64000 LOGIC DEVELOPMENT SYSTEM

ASSEMBLER SUPPLEMENT 8048 SERIES

HEWLETT PACKARD

CERTIFICATION

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

This Hewlett-Packard system product is warranted against defects in materials and workmanship for a period of 90 days from date of installation. During the warranty period, HP will, at its options, either repair or replace products which prove to be defective.

Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other cases, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service. Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HP SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HP SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

ASSISTANCE

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.

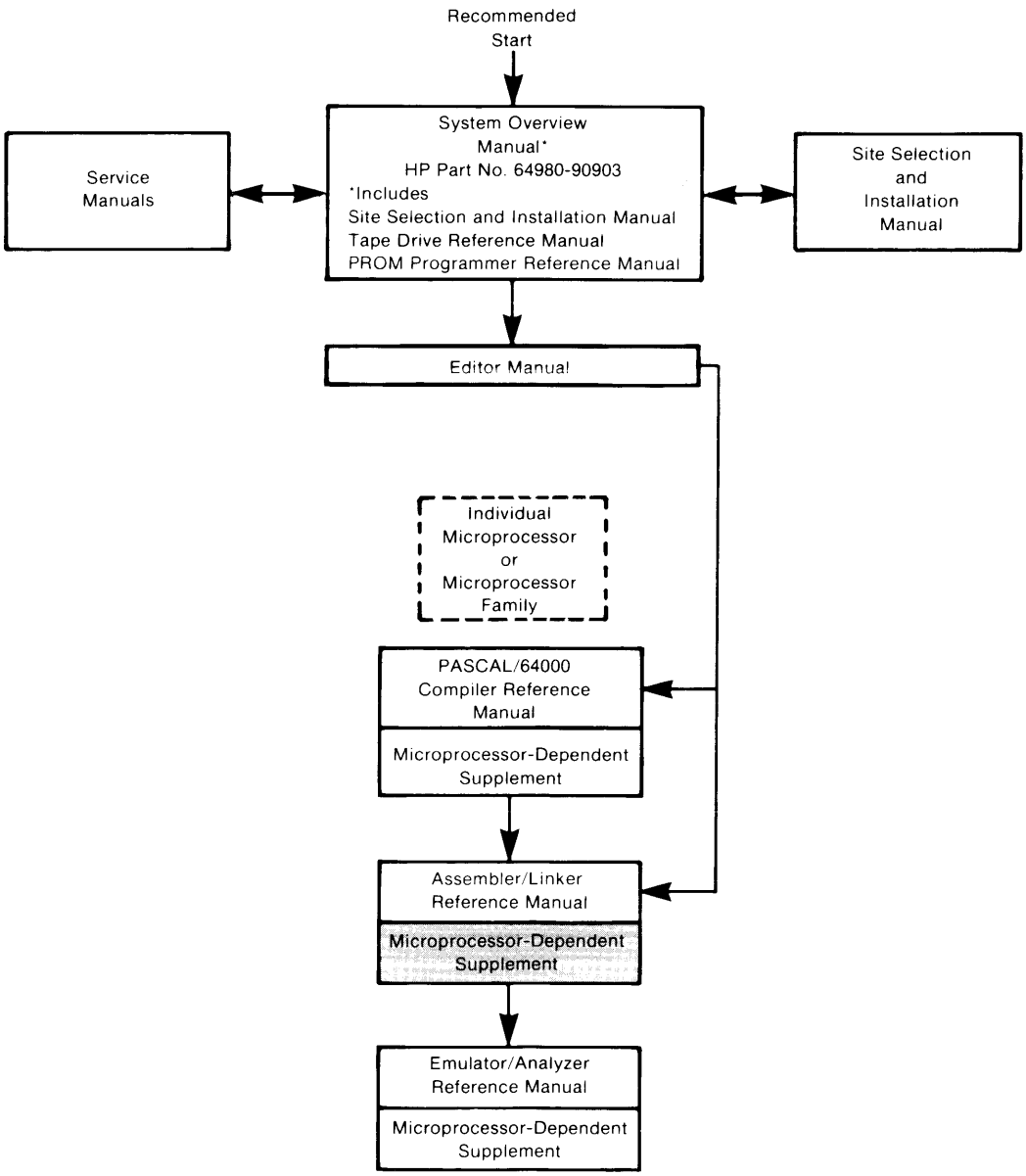
CW&A 9/79

Model 64000 Reference Manuals

The following block diagram shows the documentation scheme for the HP Model 64000 Logic Development System. The interconnecting arrows show the recommended progression through the manuals as a way of gaining familiarity with the system.

For a detailed map showing specific manuals and their part numbers, refer to the Manual Map in the System Overview Manual.

Manual Map



Printing History

Each new edition of this manual incorporates all material updated since the previous edition. Each new or revised page is indicated by a revision (rev) date. Manual change sheets are issued between editions, allowing you to correct or insert information in the current edition.

The part number on the back cover changes only when each new edition is published. Minor corrections or additions may be made as the manual is reprinted between editions.

First Printing ..... April 1980 (Part Number 64846-90902)
Reprinted ..... November 1980

Assembler Supplement
8048 Series

HP Model 64000 Logic Development System

The information in this supplement has been checked for accuracy and is believed to be correct; however, no responsibility is assumed for inaccuracies. When discrepancies are noted, refer to the manufacturer's Microprocessor Program Manual for clarification.

Table of Contents

List of Tables

Chapter 1

General Information (8048 Series)

Introduction

This chapter contains general information about the 8048 series of microprocessors/microcomputers and briefly discusses their architecture, addressing modes, and condition codes. For a detailed description of a particular device, refer to the manufacturer's User's Manual.

NOTE

The term "microprocessor" will be used in this manual for both microcomputers and microprocessors.

NOTE

If you are unfamiliar with assembly language or assemblers, read Chapter 6 in the Assembler/Linker manual for a brief review of assemblers, assembly language, and numbering systems.

NOTE

The following discussion on "Microprocessor Architecture" applies primarily to the 8048/8021 microprocessors. Differences between the 8048, 8041, and 8021 will be noted. Otherwise, the descriptions apply to all 8048-series devices. The 8041 differs somewhat from the 8048 in hardware and software design due to its external bus configuration. Therefore, a functional description of the 8041 microprocessor and how it differs from the 8048 is presented at the end of this chapter. (Any reference to the 8041 also applies to the 8741.)

Use the following assembler directives to specify the 8048-series devices.

