi2$	V_{ILC}	V_{SS} – 0.1	–	V_{SS} + 0.3	
Clock Overshoot/Undershoot – Input High Level		V_{OS}	V_{CC} – 0.5	–	V_{CC} + 0.5	Vdc
– Input Low Level			V_{SS} – 0.5	–	V_{SS} + 0.5	
Input Leakage Current		I_{in}				μAdc
(V_{in} = 0 to 5.25 V, V_{CC} = max)	Logic*		–	1.0	2.5	
(V_{in} = 0 to 5.25 V, V_{CC} = 0.0 V)	$\phi1,\phi2$		–	–	100	
Three-State (Off State) Input Current	D0-D7	I_{TSI}	–	2.0	10	μAdc
(V_{in} 0.4 to 2.4 V, V_{CC} = max)	A0-A15,R/W		–	–	100	
Output High Voltage		V_{OH}				Vdc
(I_{Load} = –205 μAdc, V_{CC} = min)	D0-D7		V_{SS} + 2.4	–	–	
(I_{Load} = –145 μAdc, V_{CC} = min)	A0-A15,R/W,VMA		V_{SS} + 2.4	–	–	
(I_{Load} = –100 μAdc, V_{CC} = min)	BA		V_{SS} + 2.4	–	–	
Output Low Voltage (I_{Load} = 1.6 mAdc, V_{CC} = min)		V_{OL}	–	–	V_{SS} + 0.4	Vdc
Power Dissipation		P_D	–	0.600	1.2	W
Capacitance #	$\phi1,\phi2$	C_{in}	80	120	160	pF
(V_{in} = 0, T_A = 25°C, f = 1.0 MHz)	TSC		–	–	15	
	DBE		–	7.0	10	
	D0-D7		–	10	12.5	
	Logic Inputs		–	6.5	8.5	
	A0-A15,R/W,VMA	C_{out}	–	–	12	pF
Frequency of Operation		f	0.1	–	1.0	MHz
Clock Timing (Figure 1)						
Cycle Time		t_{cyc}	1.0	–	10	μs
Clock Pulse Width		$PW_{\phi H}$				ns
(Measured at V_{CC} – 0.3 V)	$\phi1$		430	–	4500	
	$\phi2$		450	–	4500	
Total $\phi1$ and $\phi2$ Up Time		t_{ut}	940	–	–	ns
Rise and Fall Times	$\phi1,\phi2$	$t_{\phi r}$, $t_{\phi f}$	5.0	–	50	ns
(Measured between V_{SS} + 0.3 V and V_{CC} – 0.3 V)						
Delay Time or Clock Separation		t_d	0	–	9100	ns
(Measured at V_{OV} = V_{SS} + 0.5 V)						
Overshoot Duration		t_{OS}	0	–	40	ns

*Except $\overline{IRQ}$ and $\overline{NMI}$, which require 3 kΩ pullup load resistors for wire-OR capability at optimum operation.

#Capacitances are periodically sampled rather than 100% tested.

FIGURE 1 – CLOCK TIMING WAVEFORM



V_{OV} = V_{SS} + 0.5 V = Clock Overlap measurement point

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	Vdc
Input Voltage	V_{in}	−0.3 to +7.0	Vdc
Operating Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C
Thermal Resistance	θ_{JA}	70	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit.

READ/WRITE TIMING Figures 2 and 3, f = 1.0 MHz, Load Circuit of Figure 6.

Characteristic	Symbol	Min	Typ	Max	Unit
Address Delay	t_{AD}	–	220	300	ns
Peripheral Read Access Time $t_{acc} = t_{ut} - (t_{AD} + t_{DSR})$	t_{acc}	–	–	540	ns
Data Setup Time (Read)	t_{DSR}	100	–	–	ns
Input Data Hold Time	t_H	10	–	–	ns
Output Data Hold Time	t_H	10	25	–	ns
Address Hold Time (Address, R/W, VMA)	t_{AH}	50	75	–	ns
Enable High Time for DBE Input	t_{EH}	450	–	–	ns
Data Delay Time (Write)	t_{DDW}	–	165	225	ns
Processor Controls*					
Processor Control Setup Time	t_{PCS}	200	–	–	ns
Processor Control Rise and Fall Time	t_{PCr}, t_{PCf}	–	–	100	ns
Bus Available Delay	t_{BA}	–	–	300	ns
Three State Enable	t_{TSE}	–	–	40	ns
Three State Delay	t_{TSD}	–	–	700	ns
Data Bus Enable Down Time During $\phi 1$ Up Time (Figure 3)	$t_{\overline{DBE}}$	150	–	–	ns
Data Bus Enable Delay (Figure 3)	t_{DBED}	300	–	–	ns
Data Bus Enable Rise and Fall Times (Figure 3)	t_{DBEr}, t_{DBEf}	–	–	25	ns

*Additional information is given in Figures 12 through 16 of the Family Characteristics – see pages 17 through 20.