"8048" = 8048, 8049, 8748, 8749.
"8021" = 8021, 8022, 8035, 8039.
"8041" = 8041, 8042, 8741.

Microprocessor Architecture

Program Memory

Resident program memory in the 8048 microprocessor consists of a 1K by 8-bit ROM (2K by 8-bit in 8049) which is divided into 256-byte pages. Program memory can be expanded up to 4K using additional ROMs (not applicable to the 8021). This additional memory can be addressed directly since the 8048 contains a 12-bit program counter. Bits Ø-1Ø of the program counter address up to 2K memory locations. Bit 11, when set by mnemonic instruction SEL MB1, permits addressing to 4K memory locations.

NOTE

The 8035 and 8039 do not have internal program memories. Otherwise, they are identical to the 8048 and 8049 respectively.

There are three reserved locations in program memory:

Location Ø- Initializing the Reset function of the microprocessor causes the first instruction to be brought from location Ø. Therefore, the first instruction to be executed after initialization should be stored in location Ø.

Location 3- Initializing the Interrupt function of the microprocessor (if enabled) causes a jump to location 3. Therefore, the first word of an external interrupt service subroutine should be stored in location 3.

Location 7 - An interrupt resulting from a timer/counter overflow (if enabled) causes a jump to location 7. Therefore, the first word of a timer/counter service routine should be stored in location 7.

Data Memory

The 8048, in addition to resident program memory, contains a 64 by 8-bit RAM data memory (128 by 8-bit in 8049). The memory is made up of eight working registers plus an additional eight registers selectable by register bank switch instruction SEL RB (not applicable to the 8021), an eight-level program counter stack, and a scratch-pad area. The amount of scratch-pad memory available will depend on the number of addresses in the stack and the number of working registers selected.

The working registers are assigned to data memory locations Ø-7. The additional working registers, when selected, are assigned to data memory locations 24-31 (not applicable to the 8021). Working registers in RAM memory can be addressed directly by specifying a register number. Other locations in RAM memory are addressed indirectly by using data memory registers RØ and R1 to specify the location desired. The symbol "@" (commercial at) indicates the indirect addressing mode of operation.

Since all 64 locations can be addressed by 6 bits (Ø-5), the most significant bits (6 and 7) of the addressing registers (RØ and R1) are ignored on the 8048 (not applicable to the 8049). However, all 8 bits of register RØ and register R1 can be used in combination with the MOVX instructions (not applicable to the 8021) to indirectly address up to 256 locations when external data memory is used.

Program Counter and Stack

The program counter is an independent counter and is not directly accessible. It is initialized to zero by activating the Reset function.

The program counter stack is implemented using pairs of registers in the data memory area. Locations 8-23 are used to provide an 8-level stack. When program execution branches to a subroutine or an interrupt service routine, the 12-bit program counter and bits 4-7 of the program status word (PSW) are stored in two stack locations.

The program counter stack is addressed by stack pointer (STP) bits Ø-2 in the program status word. The setting of the STP bits indicates the location to be loaded the next time the program counter is stored. The STP is incremented each time the program counter is stored and decremented each time the program counter is restored. Unused locations in the stack may be used as a scratch pad area.

Program Status Word

The program status word (PSW) contains 8 bits of status information used by the microprocessor. The PSW bit definitions are as follows:

Bits Ø-2 - Stack Pointer address location

Bit 3 - Not used

Bit 4 - Working register bank switch:
bank Ø (RØ-R7) = bit 4 reset (Ø)
bank 1 (R24-31) = bit 4 set (1)

Bit 5 - Flag Ø (user controlled)

Bit 6 - Auxiliary Carry (AC) bit

Bit 7 - Carry (C) bit

The carry flag (C) bit is affected by addition, decimal adjust instructions, and certain rotation operations and generally indicates a carry out of the bit 7 position of register A (accumulator). It can be complemented, reset to zero, and tested by a conditional jump instruction.

The auxiliary carry (AC) bit indicates a carry out of bit 3 in register A and is only applicable when decimal arithmetic is being performed. It cannot be tested or altered directly.

Functional Description

General

The main difference between the 8048 and the 8041 is that the 8041 includes handshaking capabilities and interface protocols that enable it to function as a programmable peripheral in a large microcomputing system.

Memory Differences

In a large microcomputer system, handshaking protocol requires the 8041 to use the BUS port for interfacing with the master computer. Therefore, the 8041 program memory cannot be expanded beyond 1K and data memory cannot be expanded beyond 64 locations.

Interrupt Differences

The external interrupt function of the 8041 is also committed to a master computer interface and the EN I and DIS I instructions have a different function.

When the master computer is transferring data to the 8041, it can generate an interrupt each time the 8041's data bus buffer is filled to ensure that two writes are not given before the buffer is cleared. The EN I and DIS I instructions enable and disable this interrupt. When initiated, the interrupt passes control to program memory location 3 as in the 8048.

The master computer must check special status bits to determine if the data bus buffer is empty when data is being transferred from the 8041 to it. No interrupt is possible except by dedicating I/O lines.

Pin 1 (TØ) cannot be used as a clock output in the 8041. It can only be used as a test input.

Hardware Differences

The 8041 has two special registers that are not available in the 8048. These registers are the data bus buffer (DBB) and the status register. A description of each follows:

a. **Data Bus Buffer:** the 8-bit data bus buffer provides a temporary storage facility for data passing between the master computer and the 8041. This transfer of data can be implemented with or without program interference by using the EN I and DIS I instructions.

b. **Status Register:** this is a 4-bit register that indicates the status of the flags FØ, F1, and two special 8041 flags. The two special flags, input buffer flag (IBF) and output buffer flag (OBF), indicate the condition of the data bus buffer and are initially cleared. The two special flags are used when transferring data to or from the master computer.

Chapter 2

Operand Rules and Conventions

General Information

There are three types of data that may be needed as items in the operand field:

a. **Register Information** - operands can reference directly data contained in the processor registers such as the stack, register A, or data memory registers RØ, R1, R2, R3, R4, R5, R6, and R7. In addition, operands can reference data contained on input/output ports.

Example:

```
MOV     A,RØ        ;MOVE CONTENTS OF
                    ;REGISTER Ø TO
                    ;REGISTER A
```

b. **Immediate Data** - operands can contain immediate data. The required value is inserted directly into the operand field. The value can be in the form of numbers, an expression to be evaluated at assembly time, a symbol, or an ASCII constant enclosed in quotation marks. The immediate data indicator is the pound (#) symbol.

Examples:

```
MOV     R1,#ØFFH    ;MOVE "FF" HEX TO
                    ;REGISTER R1

MOV     R4#"A"      ;MOVE VALUE OF ASCII
                    ;CONSTANT A (Ø1ØØØØØ1)
                    ;INTO REGISTER R4
```

c. **Memory Addressing** - working registers can be addressed directly in data memory by specifying a register number. Other locations in data memory can be addressed indirectly by using either register RØ or register R1 to specify the addressed location. Because all 64 locations in data memory (including registers RØ through R7) can be addressed by six bits; bits seven and eight are ignored. However, all eight bits are used by registers RØ and R1 during certain microprocessor instructions which require up to 256 locations in external data memory.

NOTE

The "commercial at" (@) symbol indicates that indirect addressing is desired.

Examples:

```
MOV     A,@R1

MOV     R4,#ØFH
```

Identifying Types Of Information

There are nine ways to define the types of information that can be presented in the operand field. These ways are discussed in the following paragraphs.

a. **Binary Data.** Each binary number must be followed by the letter B.

Example:

```
SAM     MOV     R3,#1ØØ1ØØ11B
```

b. **Octal Data.** Each octal number must be followed by either the letter O or the letter Q.

Example:

```
SAM     MOV     R3,#55O

        or

SAM     MOV     R3,#55Q
```

c. **Hexadecimal Data.** Each hexadecimal number must begin with a number and be followed by the letter H.

Example:

```
SAM     MOV     R3,#ØF1H
```

d. **Decimal Data.** Each decimal number may be followed by the letter D or it may stand alone. Any number not specifically identified as binary, octal, or hexadecimal is assumed to be decimal.

Example:

```
SAM     MOV     R3,#55D
                or
SAM     MOV     R3,#55
```

e. **ASCII Constants.** One or more ASCII characters enclosed in quotation marks identify an ASCII constant.

Example:

```
        MOV     R3,#'T'         ;LOADS REG R3 WITH
                                ;8-BIT ASCII CODE
                                ;FOR LETTER T

SAM     DB      "FULTON'S FOLLY"
```

f. **Location Counter.** The dollar symbol ($) refers to the current location counter. The location counter contains the address where the current instruction or data statement is being assembled.

Example:

```
JUMP    JMP     $+3             ;JUMP TO ADDRESS
                                ;3 BYTES BEYOND
                                ;FIRST BYTE OF THIS
                                ;INSTRUCTION
```

g. **Label Assigned Values.** The EQU directives can be used to assign values to labels.

Example:

```
SAM     EQU     6AH
```

h. **Labels of Instruction or Data.** The label assigned to an instruction or a data definition has as its value the address of the first byte of the instruction or data. Instructions elsewhere in the program can refer to this address by its symbolic name.

Example:

```
SAM     JMP     FRED    ;JUMP TO INSTRUC-
                        ;TION AT FRED
        .
        .
        .
        .
FRED    MOV     RØ,#6AH
```

i. **Expressions.** The operand field may contain an expression consisting of one or more terms acted on by the expression operators listed in Chapter 2 of the Assembler/Linker Manual. A term may be either a symbol, a string constant, a numeric constant, or an expression. The assembler reduces the entire expression to a single value.

Terms within expressions can be connected by the following arithmetic operators:

a. The plus operator (+) produces the arithmetic sum of its operands.

b. The minus operator (−) produces the arithmetic difference of its operands or, when used alone, the arithmetic negative of its operand.

c. The asterisk operator (*) produces the arithmetic product of the operands.

d. The slant operator (/) produces the quotient of its operands and discards any remainder.

e. An instruction enclosed in parentheses is a legal expression in the operand field. For expressions in parentheses, the deepest expression in the parentheses is evaluated first.

Be careful when using the arithmetic operators because their operational results may affect the condition flags of the processor registers.

Byte Selection Operator

The assembler's relocation characteristic treats all external and relocatable symbols as 16-bit addresses. When one of these symbols appears in the operand expression of an immediate instruction, it must be followed by the high operator (H) if the high-order byte is required. The default condition is the low-byte value. For example, if the symbol SAM has a value of 1FDBH, then the operand expression SAM would default to the low-order byte ØDBH. The operand expression SAM,H will select the high-order byte 1FH.

NOTE

Since the low-order byte is assumed in all cases not specified as high-order bytes, no error messages will be generated if an address or number is too large.

Invalid Operand Instructions

There are certain operand field instructions that are not recognized by the Model 64000 macroassembler and should be avoided. A list of the invalid instructions are as follows:

a. Shift Operators:

SHL	-	shift operand left
SHR	-	shift operand right

b. Logical Operators:

NOT	-	logical 1's complement
AND	-	logical AND
OR	-	logical OR
XOR	-	logical EXCLUSIVE OR

c. Compare Operators:

EQ	-	equal
NE	-	not equal
LT	-	less than
LE	-	less than or equal
GT	-	greater than
GE	-	greater than or equal
NUL	-	test for null macro parameters

Chapter 3
Special Pseudo Instructions

Introduction

This chapter provides supplemental information to Chapter 3 in the HP Model 64000 Assembler/Linker Manual. Assembler instructions that are applicable only to the 8048 series of microprocessors are defined herein.

Define Byte

SYNTAX:

Label	Operation	Operand	Object Code (hex)
[Name]	DB	expression list	

Applicable to: 8048, 8041, and 8021 microprocessors.

The DB pseudo instruction will store data in consecutive memory locations starting with the current setting of the program counter. The operand field may contain expressions or text strings or both.

Expressions will evaluate to one-byte numbers (8 bits) in the range Ø through 255 (ØØH through ØFFH).

NOTE

If the first hex number in an expression is an alpha character, it must be preceded by a zero.

The label name is optional. If a label name is present, it will be assigned the starting value of the program counter, and will reference the first byte stored by the DB directive. Therefore, the label SAM, in the following example, refers to the letter P in the string "PRICE".

Example:

Label	Operation	Operand	Comment
SAM	DB	"PRICE"	

Define Storage Block

SYNTAX:

Label	Operation	Operand	Object Code (hex)
[Name]	DS	expression	

Applicable to: 8048, 8041, and 8021 microprocessors.

The DS pseudo instruction can be used to define a block of memory space. The value of the expression in the operand field specifies the number of bytes to be reserved in memory.

Any symbol appearing in the operand field must be predefined. If the value of the operand expression is zero, no memory will be reserved; however, if the optional label name is present, it will be assigned the current value of the program counter.