FIGURE 2 – READ DATA FROM MEMORY OR PERIPHERALS

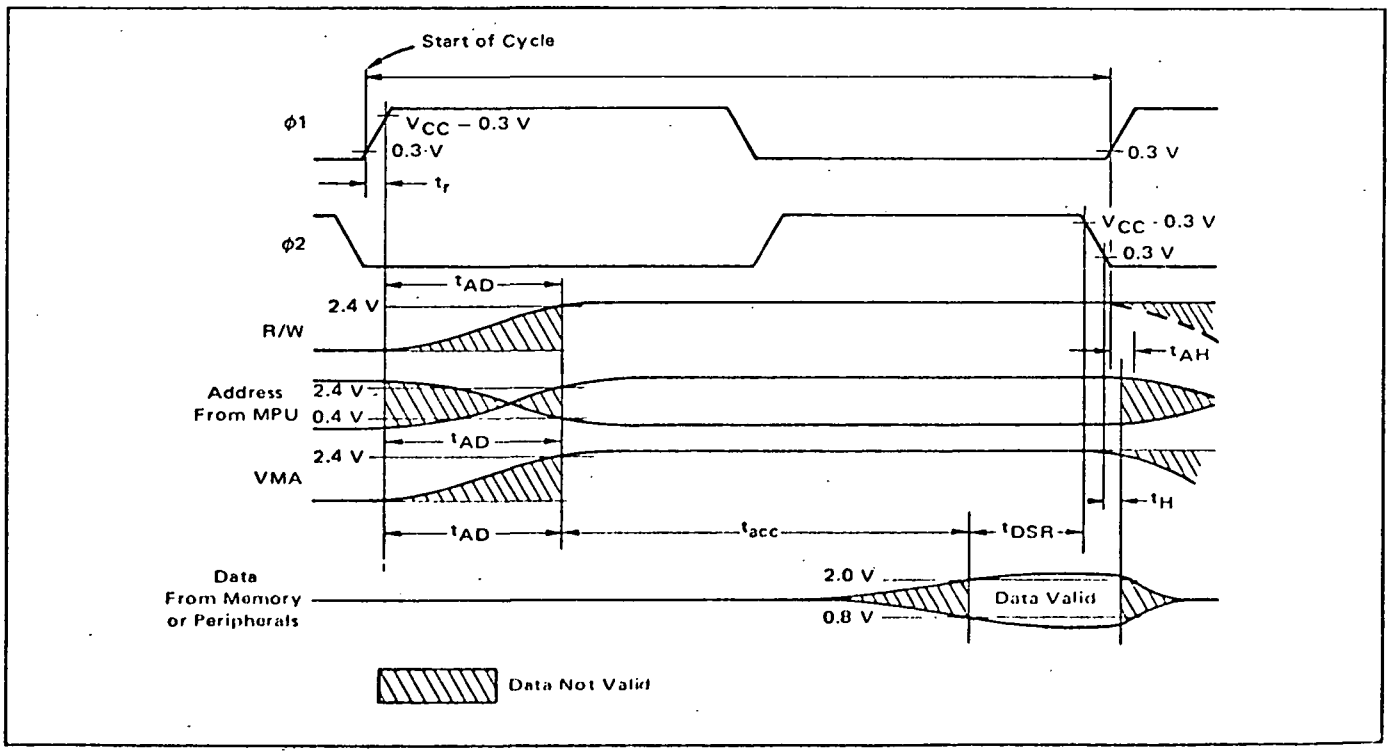


FIGURE 3 – WRITE IN MEMORY OR PERIPHERALS

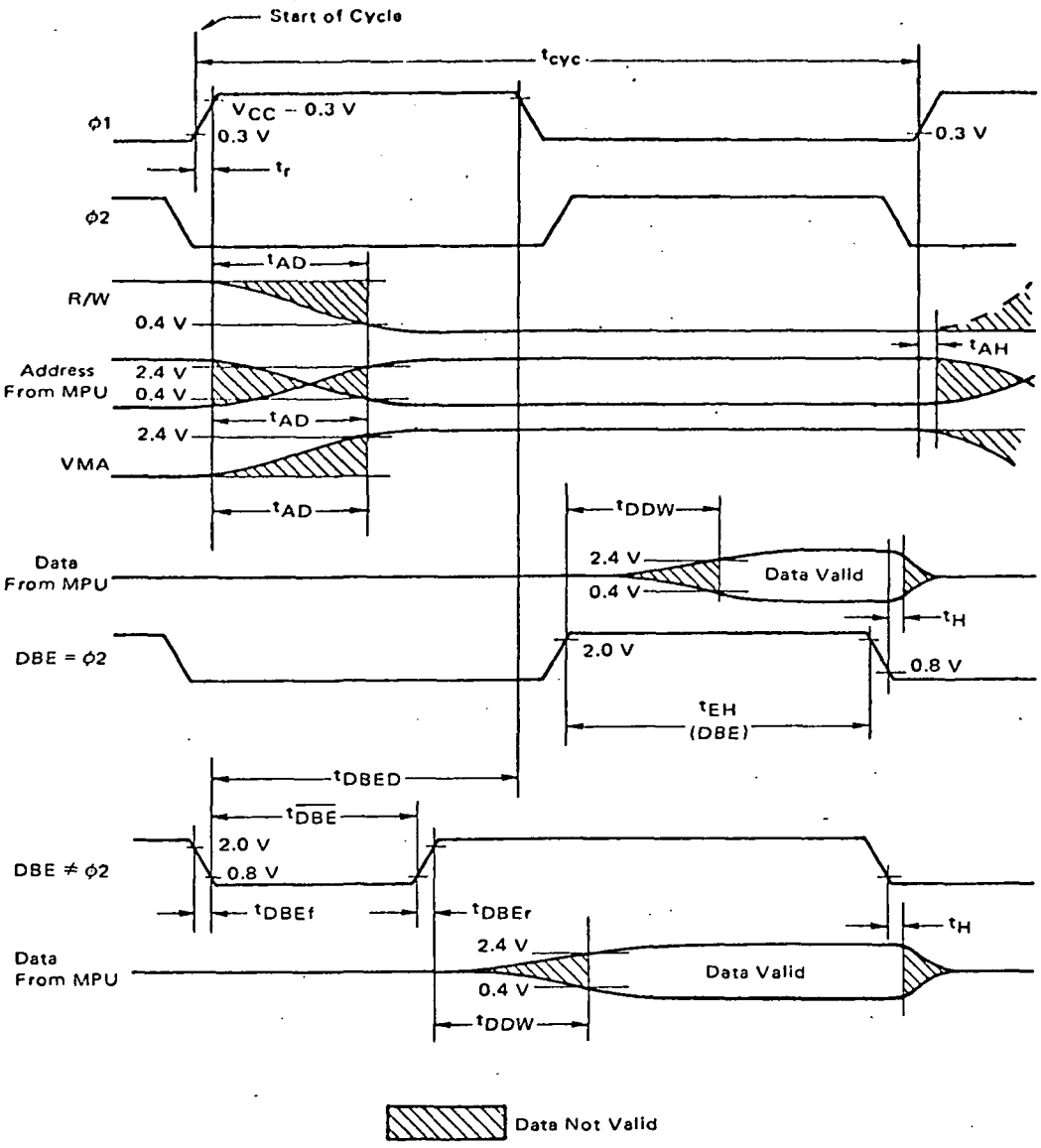


FIGURE 4 – TYPICAL DATA BUS OUTPUT DELAY versus CAPACITIVE LOADING

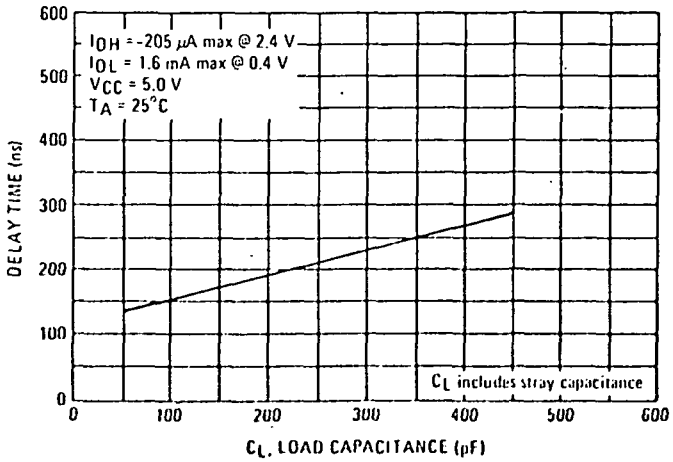


FIGURE 5 – TYPICAL READ/WRITE, VMA, AND ADDRESS OUTPUT DELAY versus CAPACITIVE LOADING

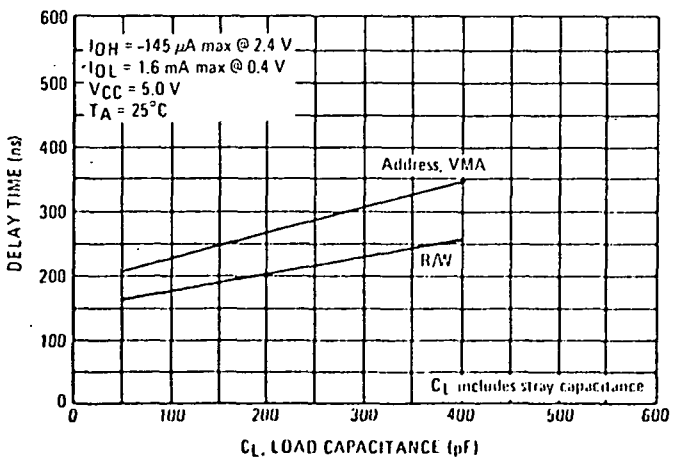
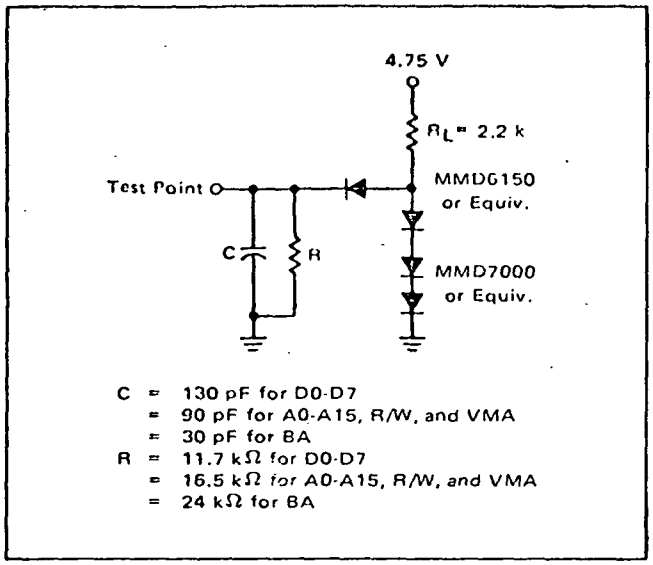


FIGURE 6 – BUS TIMING TEST LOAD



C = 130 pF for D0-D7
= 90 pF for A0-A15, R/W, and VMA
= 30 pF for BA
R = 11.7 kΩ for D0-D7
= 16.5 kΩ for A0-A15, R/W, and VMA
= 24 kΩ for BA

TYPICAL POWER SUPPLY CURRENT

FIGURE 7 – VARIATIONS WITH FREQUENCY

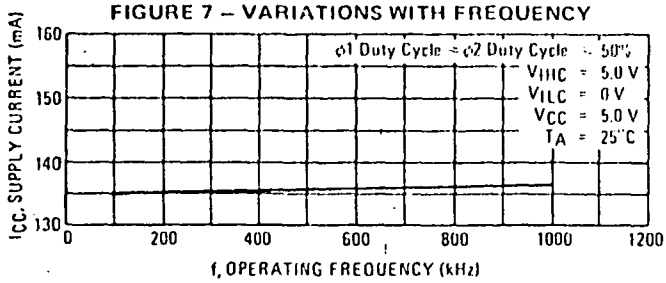
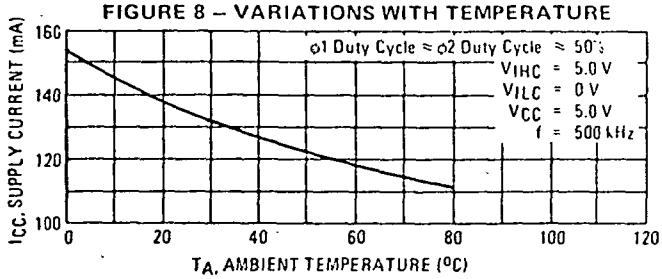
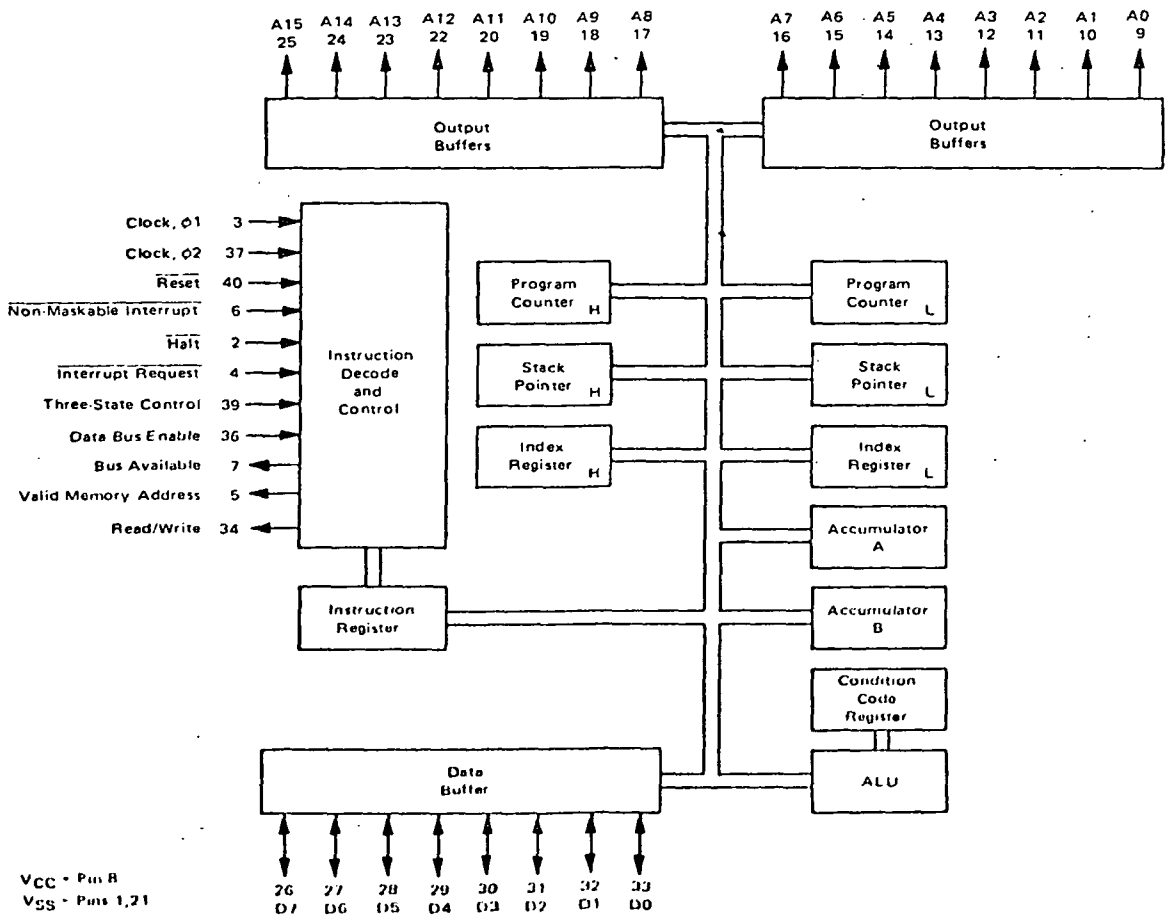


FIGURE 8 – VARIATIONS WITH TEMPERATURE



EXPANDED BLOCK DIAGRAM



Proper operation of the MPU requires that certain control and timing signals be provided to accomplish specific functions and that other signal lines be monitored to determine the state of the processor.

Clocks Phase One and Phase Two ($\phi 1, \phi 2$) -- Two pins are used for a two-phase non-overlapping clock that runs at the V_{CC} voltage level.

Address Bus (A0-A15) -- Sixteen pins are used for the address bus. The outputs are three-state bus drivers capable of driving one standard TTL load and 130 pF. When the output is turned off, it is essentially an open circuit. This permits the MPU to be used in DMA applications.

Data Bus (D0-D7) -- Eight pins are used for the data bus. It is bi-directional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130 pF.

$\overline{\text{Halt}}$ -- When this input is in the low state, all activity in the machine will be halted. This input is level sensitive. In the halt mode, the machine will stop at the end of an instruction, Bus Available will be at a one level, Valid Memory Address will be at a zero, and all other three-state lines will be in the three-state mode.

Transition of the $\overline{\text{Halt}}$ line must not occur during the last 250 ns of phase one. To insure single instruction operation, the $\overline{\text{Halt}}$ line must go high for one Clock cycle.

Three-State Control (TSC) -- This input causes all of the address lines and the Read/Write line to go into the off or high impedance state. This state will occur 700 ns after TSC = 2.0 V. The Valid Memory Address and Bus Available signals will be forced low. The data bus is not affected by TSC and has its own enable (Data Bus Enable). In DMA applications, the Three-State Control line should be brought high on the leading edge of the Phase One Clock. The $\phi 1$ clock must be held in the high state and the $\phi 2$ in the low state for this function to operate properly. The address bus will then be available for other devices to directly address memory. Since the MPU is a dynamic device, it can be held in this state for only 4.5 μs or destruction of data will occur in the MPU.

Read/Write (R/W) -- This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read (high) or Write (low) state. The normal standby state of this signal is Read (high). Three-State Control going high will turn Read/Write to the off (high impedance) state. Also, when the processor is halted, it will be in the off state. This output is capable of driving one standard TTL load and 90 pF.

Valid Memory Address (VMA) -- This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not three-state. One standard TTL load and 90 pF may be directly driven by this active high signal.

Data Bus Enable (DBE) -- This input is the three-state control signal for the MPU data bus and will enable the bus drivers when in the high state. This input is TTL compatible; however in normal operation, it would be driven by the phase two clock. During an MPU read cycle, the data bus drivers will be disabled internally. When it is desired that another device control the data bus such as in Direct Memory Access (DMA) applications, DBE should be held low.

Bus Available (BA) -- The Bus Available signal will normally be in the low state; when activated, it will go to the high state indicating that the microprocessor has stopped and that the address bus is available. This will occur if the $\overline{\text{Halt}}$ line is in the low state or the processor is in the WAIT state as a result of the execution of a WAIT instruction. At such time, all three-state output drivers will go to their off state and other outputs to their normally inactive level. The processor is removed from the WAIT state by the occurrence of a maskable (mask bit I = 0) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30 pF.

$\overline{\text{Interrupt Request}}$ ($\overline{\text{IRQ}}$) -- This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory.