The DS directive reserves memory by incrementing the program counter by the value in the operand expression.

Example:

Label	Operation	Operand	Comment
SAM	DS	15	;RESERVE 15 ;BYTES FOR SAM ;ROUTINE

Define Word

SYNTAX:

Label	Operation	Operand	Object Code (hex)
[Name]	DW	expression list	

Applicable to: 8048, 8041, and 8021 microprocessors.

The DW pseudo instruction will store each 16-bit value from the expression list as an address. The values will be stored in memory starting at the current setting of the program counter.

The most significant eight bits of the first value in the expression list will be stored at the current setting of the program counter; the least significant eight bits will be stored at the next higher location. This process will be repeated for each item in the expression list.

Expressions evaluate to one-word numbers (16 bits), typically addresses. If an expression evaluates to a single byte, it is assumed to be the low-order byte of a 16-bit word where the high-order byte is all zeros. Strings are limited to one or two characters.

If the label name is present, it will be assigned the starting address of the program counter, and thus reference the first byte stored by the DW directive.

Example:

Label	Operation	Operand	Comment
SAM	DW	ØB123H	

Chapter 4
Assembler Output Listing

General

The assembler processes the source program modules and produces an output that consists of a source program listing, a relocatable object file, and a symbol cross-reference list. Errors detected by the assembler will be noted in the output listing as error messages. Refer to Appendix D in the Assembler/Linker Manual for a listing of all error codes and their definitions.

Input/Output Files

Source Input File

The input to the assembler is a source file that is created through the editor. It consists of the following:

Example	Description
"8048"	- Assembler directive.
Source Code	- Source statements consisting of:
.	
.	Assembler Psuedos - refer to Chapter 3 (Assembler/ Linker Manual)
.	
.	Assembler Instructions - refer to Chapter 5, this Supplement
.	

Assembler Output Files

The assembler produces relocatable object modules that are stored under the same name as the source file but in a format that can be processed by the linker. If an object file does not exist at assembly time, the assembler creates one. If an object file does exist, the assembler replaces it.

List File. The list file is a formatted file designed for output to a line printer. It can also be stored in a file or applied to the display. The list can include:

a. Source statements with object code.

b. Error messages.

c. Summary of errors with a description list.

d. Symbol cross-reference list.

Symbol Cross-Reference List. All symbols are cross-referenced except local macro labels and parameters. A cross-reference listing contains:

a. Alphabetical list of program symbols.

b. Line numbers where symbols are defined.

c. All references (by line numbers) to symbols in the program.

Output Listing

An example of an assembler output listing is given in table 4-2 using the source program example listed in table 4-1. To illustrate an assembler output listing that contains error messages refer to table 4-3.

NOTE

The source program example was not written as a specific program. It merely lists a group of mnemonics to present a formatted example.

Table 4-1. Source Program Format Example

```
"8048" LIST XREF

          EXT     DSPL4,KYBD4
          ORG     ØBØØH
          STRT    CNT
          SPHL
          SEL     MBØ
          SEL     RBØ
          EN      I
          EN      TCNTI
EXEC      MOV     A,#ØFFH
          MØV     RØ,#ØØH
          MOV     R1,#1ØH
          MOV     T,A
          STRT    T
LOOP_A    IN      A,P1
          JZ      EXIT_A
          MOV     @RØ,A
          INC     RØ
          DJNZ    R1,LOOP_A
EXIT_A    MOV     A,R2
          MOV     PSW,A
          JFØ     DSPL4
          JF1     KYBD4
          CLR     FØ
          JTF     EXEC
          DEC     RØ
          MOV     A,@RØ
          OUTL    BUS,A
          DJNZ    RØ,EXIT_A
          END     ØBØØH
```

Table 4-2. Assembler Output Listing

FILE: EXEC4:SAVE **HEWLETT-PACKARD: INTEL 8048 ASSEMBLER**

LINE	LOC	CODE	ADDR	SOURCE STATEMENT		
				"8048" LIST XREF		
1						
2					EXT	DSPL4,KYBD4
3	ØBØØ				ORG	ØBØØH
4	ØBØØ	45			STRT	CNT
5	ØBØ1	E5			SEL	MBØ
6	ØBØ2	C5			SEL	RBØ
7	ØBØ3	Ø5			EN	I
8	ØBØ4	25			EN	TCNTI
9	ØBØ5	23	FF	EXEC	MOV	A,#ØFFH
1Ø	ØBØ7	B9	ØØ		MOV	R1,#ØØH
11	ØBØ9	62			MOV	T,A
12	ØBØA	55			STRT	T
13	ØBØB			LOOP_A	IN	A,P1
14	ØBØC	C6	12		JZ	EXIT_A
15	ØBØE	AØ			MOV	@RØ,A
16	ØBØF	18			INC	RØ
17	ØB1Ø	E9	ØB		DJNZ	DR1,LOOP_A
18	ØB12	FA		EXIT_A	MOV	A,R2
19	ØB13	D7			MOV	PSW,A
2Ø	ØB14	B6	ØØ		JFØ	DSPL4
21	ØB16	76	ØØ		JF1	KYBD4
22	ØB18	85			CLR	FØ
23	ØB19	A5			CLR	F1
24	ØB1A	16	Ø5		JTF	EXEC
25	ØB1C	C8			DEC	RØ
26	ØB1D	FØ			MOV	A,@RØ
27	ØB1E	Ø2			OUTL	BUS,A
28	ØB1F	E8	12		DJNZ	RØ,EXIT_A
29			ØØ		END	ØBØØH

Errors= Ø

33 END

Table 4-2. Assembler Output Listing (Cont'd)

FILE: EXEC4:SAVE **CROSS-REFERENCE TABLE** **PAGE 2**

LINE#	SYMBOL	TYPE	REFERENCES
	A	A	9,11,15,18,19,26,27
	BUS	A	27
	CNT	A	4
2	DSPL4	E	2Ø
9	EXEC	A	24
18	EXIT_A	A	14,28
	FØ	A	22
	F1	A	23
	I	A	7
2	KYBD4	E	21
13	LOOP_A	A	17
	MBØ	A	5
	PSW	A	19
	RØ	A	15,16,25,26,28
	R1	A	1Ø,17
	R2	A	18
	RBØ	A	6
	T	A	11,12
	TCNTI	A	8

NOTE: In the cross-reference table, the letter listed under the TYPE column has the following definition:

A = Absolute
C = Common (COMN)
D = Data (DATA)
E = External
M = Multiple Defined;
P = Program (PROG)
R = Predefined Register
U = Undefined

Table 4-3. Assembler Output Listing with Errors

FILE: EXCT: **HEWLETT-PACKARD: INTEL 8048 ASSEMBLER**

LINE	LOC	CODE	ADDR		SOURCE STATEMENT	
				"8048" LIST XREF		
1						
2					GLB	DSPL
3	ØFFF				ORG	ØFFFH
4	ØØØØ				DATA	
5	ØØØØ	2D		DSPL	XCH	A,R5
6	ØØØ1	94			MOVX	@R4,A
ERROR-IO						
7	ØØØ2	21			XCH	A,@R1
8	ØØØ3	E6	ØØ		JNC	DSPL
9	ØØØ5	86	ØB		JNI	LOOP_C
1Ø	ØØØ7	6Ø			ADD	A,@RØ
11	ØØØ8	51			ANL	A,@R1
12	ØØØ9	FF			ANL	A,ØFFH
ERROR—IO, see line 6						
13	ØØØA	8F			ORLD	P7,A
14	ØØØB	E7		LOOP_C	RL	A
15	ØØØC	DC			XRL	A,R4
16	ØØØD	F7			RLC	A
17	ØØØE	D1			XRL	A,@R1
18	ØØØF	3Ø			XCHD	A,@RØ
19	ØØ1Ø	26	ØØ		JNTØ	DSPL
2Ø	ØØ12	46	18		JNT1	END_C
21	ØØ14	36	ØØ		JTØ	LOOP_C
22	ØØ16	56	ØØ		JT1	DSPL
23	ØØ18	83		END_C	RET	
24					END	

Errors = 2, previous error at line 12

IO - Invalid Operand, Invalid or unexpected operand encountered, or operand is missing.

Table 4-3. Assembler Output Listing with Errors (Cont'd)

FILE: EXCT: **CROSS-REFERENCE TABLE** **PAGE 2**

LINE#	SYMBOL	TYPE	REFERENCES
	A	A	5,6,7,1Ø,11,12,13,14,15,16,17,18
5	DSPL	D	2,8,19,22
23	END_C	D	2Ø
14	LOOP_C	D	9,21
	P7	A	13
	RØ	A	18
	R1	A	7,11,17
	R4	A	6,1518
	R5	A	5

NOTE: Error messages are inserted immediately following the statement where the error occurs. All error messages (after the first error message) will contain a statement which points to the statement where the last error occurred. At the end of the source program listing, an error summary statement will be printed. The summary will contain a statement as to the total number of errors noted, along with a line reference to the previous error. It will also define all error codes listed in the source program listing.

The primary purpose of the error statement that points to the line number where the previous error occurred is to facilitate location of errors. Since some programs may be many pages in length, this feature helps the programmer locate errors quickly (as opposed to thumbing through each page of the program).

Chapter 5
Instruction Set Summary

General

All mnemonic instructions are summarized in this chapter in alphabetical order.

Each instruction consists of a mnemonic code and up to two operands. Descriptive symbols used in this chapter to represent items in mnemonic definitions are as follows:

Symbol	Description
$\overline{A}$	Complement of Register A
AC	Auxiliary Carry
addr	12-bit ROM address
Bb	Bit identifier (b = Ø-7)
BS	Bank switch
$\overline{C}$	Complement of carry flag
CRR	Conversion Result Register
D	4-bit expression
data	8-bit expression
DBF	Memory bank flip-flop
$\overline{FØ}$	Complement of Flag Ø
$\overline{F1}$	Complement of Flag 1

Symbol	Description
P	'In page' operation
PC	Program counter
Pp	Port designator (p = 1,2 or 4-7)
Rr	Register designator (r = Ø-7)
SP	Stack pointer
T	Timer
TØ	Test Ø
T1	Test 1
TF	Timer flag
<---	Transfer into
<--->	Exchange content
.	Boolean AND
⊕	Exclusive OR
⊙	Inclusive OR

Predefined Symbols

The following symbols are reserved. They have special meaning to the assembler and cannot appear as user-defined symbols.

Symbol	Definition
A	Register A
BUS	BUS port
C	Carry flag
CLK	Clock
CNT	Counter register
DBB	Data bus buffer
FØ	Flag Ø
F1	Flag 1
I	Interrupt
IBF	Input buffer flag
MBØ	Memory bank Ø
MB1	Memory bank 1
OBF	Output buffer flag
P1-P7	Ports P1 through P7
PSW	Program Status Word
RØ-R7	Register R1 through R7
RBØ	Register bank Ø

Symbol	Definition
RB1	Register bank 1
TCNT	Timer/Counter
TCNTI	Timer/Counter interrupt
@	Indirect address prefix
#	Immediate data prefix
$	Program counter content
(_)	Content of _

8048 Series Instruction Set Summary

Add to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ADD	A,RØ	68
	ADD	A,R1	69
	ADD	A,R2	6A
	ADD	A,R3	6B
	ADD	A,R4	6C
	ADD	A,R5	6D
	ADD	A,R6	6E
	ADD	A,R7	6F
	ADD	A,@RØ	6Ø
	ADD	A,@R1	61
	ADD	A,#data	Ø3

The ADD A,_ _ instruction adds the content of the designated register, memory location, or immediate data to the content of register A. The result of the operation will be stored in register A.

Symbolic Operation: (A) <--- (A)+(Rr) where r=Ø-7;
(A) <--- (A)+((Rr)) where r=Ø-1;
(A) <--- (A)+data

The carry flag bit will be affected by the result of the operation.

Example:

Label	Operation	Operand	Comment
	ADD	A,R6	

This instruction adds the content of register 6 to the content in register A. The following instruction:

Label	Operation	Operand	Comment
	ADD	A,@RØ	

will add the content of the memory location address by the first 6 bits (Ø-5) in register RØ to the content of register A.

Add with carry to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ADDC	A,RØ	78
	ADDC	A,R1	79
	ADDC	A,R2	7A
	ADDC	A,R3	7B
	ADDC	A,R4	7C
	ADDC	A,R5	7D
	ADDC	A,R6	7E
	ADDC	A,R7	7F
	ADDC	A,@RØ	7Ø
	ADDC	A,@R1	71
	ADDC	A,#data	13

The ADDC A,_ _ instruction adds the content of the carry flag bit to the content of register A. The carry flag bit will then be reset to zero. The content of the designated register, memory location, or immediate data, will then be added to the content of register A. The result of the operation will be stored in register A.