The $\overline{\text{Halt}}$ line must be in the high state for interrupts to be serviced. Interrupts will be latched internally while $\overline{\text{Halt}}$ is low.

The $\overline{\text{IRQ}}$ has a high impedance pullup device internal to the chip; however a 3 kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

$\overline{\text{Reset}}$ -- This input is used to reset and start the MPU from a power down condition, resulting from a power failure or an initial start-up of the processor. If a high level is detected on the input, this will signal the MPU to begin the restart sequence. This will start execution of a routine to initialize the processor from its reset condition. All the higher order address lines will be forced high. For the restart, the last two (FFFE, FFFF) locations in memory will be used to load the program that is addressed by the program counter. During the restart routine, the interrupt mask bit is set and must be reset before the MPU can be interrupted by $\overline{\text{IRQ}}$.

Figure 9 shows the initialization of the microprocessor after restart. $\overline{\text{Reset}}$ must be held low for at least eight clock periods after V_{CC} reaches 4.75 volts. If $\overline{\text{Reset}}$ goes high prior to the leading edge of $\phi 2$, on the next $\phi 1$ the first restart memory vector address (FFFE) will appear on the address lines. This location should contain the higher order eight bits to be stored into the program counter. Following, the next address FFFF should contain the lower order eight bits to be stored into the program counter.

Non-Maskable Interrupt ($\overline{\text{NMI}}$) – A low-going edge on this input requests that a non-mask-interrupt sequence be generated within the processor. As with the $\overline{\text{Interrupt Request}}$ signal, the processor will complete the current instruction that is being executed before it recognizes the $\overline{\text{NMI}}$ signal. The interrupt mask bit in the Condition Code Register has no effect on $\overline{\text{NMI}}$.

The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFFC and FFFD. An address loaded at these locations causes the MPU to branch to a non-maskable interrupt routine in memory.

$\overline{\text{NMI}}$ has a high impedance pullup resistor internal to the chip; however a 3 kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

Inputs $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ are hardware interrupt lines that are sampled during $\phi 2$ and will start the interrupt routine on the $\phi 1$ following the completion of an instruction.

Figure 10 is a flow chart describing the major decision paths and interrupt vectors of the microprocessor. Table 1 gives the memory map for interrupt vectors.

FIGURE 9 – INITIALIZATION OF MPU AFTER RESTART

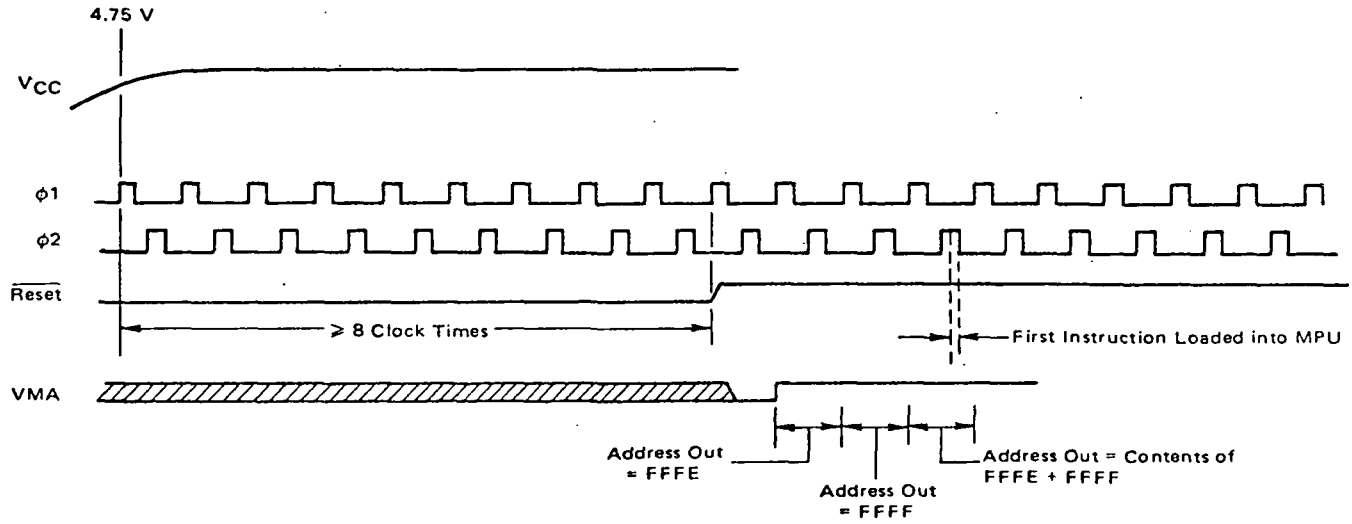
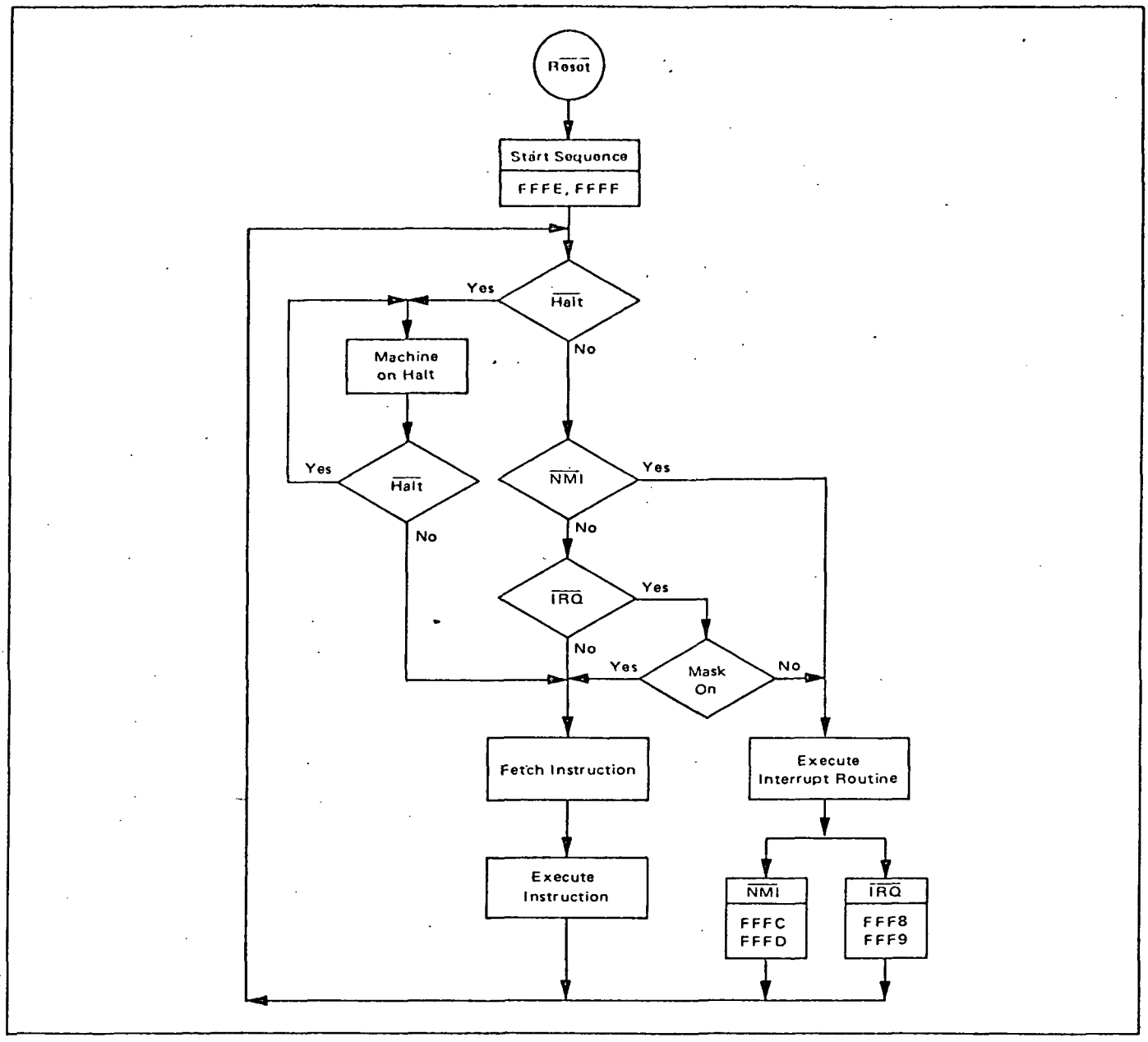


TABLE 1 – MEMORY MAP FOR INTERRUPT VECTORS

Vector MS	Vector LS	Description
FFFE	FFFF	Restart
FFFC	FFFD	Non-maskable Interrupt
FFFA	FFFB	Software Interrupt
FFF8	FFF9	Interrupt Request

FIGURE 10 – MPU FLOW CHART



MPU REGISTERS

The MPU has three 16-bit registers and three 8-bit registers available for use by the programmer (Figure 11).

Program Counter – The program counter is a two byte (16-bits) register that points to the current program address.

Stack Pointer – The stack pointer is a two byte register that contains the address of the next available location in an external push-down/pop-up stack. This stack is normally a random access Read/Write memory that may have any location (address) that is convenient. In those applications that require storage of information in the stack when power is lost, the stack must be non-volatile.

Index Register – The index register is a two byte register that is used to store data or a sixteen bit memory address for the Indexed mode of memory addressing.

Accumulators – The MPU contains two 8-bit accumulators that are used to hold operands and results from an arithmetic logic unit (ALU).

FIGURE 11 – PROGRAMMING MODEL OF THE MICROPROCESSING UNIT

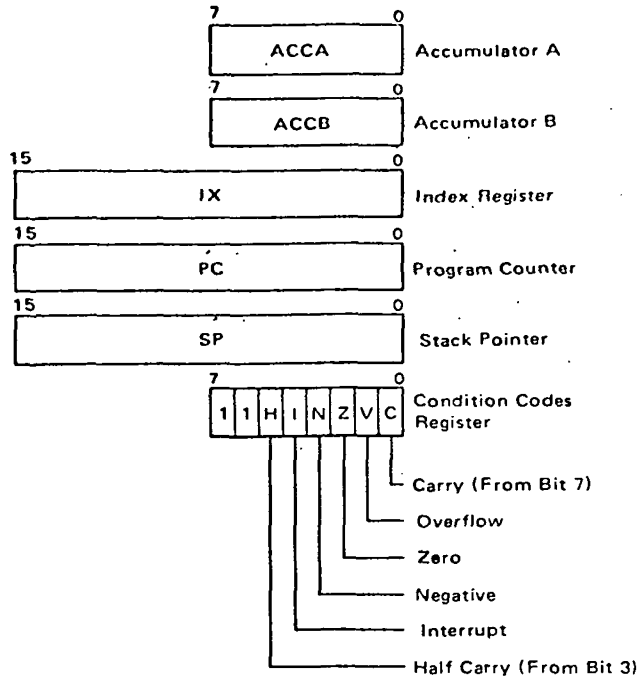
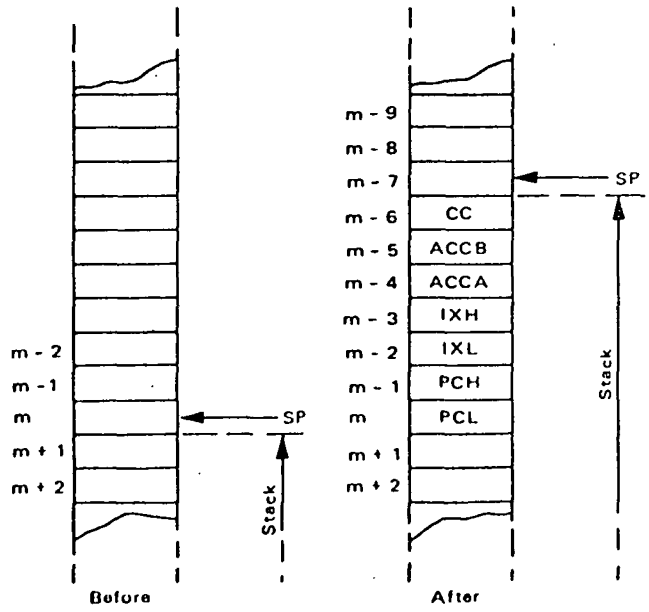


FIGURE 12 – SAVING THE STATUS OF THE MICROPROCESSOR IN THE STACK

SP = Stack Pointer
CC = Condition Codes (Also called the Processor Status Byte)
ACCB = Accumulator B
ACCA = Accumulator A
IXH = Index Register, Higher Order 8 Bits
IXL = Index Register, Lower Order 8 Bits
PCH = Program Counter, Higher Order 8 Bits
PCL = Program Counter, Lower Order 8 Bits



Condition Code Register – The condition code register indicates the results of an Arithmetic Logic Unit operation: Negative (N), Zero (Z), Overflow (V), Carry from bit 7 (C), and half carry from bit 3 (H). These bits of the Condition Code Register are used as testable conditions for the conditional branch instructions. Bit 4 is the interrupt mask bit (I). The unused bits of the Condition Code Register (b6 and b7) are ones.