Symbolic Operation: (A) <--- (A)+(Rr)+(C) where r=Ø-7;
(A) <--- (A)+((Rr))+(C) where r=Ø-1;
(A) <--- (A)+(C)+data

The carry flag bit will be affected by the result of the operation.

Example:

Label	Operation	Operand	Comment
	ADDC	A,R2	

This instruction adds the carry flag bit and the content of register 2 to the content of register A. The carry flag bit will indicate the result of operation (overflow).

Logical AND Register A with Designated Mask

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ANL	A,RØ	58
	ANL	A,R1	59
	ANL	A,R2	5A
	ANL	A,R3	5B
	ANL	A,R4	5C
	ANL	A,R5	5D
	ANL	A,R6	5E
	ANL	A,R7	5F
	ANL	A,@RØ	5Ø
	ANL	A,@R1	51
	ANL	A,#data	53

A logical "AND" operation will be performed between the byte specified by the operand field and the content of register A. The result of the operation will be stored in register A.

Symbolic Operation: (A) <--- (A) (Rr) where r=Ø-7;
(A) <--- (A) ((Rr)) where r=Ø-1;
(A) <--- (A) data

Example:

Label	Operation	Operand	Comment
	ANL	A,@RØ	

ANL A,_ _ (Cont'd)

This instruction will perform a logical "AND" operation on the data in register A and the mask contained in the memory location referenced by bits Ø-5 in register RØ.

NOTE

A mask for a logical operation can reside anywhere in resident data memory. Logical operations cannot reference external memory.

Logical AND BUS with Immediate Data

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ANL	BUS,#data	98

Not applicable to 8021, 8022, 8041, 8741.

This instruction will logically "AND" the byte of data on the BUS port with the immediate data specified in the operand field. The result of the operation will remain on the BUS port.

Symbolic Operation: (BUS) <--- (BUS) data

Example:

Label	Operation	Operand	Comment
	ANL	BUS,#ØFØH	

This instruction will logically "AND" the data on the BUS port with the binary mask 1111ØØØØ.

ANL P_,#data

Logical AND Port_ with Immediate Data

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ANL	P1,#data	99
	ANL	P2,#data	9A

Not applicable to 8021, 8022.

This instruction will perform a logical "AND" operation on the data at the designated port (P1 or P2) using the immediate data given in the operand field. The result of the operation will remain on the specified port.

Symbolic Operation: (Pp) <--- (Pp) data where p = 1-2

Example:

Label	Operation	Operand	Comment
	ANL	P1,#ØFH	

This instruction will logically "AND" the data at P1 with the binary mask ØØØØ1111.

Logical AND Port_ with Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ANLD	P4,A	9C
	ANLD	P5,A	9D
	ANLD	P6,A	9E
	ANLD	P7,A	9F

This instruction will logically "AND" the data at the designated port with the binary mask in register A. The result of the operation will remain on the specified port.

Symbolic Operation: (Pp) <--- (Pp) (A BØ-3) where p = 4-7

Call Subroutine

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CALL	address (page Ø)	14
	CALL	address (page 1)	34
	CALL	address (page 2)	54
	CALL	address (page 3)	74
	CALL	address (page 4)	94
	CALL	address (page 5)	B4
	CALL	address (page 6)	D4
	CALL	address (page 7)	F4

NOTE: page = 256 bytes

This instruction is used for entering subroutines. It will push the program counter and bits 4-7 of the program status word (PSW) onto the stack. The stack pointer (bits Ø-2 of the PSW) will be updated. Program control will then be passed to the address specified in the operand field.

Upon return from the subroutine, execution of the program will continue at the instruction following the CALL instruction routine.

Symbolic Operation: ((SP)) <--- (PC), (PSW B4-7);
(SP) <--- (SP)+1;
(PC B8-1Ø) <--- addr B8-1Ø;
(PC BØ-7) <--- addr BØ-7;
(PC B11) <--- memory bank flip-flop

CLR A

Clear Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CLR	A	27

This instruction will reset the content of register A to zero (clears the register).

Symbolic Operation: A <--- Ø

CLR C

Clear Carry Flag Bit

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CLR	C	97

This instruction will reset the carry flag bit to zero.

Symbolic Operation: C <--- Ø

CLR F_

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CLR	FØ	85
	CLR	F1	A5

Not applicable to 8021, 8022.

This instruction will clear (reset to zero) the flag designated in the operand field.

Symbolic Operation: (FØ) <--- Ø;
(F1) <--- Ø

CPL A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CPL	A	37

This instruction will perform the one's complement on the content of register A. Each one bit will be changed to zero and each zero bit will be changed to one.

Symbolic Operation: (A) <--- $\overline{(A)}$

Example:

Label	Operation	Operand	Comment
	CPL	A	;prior to instruction ;content of register ;A=1Ø1Ø1Ø1ØB

This instruction will replace the current content of register A (1Ø1Ø1Ø1ØB) with Ø1Ø1Ø1Ø1B.

CPL C

Complement Carry Flag Bit

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CPL	C	A7

This instruction will complement the carry flag bit. If it was a one, it will be reset to zero. If it was a zero, it will be set to one.

Symbolic Operation: (C) <--- $\overline{(C)}$

CPL F_ _

Complement Flag_ _

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CPL	FØ	95
	CPL	F1	B5

Not applicable to 8021, 8022.

This instruction will complement the flag (FØ or F1) designated by the operand field. If the flag was set to one, it will be reset to zero. If the flag was zero, it will be set to one.

Symbolic Operation: (FØ) <--- $\overline{(FØ)}$;

(F1) <--- $\overline{(F1)}$

DA

Decimal Adjust Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DA	A	57

This instruction will adjust the 8-bit content of register A to form two 4-bit binary coded decimal (BCD) digits.

If the low-order nibble (bits Ø-3) is greater than nine, or if the auxiliary carry flag bit is one, register A will be incremented by six.

If the high-order nibble (bits 4-7) is greater than nine, or if the carry flag bit is one, the high-order nibble will be increased by six. If an overflow occurs, the carry flag bit will be set to one; otherwise, it will be reset to zero.

The carry and auxiliary carry flags will be affected by this operation.

DEC A

Decrement Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DEC	A	Ø7

This instruction will decrement the content of register A by one.