Figure 12 shows the order of saving the microprocessor status within the stack.

MPU INSTRUCTION SET

The MC6800 has a set of 72 different instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions (Tables 2 thru 6).

MPU ADDRESSING MODES

The MC6800 eight-bit microprocessing unit has seven address modes that can be used by a programmer, with the addressing mode a function of both the type of instruction and the coding within the instruction. A summary of the addressing modes for a particular instruction can be found in Table 7 along with the associated instruction execution time that is given in machine cycles. With a clock frequency of 1 MHz, these times would be microseconds.

Accumulator (ACCX) Addressing – In accumulator only addressing, either accumulator A or accumulator B is specified. These are one-byte instructions.

Immediate Addressing – In immediate addressing, the operand is contained in the second byte of the instruction except LDS and LDX which have the operand in the second and third bytes of the instruction. The MPU addresses this location when it fetches the immediate instruction for execution. These are two or three-byte instructions.

Direct Addressing – In direct addressing, the address of the operand is contained in the second byte of the instruction. Direct addressing allows the user to directly address the lowest 256 bytes in the machine i.e., locations zero through 255. Enhanced execution times are achieved by storing data in these locations. In most configurations, it should be a random access memory. These are two-byte instructions.

Extended Addressing – In extended addressing, the address contained in the second byte of the instruction is used as the higher eight-bits of the address of the operand. The third byte of the instruction is used as the lower eight-bits of the address for the operand. This is an absolute address in memory. These are three-byte instructions.

Indexed Addressing – In indexed addressing, the address contained in the second byte of the instruction is added to the index register's lowest eight bits in the MPU. The carry is then added to the higher order eight bits of the index register. This result is then used to address memory. The modified address is held in a temporary address register so there is no change to the index register. These are two-byte instructions.

Implied Addressing – In the implied addressing mode the instruction gives the address (i.e., stack pointer, index register, etc.). These are one-byte instructions.

Relative Addressing – In relative addressing, the address contained in the second byte of the instruction is added to the program counter's lowest eight bits plus two. The carry or borrow is then added to the high eight bits. This allows the user to address data within a range of -125 to +129 bytes of the present instruction. These are two-byte instructions.

TABLE 2 – MICROPROCESSOR INSTRUCTION SET – ALPHABETIC SEQUENCE

ABA	Add Accumulators
ADC	Add with Carry
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
BCC	Branch if Carry Clear
BCS	Branch if Carry Set
BEQ	Branch if Equal to Zero
BGE	Branch if Greater or Equal Zero
BGT	Branch if Greater than Zero
BHI	Branch if Higher
BIT	Bit Test
BLE	Branch if Less or Equal
BLS	Branch if Lower or Same
BLT	Branch if Less than Zero
BMI	Branch if Minus
BNE	Branch if Not Equal to Zero
BPL	Branch if Plus
BRA	Branch Always
BSR	Branch to Subroutine
BVC	Branch if Overflow Clear
BVS	Branch if Overflow Set
CBA	Compare Accumulators
CLC	Clear Carry
CLI	Clear Interrupt Mask
CLR	Clear
CLV	Clear Overflow
CMP	Compare
COM	Complement
CPX	Compare Index Register
DAA	Decimal Adjust
DEC	Decrement
DES	Decrement Stack Pointer
DEX	Decrement Index Register
EOR	Exclusive OR
INC	Increment
INS	Increment Stack Pointer
INX	Increment Index Register
JMP	Jump
JSR	Jump to Subroutine
LDA	Load Accumulator
LDS	Load Stack Pointer
LDX	Load Index Register
LSR	Logical Shift Right
NEG	Negate
NOP	No Operation
ORA	Inclusive OR Accumulator
PSH	Push Data
PUL	Pull Data
ROL	Rotate Left
ROR	Rotate Right
RTI	Return from Interrupt
RTS	Return from Subroutine
SBA	Subtract Accumulators
SBC	Subtract with Carry
SEC	Set Carry
SEI	Set Interrupt Mask
SEV	Set Overflow
STA	Store Accumulator
STS	Store Stack Register
STX	Store Index Register
SUB	Subtract
SWI	Software Interrupt
TAB	Transfer Accumulators
TAP	Transfer Accumulators to Condition Code Reg.
TBA	Transfer Accumulators
TPA	Transfer Condition Code Reg. to Accumulator
TST	Test
TSX	Transfer Stack Pointer to Index Register
TXS	Transfer Index Register to Stack Pointer
WAI	Wait for Interrupt

TABLE 3 – ACCUMULATOR AND MEMORY INSTRUCTIONS

		ADDRESSING MODES													BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	COND. CODE REG.					
		IMMED			DIRECT			INDEX			EXTND			IMPLIED			5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	H	I	N	Z	V	C
Add	ADDA	3B	2	2	9B	3	2	AB	5	2	BB	4	3				↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				•	•	R	S	R	R
	CLRA													4F	2	1	•	•	R	S	R	R
	CLRB													5F	2	1	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				•	•	↕	↕	R	S
	COMA													43	2	1	•	•	↕	↕	R	S
	COMB													53	2	1	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				•	•	↕	↕	①	②
	NEGA													40	2	1	•	•	↕	↕	①	②
	NEGB													50	2	1	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	•	•	↕	↕	↕	③
																	Converts Binary Add. of BCD Characters into BCD Format					
Decrement	DEC							6A	7	2	7A	6	3				M – 1 → M	•	•	↕	↕	4	•
	DECA													4A	2	1	A – 1 → A	•	•	↕	↕	4	•
	DECB													5A	2	1	B – 1 → B	•	•	↕	↕	4	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP – 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP – 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B	•	•	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A – M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B – M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A – B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A – M – C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B – M – C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M – 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A – 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B – 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

LEGEND:

OP	Operation Code (Hexadecimal).	+	Boolean Inclusive OR.
~	Number of MPU Cycles.	⊕	Boolean Exclusive OR.
#	Number of Program Bytes.	M	Complement of M.
+	Arithmetic Plus.	→	Transfer Into.
–	Arithmetic Minus.	0	Bit Zero.
·	Boolean AND.	00	Byte Zero.

M_{SP} Contents of memory location pointed to by Stack Pointer.

Note: Accumulator addressing mode instructions are included in the column for IMPLIED addressing.

CONDITION CODE SYMBOLS:

H	Half-carry from bit 3.
I	Interrupt mask
N	Negative (sign bit)
Z	Zero (byte)
V	Overflow, 2's complement
C	Carry from bit 7
R	Reset Always
S	Set Always
↕	Test and set if true, cleared otherwise
•	Not Affected

TABLE 4 – INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

POINTER OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN/ARITHMETIC OPERATION	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M+1)$	•	•	⑦	↕	⑧	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	⑨	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	⑨	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

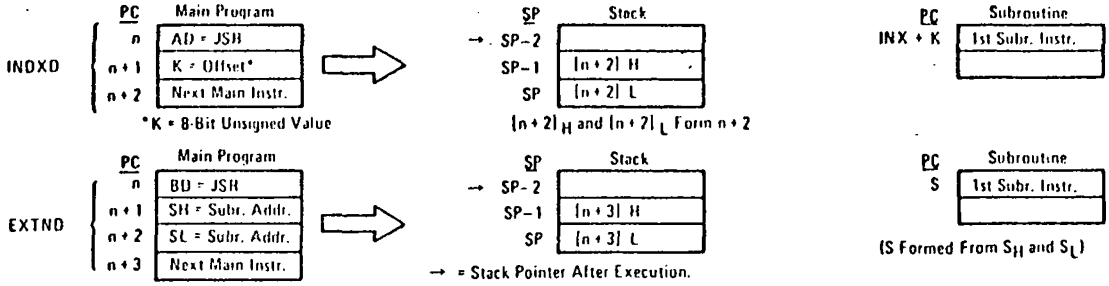
TABLE 5 – JUMP AND BRANCH INSTRUCTIONS

OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BRANCH TEST	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										$N \oplus V = 0$	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										$Z + (N \oplus V) = 0$	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										$Z + (N \oplus V) = 1$	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										$N \oplus V = 1$	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		⑩					
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•	•
Wait for Interrupt *	WAI										3E	9	1		•	⑪	•	•	•	•

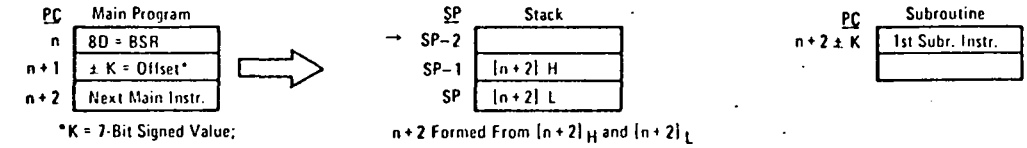
*WAI puts Address Bus, R/W, and Data Bus in the three-state mode while VMA is held low.

SPECIAL OPERATIONS

JSR, JUMP TO SUBROUTINE:

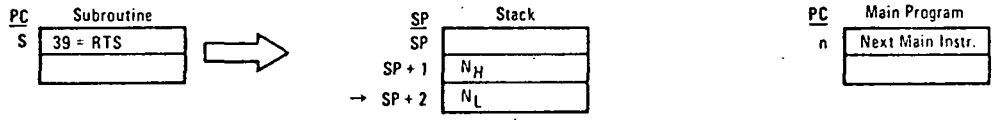


BSR, BRANCH TO SUBROUTINE:



JMP, JUMP:

RTS, RETURN FROM SUBROUTINE:



RTI, RETURN FROM INTERRUPT:

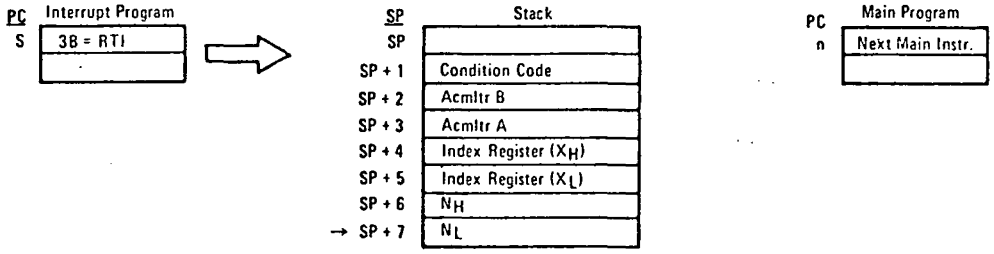


TABLE 6 – CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

OPERATIONS	MNEMONIC	IMPLIED OP	~	#	BOOLEAN OPERATION	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	A → CCR	(12)					
CCR → Acmltr A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

CONDITION CODE REGISTER NOTES: (Bit set if test is true and cleared otherwise)

1 (Bit V) Test: Result = 10000000?
2 (Bit C) Test: Result = 00000000?
3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set)
4 (Bit V) Test: Operand = 10000000 prior to execution?
5 (Bit V) Test: Operand = 01111111 prior to execution?
6 (Bit V) Test: Set equal to result of N⊕C after shift has occurred.
7 (Bit N) Test: Sign bit of most significant (MS) byte = 1?
8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes?
9 (Bit N) Test: Result less than zero? (Bit 15 = 1)
10 (All) Load Condition Code Register from Stack. (See Special Operations)
11 (Bit I) Set when interrupt occurs. If previously set, a Non Maskable Interrupt is required to exit the wait state.
12 (All) Set according to the contents of Accumulator A.

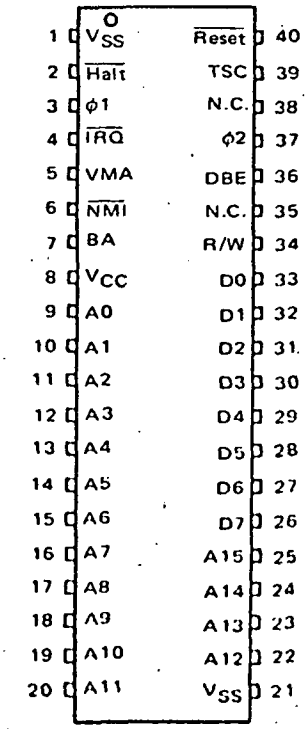
TABLE 7 – INSTRUCTION ADDRESSING MODES AND ASSOCIATED EXECUTION TIMES
(Times in Machine Cycles)

	(Dual Operand)	ACCX	Immediate	Direct	Extended	Indexed	Implied	Relative
ABA		•	•	•	•	•	2	•
ADC	x	•	2	3	4	5	•	•
ADD	x	•	2	3	4	5	•	•
AND	x	•	2	3	4	5	•	•
ASL		2	•	•	6	7	•	•
ASR		2	•	•	6	7	•	•
BCC		•	•	•	•	•	•	4
BCS		•	•	•	•	•	•	4
BEA		•	•	•	•	•	•	4
BGE		•	•	•	•	•	•	4
BGT		•	•	•	•	•	•	4
BHI		•	•	•	•	•	•	4
BIT	x	•	2	3	4	5	•	•
BLE		•	•	•	•	•	•	4
BLS		•	•	•	•	•	•	4
BLT		•	•	•	•	•	•	4
BMI		•	•	•	•	•	•	4
BNE		•	•	•	•	•	•	4
BPL		•	•	•	•	•	•	4
BRA		•	•	•	•	•	•	4
BSR		•	•	•	•	•	•	8
BVC		•	•	•	•	•	•	4
BVS		•	•	•	•	•	•	4
CBA		•	•	•	•	•	2	•
CLC		•	•	•	•	•	2	•
CLI		•	•	•	•	•	2	•
CLR		2	•	•	6	7	•	•
CLV		•	•	•	•	•	2	•
CMP	x	•	2	3	4	5	•	•
COM		2	•	•	6	7	•	•
CPX		•	3	4	5	6	•	•
DAA		•	•	•	•	•	2	•
DEC		2	•	•	6	7	•	•
DES		•	•	•	•	•	4	•
DEX		•	•	•	•	•	4	•
EOR	x	•	2	3	4	5	•	•

	(Dual Operand)	ACCX	Immediate	Direct	Extended	Indexed	Implied
INC		2	•	•	6	7	•
INS		•	•	•	•	•	4
INX		•	•	•	•	•	4
JMP		•	•	•	3	4	•
JSR		•	•	•	9	8	•
LDA	x	•	2	3	4	5	•
LDS		•	3	4	5	6	•
LDX		•	3	4	5	6	•
LSR		2	•	•	6	7	•
NEG		2	•	•	6	7	•
NOP		•	•	•	•	•	2
ORA	x	•	2	3	4	5	•
PSH		•	•	•	•	•	4
PUL		•	•	•	•	•	4
ROL		2	•	•	6	7	•
ROR		2	•	•	6	7	•
RTI		•	•	•	•	•	10
RTS		•	•	•	•	•	5
SBA		•	•	•	•	•	2
SBC	x	•	2	3	4	5	•
SEC		•	•	•	•	•	2
SEI		•	•	•	•	•	2
SEV		•	•	•	•	•	2
STA	x	•	•	4	5	6	•
STS		•	•	5	6	7	•
STX		•	•	5	6	7	•
SUB	x	•	2	3	4	5	•
SWI		•	•	•	•	•	12
TAB		•	•	•	•	•	2
TAP		•	•	•	•	•	2
TBA		•	•	•	•	•	2
TPA		•	•	•	•	•	2
TST		2	•	•	6	7	•
TSX		•	•	•	•	•	4
TSX		•	•	•	•	•	4
WAI		•	•	•	•	•	9

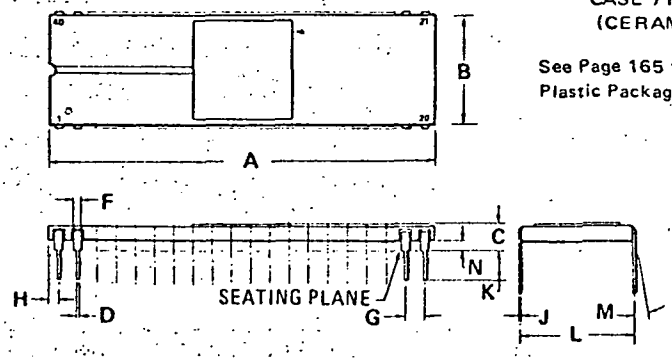
NOTE: Interrupt time is 12 cycles from the end of the instruction being executed, except following a WAI instruction. Then it is 4 cycles.

PIN ASSIGNMENT



PACKAGE DIMENSIONS
CASE 715-02
(CERAMIC)

See Page 165 for Plastic Package dimensions.



DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	50.29	51.31	1.980	2.020
B	14.86	15.62	0.585	0.615
C	2.54	4.19	0.100	0.165
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
L	14.60	15.37	0.575	0.605
M	–	10°	–	10°
N	0.51	1.57	0.020	0.060

NOTE:
1. LEADS, TRUE POSITIONED WITHIN 0.25 mm (0.010) DIA (AT SEATING PLANE), AT MAX. MAT'L CONDITION.

SUMMARY OF CYCLE BY CYCLE OPERATION

Table 8 provides a detailed description of the information present on the Address Bus, Data Bus, Valid Memory Address line (VMA), and the Read/Write line (R/W) during each cycle for each instruction.

This information is useful in comparing actual with expected results during debug of both software and hardware as the control program is executed. The information is categorized in groups according to Addressing Mode and Number of Cycles per instruction. (In general, instructions with the same Addressing Mode and Number of Cycles execute in the same manner; exceptions are indicated in the table.)

TABLE 8 – OPERATION SUMMARY

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
IMMEDIATE						
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Operand Data
CPX LDS LDX	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Operand Data (High Order Byte)
		3	1	Op Code Address + 2	1	Operand Data (Low Order Byte)
DIRECT						
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	1	Address of Operand	1	Operand Data
CPX LDS LDX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	1	Address of Operand	1	Operand Data (High Order Byte)
		4	1	Operand Address + 1	1	Operand Data (Low Order Byte)
STA	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Destination Address
		3	0	Destination Address	1	Irrelevant Data (Note 1)
		4	1	Destination Address	0	Data from Accumulator
STS STX	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	0	Address of Operand	1	Irrelevant Data (Note 1)
		4	1	Address of Operand	0	Register Data (High Order Byte)
		5	1	Address of Operand + 1	0	Register Data (Low Order Byte)
INDEXED						
JMP	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	1	Index Register Plus Offset	1	Operand Data
CPX LDS LDX	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	1	Index Register Plus Offset	1	Operand Data (High Order Byte)
		6	1	Index Register Plus Offset + 1	1	Operand Data (Low Order Byte)

TABLE 8 – OPERATION SUMMARY (Continued)

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
INDEXED (Continued)						
STA	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		6	1	Index Register Plus Offset	0	Operand Data
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	7	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	1	Index Register Plus Offset	1	Current Operand Data
		6	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		7	1/0 (Note 3)	Index Register Plus Offset	0	New Operand Data (Note 3)
STS STX	7	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		6	1	Index Register Plus Offset	0	Operand Data (High Order Byte)
		7	1	Index Register Plus Offset + 1	0	Operand Data (Low Order Byte)
JSR	8	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer – 2	1	Irrelevant Data (Note 1)
		7	0	Index Register	1	Irrelevant Data (Note 1)
		8	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
EXTENDED						
JMP	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Jump Address (High Order Byte)
		3	1	Op Code Address + 2	1	Jump Address (Low Order Byte)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data
CPX LDS LDX	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data (High Order Byte)
		5	1	Address of Operand + 1	1	Operand Data (Low Order Byte)
STA A STA B	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Destination Address (High Order Byte)
		3	1	Op Code Address + 2	1	Destination Address (Low Order Byte)
		4	0	Operand Destination Address	1	Irrelevant Data (Note 1)
		5	1	Operand Destination Address	0	Data from Accumulator
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Current Operand Data
		5	0	Address of Operand	1	Irrelevant Data (Note 1)
		6	1/0 (Note 3)	Address of Operand	0	New Operand Data (Note 3)

TABLE 8 – OPERATION SUMMARY (Continued)

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
EXTENDED (Continued)						
STS STX	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	0	Address of Operand	1	Irrelevant Data (Note 1)
		5	1	Address of Operand	0	Operand Data (High Order Byte)
		6	1	Address of Operand + 1	0	Operand Data (Low Order Byte)
JSR	9	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Subroutine (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
		4	1	Subroutine Starting Address	1	Op Code of Next Instruction
		5	1	Stack Pointer	0	Return Address (Low Order Byte)
		6	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		7	0	Stack Pointer – 2	1	Irrelevant Data (Note 1)
		8	0	Op Code Address + 2	1	Irrelevant Data (Note 1)
		9	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
INHERENT						
ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
DES DEX INS INX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Previous Register Contents	1	Irrelevant Data (Note 1)
		4	0	New Register Contents	1	Irrelevant Data (Note 1)
PSH	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	1	Stack Pointer	0	Accumulator Data
		4	0	Stack Pointer – 1	1	Accumulator Data
PUL	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer + 1	1	Operand Data from Stack
TSX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	0	New Index Register	1	Irrelevant Data (Note 1)
TXS	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Index Register	1	Irrelevant Data
		4	0	New Stack Pointer	1	Irrelevant Data
RTS	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 2)
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer + 1	1	Address of Next Instruction (High Order Byte)
		5	1	Stack Pointer + 2	1	Address of Next Instruction (Low Order Byte)

TABLE 8 – OPERATION SUMMARY (Continued)

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
INHERENT (Continued)						
WAI	9	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	1	Stack Pointer	0	Return Address (Low Order Byte)
		4	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		5	1	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	1	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	1	Stack Pointer – 4	0	Contents of Accumulator A
		8	1	Stack Pointer – 5	0	Contents of Accumulator B
		9	1	Stack Pointer – 6 (Note 4)	1	Contents of Cond. Code Register
RTI	10	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 2)
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer + 1	1	Contents of Cond. Code Register from Stack
		5	1	Stack Pointer + 2	1	Contents of Accumulator B from Stack
		6	1	Stack Pointer + 3	1	Contents of Accumulator A from Stack
		7	1	Stack Pointer + 4	1	Index Register from Stack (High Order Byte)
		8	1	Stack Pointer + 5	1	Index Register from Stack (Low Order Byte)
		9	1	Stack Pointer + 6	1	Next Instruction Address from Stack (High Order Byte)
		10	1	Stack Pointer + 7	1	Next Instruction Address from Stack (Low Order Byte)
SWI	12	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 1)
		3	1	Stack Pointer	0	Return Address (Low Order Byte)
		4	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		5	1	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	1	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	1	Stack Pointer – 4	0	Contents of Accumulator A
		8	1	Stack Pointer – 5	0	Contents of Accumulator B
		9	1	Stack Pointer – 6	0	Contents of Cond. Code Register
		10	0	Stack Pointer – 7	1	Irrelevant Data (Note 1)
		11	1	Vector Address FFFA (Hex)	1	Address of Subroutine (High Order Byte)
		12	1	Vector Address FFFB (Hex)	1	Address of Subroutine (Low Order Byte)
RELATIVE						
BCC BHI BNE BCS BLE BPL BEQ BLS BRA BGE BLT BVC BGT BMI BVS	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Branch Offset
		3	0	Op Code Address + 2	1	Irrelevant Data (Note 1)
		4	0	Branch Address	1	Irrelevant Data (Note 1)
BSR	8	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Branch Offset
		3	0	Return Address of Main Program	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer – 2	1	Irrelevant Data (Note 1)
		7	0	Return Address of Main Program	1	Irrelevant Data (Note 1)
		8	0	Subroutine Address	1	Irrelevant Data (Note 1)

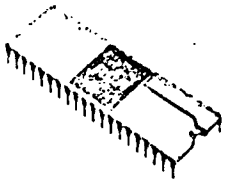
Note 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.
Note 2. Data is ignored by the MPU.
Note 3. For TST, VMA 0 and Operand data does not change.
Note 4. While the MPU is waiting for the interrupt, Bus Available will go high indicating the following states of the control lines: VMA is low: Address Bus, R/W, and Data Bus are all in the high impedance state.