Symbolic Operation: (A) <--- (A)-1

Decrement Register R_

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DEC	RØ	C8
	DEC	R1	C9
	DEC	R2	CA
	DEC	R3	CB
	DEC	R4	CC
	DEC	R5	CD
	DEC	R6	CE
	DEC	R7	CF

Not applicable to 8021, 8022.

This instruction will decrement the content of the register designated in the operand field by one.

Symbolic Operation: (Rr) <--- (Rr)-1 where r = Ø-7

DIS I

Disable External Interrupt

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DIS	I	15

Not applicable to 8021.

This instruction will disable external interrupts in the 8048 microprocessor.

This instruction will disable the write interrupt in the 8041 microprocessor.

DIS TCNTI

Disable Timer/Counter Interrupt

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DIS	TCNTI	35

Not applicable to 8021.

This instruction will disable timer/counter interrupts and any timer interrupt request will be cleared. The interrupt sequence will not be initiated by an overflow, but the timer flag will be set and time accumulation continued.

Decrement Register and Test

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DJNZ	RØ,address	E8
	DJNZ	R1,address	E9
	DJNZ	R2,address	EA
	DJNZ	R3,address	EB
	DJNZ	R4,address	EC
	DJNZ	R5,address	ED
	DJNZ	R6,address	EE
	DJNZ	R7,address	EF

This instruction will decrement the register in the operand field. If the register contains all zeros, program execution will continue with the next instruction. If the designated register contents are not zero when tested, program control will jump to the specified address.

Symbolic Operation: (Rr) <--- (Rr)-1 where r = Ø-7

If Rr = Ø:
(PC BØ-7) <--- addr

Examples:

Label	Operation	Operand	Comment
	DJNZ	R3,SAM	
	ADD	A,R1	

This instruction will decrement register R3. If the content of R3 is not all zeros, a jump will occur to a location designed by SAM. If the content of register R3 is all zeros, program execution will continue with the ADD instruction.

EN I

Enable External Interrupt

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	EN	I	Ø5

Not applicable to 8021.

This instruction will enable external interrupts in the 8048 microprocessor.

This instruction will enable the write interrupt in the 8041 microprocessor.

EN TCNTI

Enable Timer/Counter Interrupt

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	EN	TCNTI	25

Not applicable to 8021.

This instruction will enable the timer/counter interrupts. An overflow of the timer/counter will initiate the interrupt sequence.

ENTO CLK

Enable Clock Output

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ENTO	CLK	75

Not applicable to 8021, 8022, 8041, 8741.

This instruction will enable the test Ø pin so that it will act as the clock output. This function will be disabled by a system reset.

IN A,DBB

Input DBB Data to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	IN	A,DBB	22

Applicable to 8041, 8741 only.

This instruction will transfer data from the data bus buffer to register A.

Input Port_ _ Data to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	IN	A,PØ	Ø8 (8021, 8022 only)
	IN	A,P1	Ø9
	IN	A,P2	ØA

This instruction will transfer the data on the designated port into register A.

Symbolic Operation: (A) <--- (Pp) where p = 1-2

Increment Register

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	INC	A	17
	INC	RØ	18
	INC	R1	19
	INC	R2	1A
	INC	R3	1B
	INC	R4	1C
	INC	R5	1D
	INC	R6	1E
	INC	R7	1F
	INC	@RØ	1Ø
	INC	@R1	11

This instruction will increment the content of the designated register or resident memory location by one.

NOTE

External data memory content cannot be incremented directly.

Symbolic Operation: (A) <--- (A)+1;
(Rr) <--- (Rr)+1 where r = Ø-7;
((Rr)) <--- ((Rr)) +1 where r = Ø-1

INS

Strobe Input of BUS Data to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	INS	A,BUS	Ø8

Not applicable to 8021, 8022, 8041, 8741.

This instruction will transfer the data on the BUS port into register A when the RD (read) pulse goes low.

Symbolic Operation: (A) <--- (BUS)

JB_ _

Jump if Register A Bit is Set

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JBØ	address	12
	JB1	address	32
	JB2	address	52
	JB3	address	72
	JB4	address	92
	JB5	address	B2
	JB6	address	D2
	JB7	address	F2

Not applicable to 8021, 8022.

This instruction will cause program control to pass to the address specified by the operand if the designated bit in register A was set to one.

Symbolic Operation: If Bb = 1 (set): where b = Ø-7
(PC BØ-7) <--- addr

If Bb = Ø (reset):
(PC) = (PC)+2

Jump if Carry Flag Bit is Set

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JC	address	F6

This instruction will cause program control to pass to the address specified by the operand if the carry flag bit is set to one.

Symbolic Operation: If C = 1 (set):
(PC BØ-7) <--- addr

If C = Ø (reset):
(PC) = (PC)+2

Jump if Flag_ _ is Set

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JFØ	address	B6
	JF1	address	76

Not applicable to 8021, 8022.

This instruction will cause program control to pass to the address specified by the operand if the designated flag is set to one.

Symbolic Operation: If FØ = 1 (set):
(PC BØ-7) <--- addr

If FØ = Ø (reset):
(PC) = (PC)+2

If F1 = 1 (set):
(PC) BØ-7) <--- addr

If F1 = Ø (reset):
(PC) <--- (PC)+2

Direct Jump within 2K Block

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JMP	address	page Ø = Ø4 page 1 = 24 page 2 = 44 page 3 = 64 page 4 = 84 page 5 = A4 page 6 = C4 page 7 = E4

This instruction allows the user to jump unconditionally across page boundaries. The instruction addresses program memory locations directly by replacing bits Ø-1Ø of the program counter with the specified address. Bit 11 of the program counter will be determined by the most recent SEL MB instruction.

Symbolic Operation: (PC B8-1Ø) <--- addr B8-1Ø
(PC BØ-7) <--- addr BØ-7
(PC B11) <--- memory bank flip-flop

JMPP

Indirect Jump within Page

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JMPP	@A	B3

This instruction causes a jump to memory location addressed by the content of register A.

Symbolic Operation: (PC BØ-7) <--- ((A))

JNC

Jump if Carry Not Set

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JNC	address	E6

This instruction will pass program control to the specified address if the carry flag bit is not set. (C=Ø).

Symbolic Operation: If C = Ø (reset):
(PC BØ-7) <--- addr

If C = 1 (set):
(PC) = (PC)+2

JNI

Jump if Interrupt Input Low

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JNI	address	86

Not applicable to 8021, 8022, 8041, 8741.

This instruction will pass program control to the specified address if an external interrupt has been signaled (interrupt input signal = Ø).