RCA
Solid State
Division

Microprocessor Products

CDP1802D
CDP1802CD

Preliminary Data▲



40-Lead Dual-In-Line Ceramic Package (D)

CDP1802D
CDP1802CD H-1847

COSMAC Microprocessor

Features:

- Static silicon-gate CMOS circuitry—CD4000 series compatible
- Instruction fetch-execute time of 2.5/3.75 μs at V_{DD} = 10 V
- Compatible with CDP1801 software
- Full military-temperature range (−55 to +125°C)
- High noise immunity, wide operating-voltage range
- Single voltage supply
- No minimum clock frequency
- Low power
- TTL compatible
- Single-phase clock; optional on-chip crystal-controlled oscillator
- Simple control of reset, start, and pause
- 8-bit parallel organization with bidirectional data bus
- Any combination of standard RAM and ROM
- Memory addressing up to 65,536 bytes
- Flexible programmed I/O mode
- Program interrupt mode
- On-chip DMA
- Four I/O flag inputs directly tested by branch instructions
- Programmable output port
- 91 easy-to-use instructions
- 16 x 16 matrix of registers for use as multiple program counters, data pointers, or data registers

The RCA-CDP1802 is an LSI COS/MOS 8-bit register-oriented central-processing unit (CPU) designed for use as a general-purpose computing or control element in a wide range of stored-program systems or products.

It has the same basic COSMAC architecture as the CDP1801 microprocessor (see Fig. 5), but has an expanded instruction set, including a more powerful set of branch and ALU instructions, and enhanced hardware and performance features. The CDP1802 incorporates both the register and control chips of the CDP1801 on a single chip in a 40-lead hermetic dual-in-line ceramic package.

The CDP1802D is functionally identical to the CDP1802CD. The CDP1802D has a recommended operating voltage range of 3-12 volts; the CDP1802CD, a recommended operating voltage range of 4-6 volts.

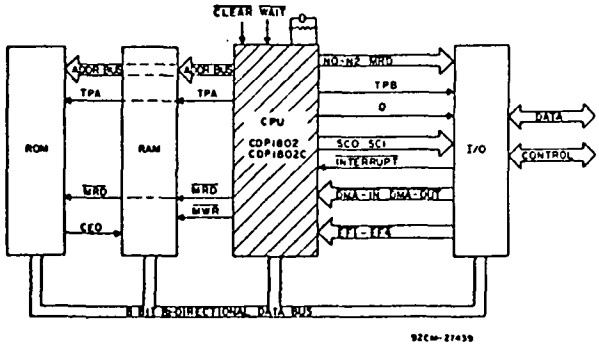


Fig. 1—Typical CDP1802 microprocessor system.

▲The Preliminary Data are intended for guidance purposes in evaluating the device for equipment design. The device is now being designed for inclusion in our standard line of commercially available products. For current information on the status of this program, please contact your RCA Sales Office.

Printed in USA/2-76

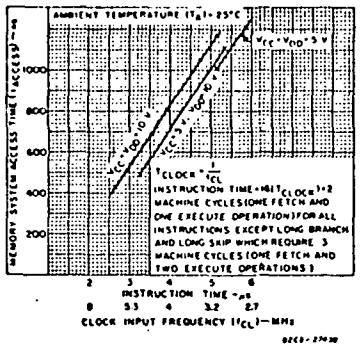


Fig. 3—Typical instruction time vs. memory system access time.

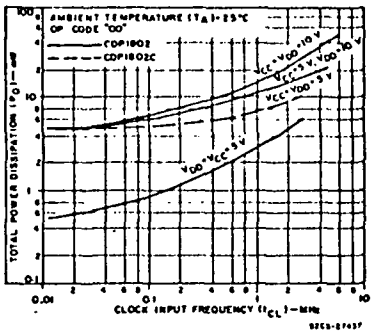


Fig. 4—Typical total power dissipation vs. clock input frequency.

ARCHITECTURE

The COSMAC block diagram is shown in Fig. 5. The principal feature of this system is a register array (R) consisting of sixteen 16-bit scratchpad registers. Individual registers in the array (R) are designated (selected) by a 4-bit binary code from one of the 4-bit registers labeled N, P, and X. The contents of any register can be directed to any one of the following three paths:

1. the external memory (multiplexed, higher-order byte first, on to 8 memory address lines);
2. the D register (either of the two bytes can be gated to D);
3. the increment/decrement circuit where it is increased or decreased by one and stored back in the selected 16-bit register.

The three paths, depending on the nature of the instruction, may operate independently or in various combinations in the same machine cycle.

With two exceptions, COSMAC instructions consist of two 8-clock-pulse machine cycles. The first cycle is the fetch cycle, and the second—and third, if necessary—are execute cycles. During the fetch cycle the four bits in the P designator select one of the 16 registers R(P) as the current program counter. The selected register R(P) contains the address of the memory location from which the instruction is to be fetched. When the instruction is read out from the memory, the higher-order 4 bits of the instruction byte are loaded into the I register and the lower-order 4 bits into the N register. The content of the program counter is automatically incremented by one so that R(P) is now "pointing" to the next byte in the memory.

The X designator selects one of the 16 registers R(X) to "point" to the memory for an operand (or data) in certain ALU or I/O operations.

The N designator can perform the following five functions depending on the type of instruction fetched:

1. designate one of the 16 registers in R to be acted upon during register operations;
2. indicate to the I/O devices a command code or device-selection code for peripherals;
3. indicate the specific operation to be executed during the ALU instructions, types of tests to be performed during the Branch instructions, or the specific operation required in a class of miscellaneous instructions (70-73 and 78-7B);
4. indicate the value to be loaded into P to designate a new register to be used as the program counter R(P);
5. indicate the value to be loaded into X to designate a new register to be used as data pointer R(X).

The registers in R can be assigned by a programmer in three different ways: as program counters, as data pointers, or as scratchpad locations (data registers) to hold two bytes of data.

Program Counters

Any register can be the main program counter; the address of the selected register is held in the P designator. Other registers in R can be used as subroutine program counters. By a single instruction the contents of the P register can be changed to effect a "call" to a subroutine. When interrupts are being serviced, register R(1) is used as the program counter for the interrupt servicing routine. At all other times the register designated as program counter is at the discretion of the user.

Data Pointers

The registers in R may be used as data pointers to indicate a location in memory. The register designated by X (i.e., R(X)) points to memory for the following instructions (see Table I):

1. ALU operations F0-F5, F7, 74, 75, 77;
2. output instructions 61 through 67;
3. input instructions 69 through 6F;
4. certain miscellaneous instructions—70-73, 78.

The register designated by N (i.e., R(N)) points to memory for the "load D from memory" instructions 0N and 4N and the "Store D" instruction 5N. The register designated by P (i.e., the program counter) is

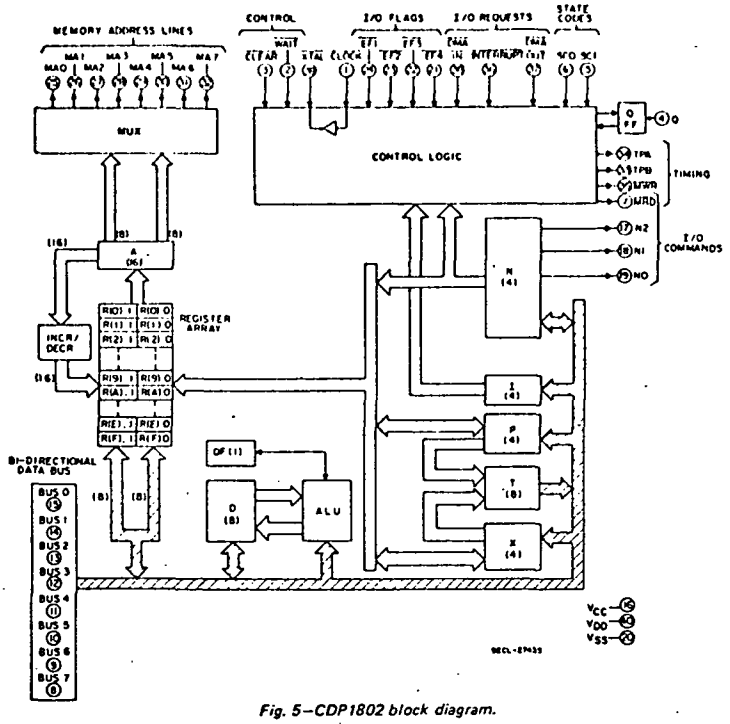


Fig. 5–CDP1802 block diagram.

used as the data pointer for ALU instructions F8-FD, FF, 7C, 7D, 7F. During these instruction executions the operation is referred to as "data immediate".

Another important use of R as a data pointer supports the built-in Direct-Memory-Access (DMA) function. When a DMA-In or DMA-Out request is received, one machine cycle is "stolen". This operation occurs at the end of the execute machine cycle in the current instruction. Register R(0) is always used as the data pointer during the DMA operation. The data is read from (DMA-Out) or written into (DMA-In) the memory location pointed to by the R(0) register. At the end of the transfer, R(0) is incremented by one so that the processor is ready to act upon the next DMA byte transfer request. This feature in the COSMAC architecture saves a substantial amount of logic when fast exchanges of blocks of data are required, such as with magnetic discs or during CRT-display-refresh cycles.

A program load facility, using the DMA-In channel, is provided to enable users to load programs into the memory. This facility provides a simple, one-step means for initially entering programs into the microprocessor system and eliminates the requirement for specialized "bootstrap" ROM's.

Data Registers

When registers in R are used to store bytes of data, four instructions are provided which allow D to receive from or write into either the higher-order- or lower-order-byte portions of the register designated by N. By this mechanism (together with loading by data immediate) program pointer and data pointer designations are initialized. Also, this technique allows scratchpad registers in R to be used to hold general data. By employing increment or decrement instructions, such registers may be used as loop counters.

The Q Flip Flop

An internal flip flop, Q, can be set or reset by instruction and can be sensed by conditional branch instructions. The output of Q is also available as a microprocessor output.

Interrupt Servicing

Register R(1) is always used as the program counter whenever interrupt servicing is initiated. When an interrupt request comes in and the interrupt is allowed by the program (again, nothing takes place until the completion of the current instruction) the contents of the X and P registers are stored in the temporary register T, and X and P are set to new values; hex digit 2 in X and hex digit 1 in P. Interrupt enable is automatically deactivated to inhibit further interruptions. The interrupt routine is now in control; the contents of T are saved by means of a single instruction (78) in the memory location pointed to by R(X). At the conclusion of the interrupt, the routine restores the pre-interrupted values of X and P with a single instruction (70 or 71). The interrupt-enable flip-flop can be activated to permit further interrupts or can be disabled to prevent them.

COSMAC Register Summary

D	8 Bits	Data Register (Accumulator)	N	4 Bits	Holds Low-Order Instr. Digit
DF	1 Bit	Data Flag (ALU Carry)	I	4 Bits	Holds High Order Instr. Digit
R	16 Bits	1 of 16 Scratchpad Registers	T	8 Bits	Holds old X, P after Interrupt (X is high byte)
P	4 Bits	Designates which register is Program Counter	IE	1 Bit	Interrupt Enable
X	4 Bits	Designates which register is Data Pointer	Q	1 Bit	Output Flip Flop

INSTRUCTION SET

The COSMAC instruction summary is given in Table I. Hexadecimal notation is used to refer to the 4-bit binary codes.

In all registers bits are numbered from the least significant bit (LSB) to the most significant bit (MSB) starting with 0.

R(W): Register designated by W, where W=N or X, or P

R(W).0: Lower-order byte of R(W)
R(W).1: Higher-order byte of R(W)
N0 = Least significant Bit of N Register

Operation Notation
M(R(N)) → D; R(N) + 1

This notation means: The memory byte pointed to by R(N) is loaded into D, and R(N) is incremented by 1.

TABLE I – INSTRUCTION SUMMARY
(For Notes, see page 9)

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
MEMORY REFERENCE			
LOAD VIA N	LDN	0N	M(R(N))→D; FOR N NOT 0
LOAD ADVANCE	LDA	4N	M(R(N))→D; R(N) +1
LOAD VIA X	LDX	F0	M(R(X))→D
LOAD VIA X AND ADVANCE	LDXA	72	M(R(X))→D; R(X) +1
LOAD IMMEDIATE	LDI	F8	M(R(P))→D; R(P) +1
STORE VIA N	STR	5N	D→M(R(N))
STORE VIA X AND DECREMENT	STXD	73	D→M(R(X)); R(X) −1
REGISTER OPERATIONS			
INCREMENT REG N	INC	1N	R(N) +1
DECREMENT REG N	DEC	2N	R(N) −1
INCREMENT REG X	IRX	60	R(X) +1
GET LOW REG N	GLO	8N	R(N).0→D
PUT LOW REG N	PLO	AN	D→R(N).0
GET HIGH REG N	GHI	9N	R(N).1→D
PUT HIGH REG N	PHI	BN	D→R(N).1
LOGIC OPERATIONS♦♦			
OR	OR	F1	M(R(X)) OR D→D
OR IMMEDIATE	ORI	F9	M(R(P)) OR D→D; R(P) +1
EXCLUSIVE OR	XOR	F3	M(R(X)) XOR D→D
EXCLUSIVE OR IMMEDIATE	XRI	FB	M(R(P)) XOR D→D; R(P) +1
AND	AND	F2	M(R(X)) AND D→D
AND IMMEDIATE	ANI	FA	M(R(P)) AND D→D; R(P) +1
SHIFT RIGHT	SHR	F6	SHIFT D RIGHT, LSB(D)→DF, 0→MSB(D)
SHIFT RIGHT WITH CARRY	SHRC	76♦	SHIFT D RIGHT, LSB(D)→DF, DF→MSB(D)
RING SHIFT RIGHT	RSHR		
SHIFT LEFT	SHL	FE	SHIFT D LEFT, MSB(D)→DF, 0→LSB(D)
SHIFT LEFT WITH CARRY	SHLC	7E♦	SHIFT D LEFT, MSB(D)→DF, DF→LSB(D)
RING SHIFT LEFT	RSHL		

♦NOTE: THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.

♦♦NOTE: THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE DF.
AFTER AN ADD INSTRUCTION
DF = 1 DENOTES A CARRY HAS OCCURRED
DF = 0 DENOTES A CARRY HAS NOT OCCURRED
AFTER A SUBTRACT INSTRUCTION
DF = 1 DENOTES NO BORROW. D IS A TRUE POSITIVE NUMBER
DF = 0 DENOTES A BORROW. D IS TWO'S COMPLEMENT
THE SYNTAX "−(NOT DF)" DENOTES THE SUBTRACTION OF THE BORROW

INSTRUCTION SUMMARY (CONT'D)

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
ARITHMETIC OPERATIONS♦♦			
ADD	ADD	F4	M(R(X)) +D→DF, D
ADD IMMEDIATE	ADI	FC	M(R(P)) +D→DF, D; R(P) +1
ADD WITH CARRY	ADC	74	M(R(X)) +D +DF→DF, D
ADD WITH CARRY, IMMEDIATE	ADCI	7C	M(R(P)) +D +DF→DF, D R(P) +1
SUBTRACT D	SD	F5	M(R(X))−D→DF, D
SUBTRACT D IMMEDIATE	SDI	FD	M(R(P))−D→DF, D; R(P) +1
SUBTRACT D WITH BORROW	SDB	75	M(R(X))−D−(NOT DF)→DF, D
SUBTRACT D WITH BORROW, IMMEDIATE	SDBI	7D	M(R(P))−D−(NOT DF)→DF, D; R(P) +1
SUBTRACT MEMORY	SM	F7	D−M(R(X))→DF, D
SUBTRACT MEMORY IMMEDIATE	SMI	FF	D−M(R(P))→DF, D; R(P) +1
SUBTRACT MEMORY WITH BORROW	SMB	77	D−M(R(X))−(NOT DF)→DF, D
SUBTRACT MEMORY WITH BORROW, IMMEDIATE	SMBI	7F	D−M(R(P))−(NOT DF)→DF, D R(P) +1
BRANCH INSTRUCTIONS–SHORT BRANCH			
SHORT BRANCH	BR	30	M(R(P))→R(P).0
NO SHORT BRANCH (SEE SKP)	NBR	38♦	R(P) +1
SHORT BRANCH IF D=0	BZ	32	IF D=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF D NOT 0	BNZ	3A	IF D NOT 0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF DF=1	BDF	33♦	IF DF=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF POS OR ZERO	BPZ		
SHORT BRANCH IF EQUAL OR GREATER	BGE		
SHORT BRANCH IF DF=0	BNF	3B♦	IF DF=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF MINUS	BM		
SHORT BRANCH IF LESS	BL		
SHORT BRANCH IF Q=1	BQ	31	IF Q=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF Q=0	BNQ	39	IF Q=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF1=1	B1	34	IF EF1=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF1=0	BN1	3C	IF EF1=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF2=1	B2	35	IF EF2=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF2=0	BN2	3D	IF EF2=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF3=1	B3	36	IF EF3=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF3=0	BN3	3E	IF EF3=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF4=1	B4	37	IF EF4=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF4=0	BN4	3F	IF EF4=0, M(R(P))→R(P).0 ELSE R(P) +1

♦NOTE: THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED

♦♦NOTE: THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE DF.
AFTER AN ADD INSTRUCTION:
DF = 1 DENOTES A CARRY HAS OCCURRED
DF = 0 DENOTES A CARRY HAS NOT OCCURRED
AFTER A SUBTRACT INSTRUCTION:
DF = 1 DENOTES NO BORROW, D IS A TRUE POSITIVE NUMBER
DF = 0 DENOTES A BORROW, D IS TWO'S COMPLEMENT
THE SYNTAX "−(NOT DF)" DENOTES THE SUBTRACTION OF THE BORROW

INSTRUCTION SUMMARY (CONT'D)

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
BRANCH INSTRUCTIONS- LONG BRANCH			
LONG BRANCH	LBR	C0	M(R(P))→R(P).1 M(R(P) +1)→R(P).0
NO LONG BRANCH (SEE LSKP)	NLBR	C8♦	R(P) +2
LONG BRANCH IF D=0	LBZ	C2	IF D=0, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
LONG BRANCH IF D NOT 0	LBNZ	CA	IF D NOT 0, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
LONG BRANCH IF DF=1	LBDF	C3	IF DF=1, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
LONG BRANCH IF DF=0	LBNF	CB	IF DF=0, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
LONG BRANCH IF Q=1	LBQ	C1	IF Q=1, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
LONG BRANCH IF Q=0	LBNQ	C9	IF Q=0, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
SKIP INSTRUCTIONS			
SHORT SKIP (SEE NBR)	SKP	38♦	R(P) +1
LONG SKIP (SEE NLBR)	LSKP	C8♦	R(P) +2
LONG SKIP IF D=0	LSZ	CE	IF D=0, R(P) +2 ELSE CONTINUE
LONG SKIP IF D NOT 0	LSNZ	C6	IF D NOT 0, R(P) +2 ELSE CONTINUE
LONG SKIP IF DF=1	LSDF	CF	IF DF=1, R(P) +2 ELSE CONTINUE
LONG SKIP IF DF=0	LSNF	C7	IF DF=0, R(P) +2 ELSE CONTINUE
LONG SKIP IF Q=1	LSQ	CD	IF Q=1, R(P) +2 ELSE CONTINUE
LONG SKIP IF Q=0	LSNQ	C5	IF Q=0, R(P) +2 ELSE CONTINUE
LONG SKIP IF IE=1	LSIE	CC	IF IE=1, R(P) +2 ELSE CONTINUE
CONTROL INSTRUCTIONS			
IDLE	IDL	00#	WAIT FOR DMA OR INTERRUPT; M(R(0))→BUS
NO OPERATION	NOP	C4	CONTINUE
SET P	SEP	DN	N→P
SET X	SEX	EN	N→X
SET Q	SEQ	7B	1→Q
RESET Q	REQ	7A	0→Q
SAVE	SAV	78	T→M(R(X))
PUSH X,P TO STACK	MARK	79	(X,P)→T; (X,P)→M(R(2)) THEN P→X; R(2)−1
RETURN	RET	70	M(R(X))→(X,P); R(X) +1 1→IE
DISABLE	DIS	71	M(R(X))→(X,P); R(X) +1 0→IE

#An idle instruction initiates a repeating S1 cycle. The processor will continue to idle until an I/O request ($\overline{\text{INTERRUPT}}$, $\overline{\text{DMA-IN}}$, or $\overline{\text{DMA-OUT}}$) is activated. When the request is acknowledged, the IDLE cycle is terminated and the I/O request is serviced, and then normal operation is resumed.

♦NOTE: THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.

INSTRUCTION SUMMARY (CONT'D)

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
INPUT–OUTPUT BYTE TRANSFER			
OUTPUT 1	OUT 1	61	M(R(X))→BUS; R(X) +1; N LINES = 1
OUTPUT 2	OUT 2	62	M(R(X))→BUS; R(X) +1; N LINES = 2
OUTPUT 3	OUT 3	63	M(R(X))→BUS; R(X) +1; N LINES = 3
OUTPUT 4	OUT 4	64	M(R(X))→BUS; R(X) +1; N LINES = 4
OUTPUT 5	OUT 5	65	M(R(X))→BUS; R(X) +1; N LINES = 5
OUTPUT 6	OUT 6	66	M(R(X))→BUS; R(X) +1; N LINES = 6
OUTPUT 7	OUT 7	67	M(R(X))→BUS; R(X) +1; N LINES = 7
INPUT 1	INP 1	69	BUS→M(R(X)); BUS→D; N LINES = 1
INPUT 2	INP 2	6A	BUS→M(R(X)); BUS→D; N LINES = 2
INPUT 3	INP 3	6B	BUS→M(R(X)); BUS→D; N LINES = 3
INPUT 4	INP 4	6C	BUS→M(R(X)); BUS→D; N LINES = 4
INPUT 5	INP 5	6D	BUS→M(R(X)); BUS→D; N LINES = 5
INPUT 6	INP 6	6E	BUS→M(R(X)); BUS→D; N LINES = 6
INPUT 7	INP 7	6F	BUS→M(R(X)); BUS→D; N LINES = 7

1 Long-Branch, Long-Skip and No Op instructions are the only instructions that require three cycles to complete (1 fetch + 2 execute).

Long-Branch instructions are three bytes long. The first byte specifies the condition to be tested; and the second and third byte, the branching address.

The long-branch instructions can:

a) Branch unconditionally
b) Test for D=0 or D≠0
c) Test for DF=0 or DF=1
d) Test for Q=0 or Q=1
e) effect an unconditional no branch

If the tested condition is met, then branching takes place; the branching address bytes are loaded in the high-and-low-order bytes of the current program counter, respectively. This operation effects a branch to any memory location.

If the tested condition is not met, the branching address bytes are skipped over, and the next instruction in sequence is fetched and executed. This operation is taken for the case of unconditional no branch.

2. The short-branch instructions are two bytes long. The first byte specifies the condition to be tested, and the second specifies the branching address.

The short-branch instructions can:

a) Branch unconditionally
b) Test for D=0 or D≠0
c) Test for DF=0 or DF=1
d) Test for Q=0 or Q=1
e) Test the status (1 or 0) of the four EF flags
f) Effect an unconditional no branch

If the tested condition is met, then branching takes place; the branching address byte is loaded into the low-order byte position of the current program counter. This effects a branch with the current 256-byte page of the memory, i.e., the page which holds the branching address. If the tested condition is not met, the branching address byte is skipped over, and the next instruction in sequence is fetched and executed. This same action is taken in the case of unconditional no branch.

3. The skip instructions are one byte long. There is one Unconditional Short-Skip (SKP) and eight Long-Skip instructions.

The Unconditional Short-Skip instruction takes 2 cycles to complete (1 fetch + 1 execute). Its action is to skip over the byte following it. Then the next instruction in sequence is fetched and executed. This SKP instruction is identical to the unconditional no-branch instruction (NBR) except that the skipped-over byte is not considered part of the program.

The Long-Skip instructions take three cycles to complete (1 fetch + 2 execute).

They can:

a) Skip unconditionally
b) Test for D=0 or D≠0
c) Test for DF=0 or DF=1
d) Test for Q=0 or Q=1
e) Test for IE=1

If the tested condition is met, then Long Skip takes place; the current program counter is incremented twice. Thus two bytes are skipped over and the next instruction in sequence is fetched and executed. If the tested condition is not met, then no action is taken. Execution is continued by fetching the next instruction in sequence.

SIGNAL DESCRIPTIONS

BUS 0 to BUS 7 (Data Bus)

8-bit directional DATA BUS lines. These lines are used for transferring data between the memory, the microprocessor, and I/O devices.

N0 to N2 (I/O Command)

Issued by an I/O instruction to signal the I/O control logic of a data transfer between memory and I/O interface. These lines can be used to issue command codes or device selection codes to the I/O devices (independently or combined with the memory byte on the data bus when an I/O instruction is being executed). The N bits are low at all times except when an I/O instruction is being executed. During this time their state is the same as the corresponding bits in the N register.

The direction of data flow is defined in the I/O instruction by bit N3 and is indicated by the level of the MRD signal.

$\overline{MRD} = V_{CC}$: Data from I/O to CPU and Memory

$\overline{MRD} = V_{SS}$: Data from Memory to I/O

$\overline{EF1}$ to $\overline{EF4}$ (4 Flags)

These levels enable the I/O controllers to transfer status information to the processor. The levels can be tested by the conditional branch instructions. They can be used in conjunction with the INTERRUPT request line to establish interrupt priorities. These flags can also be used by I/O devices to "call the attention" of the processor, in which case the program must routinely test the status of these flag(s). The flag(s) are sampled at the beginning of every S1 cycle.

$\overline{INTERRUPT}$, $\overline{DMA\text{-}IN}$, $\overline{DMA\text{-}OUT}$ (3 I/O Requests)

These signals are sampled by the CDP1802 during the interval between the leading edge of TPB and the leading edge of TPA.

Interrupt Action: X and P are stored in T after executing current instruction; designator X is set to 2; designator P is set to 1; interrupt enable is reset to 0 (inhibit); and instruction execution is resumed.

DMA Action: Finish executing current instruction; R(0) points to memory area for data transfer; data is loaded into or read out of memory; and increment R(0).

Note: In the event of concurrent DMA and INTERRUPT requests, DMA-IN has priority followed by DMA-OUT and then INTERRUPT.

SC0, SC1, (2 State Code Lines)

These lines indicate that the CPU is: 1) fetching an instruction, or 2) executing an instruction, or 3) processing a DMA request, or 4) acknowledging an interrupt request. The levels of state code are tabulated below. All states are valid at TPA. H = V_{CC}, L = V_{SS}.

State Type	State Code Lines	
	SC1	SC0
S0 (Fetch)	L	L
S1 (Execute)	L	H
S2 (DMA)	H	L
S3 (Interrupt)	H	H

TPA, TPB (2 Timing Pulses)

Positive pulses that occur once in each machine cycle (TPB follows TPA). They are used by I/O controllers to interpret codes and to time interaction with the data bus. The trailing edge of TPA is used by the memory system to latch the higher-order byte of the 16-bit memory address. TPA is suppressed in IDLE when the CPU is in the load mode.

MA0 to MA7 (8 Memory Address Lines)

The higher-order byte of a 16-bit COSMAC memory address appears on the memory address lines MA0-7 first. Those bits required by the memory system are strobed into external address latches by timing pulse TPA. The low-order byte of the 16-bit address appears on the address lines after the termination of TPA. Latching of all 8 higher-order address bits would permit a memory system of 64K bytes.

$\overline{MWR}$ (Write Pulse)

A negative pulse appearing in a memory-write cycle, after the address lines have stabilized.

$\overline{MRD}$ (Read Level)

A low level on $\overline{MRD}$ indicates a memory read cycle. It can be used to control three-state outputs from the addressed memory which may have a common data input and output bus. If a memory does not have a three-state high-impedance output, $\overline{MRD}$ is useful for driving memory/bus separator gates. It is also used to indicate the direction of data transfer during an I/O instruction:

$\overline{MRD} = V_{CC}$: Data from I/O to CPU and Memory

$\overline{MRD} = V_{SS}$: Data from Memory to I/O

Q

Single bit output from the CPU which can be set or reset under program control. During SEQ or REQ instruction execution, Q is set or reset between the trailing edge of TPA and the leading edge of TPB.

CLOCK

Input for externally generated single-phase clock. A typical clock frequency is 6.4 MHz at $V_{CC} = V_{DD} = 10$ volts. The clock is counted down internally to 8 clock pulses per machine cycle.

$\overline{XTAL}$

Connection to be used with clock input terminal, for an external crystal, if the on-chip oscillator is utilized. The crystal is connected between terminals 1 and 39 (CLOCK and $\overline{XTAL}$) in parallel with a resistance (10 megohms typ.). Frequency trimming capacitors may be required at terminals 1 and 39.

$\overline{WAIT}$, $\overline{CLEAR}$ (2 Control Lines)

Provide four control modes as listed in the following truth table:

$\overline{CLEAR}$	$\overline{WAIT}$	MODE
L	L	Load
L	H	Reset
H	L	Pause
H	H	Run

The function of the modes are defined as follows:

Load

Holds the CPU in the IDLE execution state and allows an I/O device to load the memory without the need for a "bootstrap" loader. It modifies the IDLE condition so that DMA-IN operation does not force execution of the next instruction.

Reset

Registers I, N, Q are reset, IE is set and 0's (V_{SS}) are placed on the data bus. TPA and TPB are suppressed while reset is held and the CPU is placed in S1. The first machine cycle after termination of reset is an initialization cycle. During this cycle the CPU remains in S1 and registers X, P, and R(0) are reset. Interrupt and DMA servicing are suppressed during the initialization cycle.

The next cycle is an S0, S1, or an S2 but never an S3. With the use of a 71 instruction followed by 00 at memory locations 0000 and 0001, this feature may be used to reset IE, so as to preclude interrupts until ready for them. Power-up reset can be realized by connecting an external RC to $\overline{CLEAR}$.

Pause

Stops the internal CPU timing generator on the first negative high-to-low transition of the input clock. The oscillator continues to operate, but subsequent clock transitions are ignored.

Run

May be initiated from the Pause or Reset mode functions. If initiated from Pause, the CPU resumes operation on the first negative high-to-low transition of the input clock. When initiated from the Reset operation, the first machine cycle following Reset is always the initialization cycle. The initialization cycle is then followed by a DMA (S2) cycle or fetch (S0) from location 0000 in memory.

V_{DD}, V_{SS}, V_{CC} (Power Levels)

The internal voltage supply V_{DD} is isolated from the Input/Output voltage supply V_{CC} so that the processor may operate at maximum speed while interfacing with various external circuit technologies, including T^2L at 5 volts. V_{CC} must be less than or equal to V_{DD}. All outputs swing from V_{SS} to V_{CC}. The recommended input voltage swing is V_{SS} to V_{CC}.

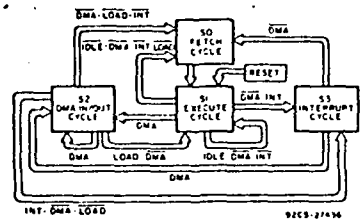


Fig. 6–CDP1802 microprocessor state transitions (Run Mode).

The CDP1802 and CDP1802C CPU state transitions when in the RUN mode are shown in Fig. 6. Each machine cycle requires the same period of time–8 clock pulses. The execution of an instruction requires either two or three machine cycles, S0 followed by a single S1 cycle or two S1 cycles. S2 is the response to a DMA request and S3 is the interrupt response.

OPERATING AND HANDLING CONSIDERATIONS FOR CDP1802D AND CDP1802CD

1. **Handling**

All inputs and outputs of this device have a network for electrostatic protection during handling. Recommended handling practices for COS/MOS devices are described in ICAN-6000 "Handling and Operating Considerations for MOS Integrated Circuits", available on request from RCA Solid State Division, Box 3200, Somerville, N.J. 08876.

2. **Operating**

Operating Voltage

During operation near the maximum supply voltage limit, care should be taken to avoid or suppress power supply turn-on and turn-off transients, power supply ripple, or ground noise; any of these conditions must not cause V_{DD}-V_{SS} to exceed the absolute maximum rating. V_{CC} must be less than equal to V_{DD}. Power supplies should be sequenced to insure compliance.

Input Signals

To prevent damage to the input protection circuit, input signals should never be greater than V_{DD} nor less than V_{SS}. Input currents must not exceed 10 mA even when the power supply is off.

Unused Inputs

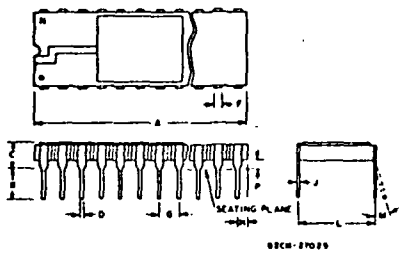
A connection must be provided at every input terminal. All unused input terminals must be connected to either V_{DD} or V_{SS}, whichever is appropriate.

Output Short Circuits

Shorting of outputs to V_{DD} or V_{SS} may damage COS/MOS devices by exceeding the maximum device dissipation.

DIMENSIONAL OUTLINE

CDP1802D, CDP1802CD
40-Lead Dual-In-Line Ceramic



DIM.	MILLIMETERS		INCHES	
	MIN.	MAX.	MIN.	MAX.
A	50.30	51.30	1.980	2.020
C	2.42	3.93	0.095	0.155
D	0.43	0.56	0.017	0.023
F	1.27 REF.		0.050 REF.	
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.30	0.008	0.012
K	3.18	4.45	0.125	0.175
L	14.74	15.74	0.580	0.620
M	–	7°	–	7°
P	0.64	1.27	0.025	0.050
N	40		40	

NOTES

1. Leads within 0.13 mm (0.005) radius of true position at maximum material condition.
2. Dimension "L" to center of leads when formed parallel.
3. When this device is supplied solder dipped, the maximum lead thickness (narrow portion) will not exceed 0.013 in. (0.33 mm)

When incorporating RCA Solid State Devices in equipment, it is recommended that the designer refer to "Operating Considerations for RCA Solid State Devices", Form No. 1CE 402, available on request from RCA Solid State Division, Box 3200, Somerville, N. J. 08876.