Symbolic Operation: If I = Ø (reset):
(PC BØ-7) <--- addr

If I = 1 (set):
(PC) = (PC)+2

JNIBF

Jump if Input Buffer Flag is Not Set

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JNIBF	address	D6

Applicable to 8041, 8741 only.

This instruction will pass program control to the specified address if the input buffer flag is not set.

Symbolic Operation: If I = Ø (reset):
(PC BØ-7) <--- addr

If I = 1 (set):
(PC) = (PC)+2

Jump if Test_ _ Low

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JNTØ	address	26 (not in 8021)
	JNT1	address	46

Depending upon the selected opcode, this instruction will pass program control to the specified address if the test Ø or test 1 signal is low (=Ø).

Symbolic Operation: If TØ = Ø (reset):
(PC BØ-7) <--- addr

If TØ = 1 (set):
(PC) = (PC)+2

If T1 = Ø (reset):
(PC BØ-7) <--- addr

If T1 = 1 (set):
(PC) = (PC)+2

JNZ

Jump if Register A Not Zero

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JNZ	address	96

This instruction will pass program control to the specified address if the content of register A is nonzero.

Symbolic Operation: If A ≠ Ø:
(PC BØ-7) <--- addr

If A = Ø:
(PC) = (PC)+2

JOBF

Jump if Output Buffer Flag is Set

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JOBF	address	86

Applicable to 8041, 8741 only.

This instruction will pass program control to the specified address if the output buffer flag is set.

Symbolic Operation: If OBF = 1 (set):
(PC BØ-7) <--- addr

Jump if Timer Flag Set

SYNTAX

Label	Operation	Operand	Object Code (hex)
	JTF	address	16

This instruction will pass program control to the specified address if the timer flag is set (=1).

Testing the timer flag will reset it to zero.

Symbolic Operation: If TF = 1 (set):
(PC BØ-7) <--- addr

If TF = Ø (reset):
(PC) = (PC)+2

Jump if Test_ _ High

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JTØ	address	36 (8048 and 8041 only)
	JT1	address	56

Depending upon the selected opcode, this instruction will pass program control to the specified address if the test Ø or test 1 signal is high (=1).

Symbolic Operation: If TØ = 1 (set):
(PC BØ-7) <--- addr

If TØ = Ø (reset):
(PC) = (PC)+2

If T1 = 1 (set):
(PC BØ-7) <--- addr

If T1 = Ø (reset):
(PC) = (PC)+2

Jump if Register A Zero

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JZ	address	C6

This instruction will pass program control to the specified address if the content of register A is all zeros.

Symbolic Operation: If A = Ø:
(PC BØ-7) <--- addr

If A ≠ Ø:
(PC) = (PC)+2

Move Designated Contents to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOV	A,RØ	F8
	MOV	A,R1	F9
	MOV	A,R2	FA
	MOV	A,R3	FB
	MOV	A,R4	FC
	MOV	A,R5	FD
	MOV	A,R6	FE
	MOV	A,R7	FF
	MOV	A,@RØ	FØ
	MOV	A,@R1	F1
	MOV	A,#data	23
	MOV	A,PSW	D7 (Not in 8021, 8022)
	MOV	A,T	42

This instruction moves the content of the designated register, memory location, or immediate data into register A. The content of the designated register or memory location is unaffected by this instruction.

Symbolic Operation: (A) <--- (Rr) where r = Ø-7
(A) <--- ((Rr)) where r = Ø-1
(A) <--- data
(A) <--- (PSW)
(A) <--- (T)

MOV PSW,A

Move Register A Contents to PSW

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOV	PSW,A	D7

Not applicable to 8021, 8022.

This instruction moves the content of register A into the program status word (PSW). The content of register A will not be affected.

All condition bits and the stack pointer are affected by this move.

Symbolic Operation: (PSW) <--- (A)

Move Content to Designated Register

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOV	RØ,A	A8
	MOV	R1,A	A9
	MOV	R2,A	AA
	MOV	R3,A	AB
	MOV	R4,A	AC
	MOV	R5,A	AD
	MOV	R6,A	AE
	MOV	R7,A	AF
	MOV	RØ,#data	B8
	MOV	R1,#data	B9
	MOV	R2,#data	BA
	MOV	R3,#data	BB
	MOV	R4,#data	BC
	MOV	R5,#data	BD
	MOV	R6,#data	BE
	MOV	R7,#data	BF

This instruction moves the content of register A or the immediate data into the designated register.

The content of register A is unaffected by this move.

Symbolic Operation: (Rr) <--- (A) where r=Ø-7
(Rr) <--- data where r=Ø-7

MOV @R,_ _

Move Content to Data Memory

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOV	@RØ,A	AØ
	MOV	@R1,A	A1
	MOV	@RØ,#data	BØ
	MOV	@R1,#data	B1

This instruction moves the content of register A or the immediate data into the memory location whose address was specified by bits Ø-5 in the designated register (RØ or R1).

The contents of register A and the designated register are unaffected by this move.

Symbolic Operation: ((Rr)) <--- (A) where r = Ø-1

((Rr)) <--- data where r = Ø-1

MOV T,A

Move Content of Register A to Timer/Counter

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOV	T,A	62

This instruction moves the content in register A to the timer/counter register. The content of register A is unaffected by this move.

Symbolic Operation: (T) <--- (A)

MOVD A,P_ _

Move Port_ _ Data to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOVD	A,P4	ØC
	MOVD	A,P5	ØD
	MOVD	A,P6	ØE
	MOVD	A,P7	ØF

This instruction moves the data on the specified input/output port into bits Ø-3 of register A. Bits 4-7 of register A are zeroed.

Symbolic Operation: (A BØ-3) <--- (Pp) where p = 4-7
(A B4-7) <--- Ø

MOVD P_,A

Move Register A Data to Port_ _

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOVD	P4,A	3C
	MOVD	P5,A	3D
	MOVD	P6,A	3E
	MOVD	P7,A	3F

This instruction moves bits Ø-3 of register A to the designated input/output port. Bits 4-7 of register A will be unaffected.

Symbolic Operation: (Pp) <--- (A BØ-3) where p = 4-7

MOVP A,@A

Move Current Page Data to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOVP	A,@A	A3

This instruction moves the content of the program memory location addressed by register A into register A. Only bits Ø-7 of the program counter are used, limiting the program memory reference to the current page.

Symbolic Operation: (PC BØ-7) <---(A)
(A) <--- ((PC))

NOTE

If this instruction appears in location 255 of a program memory page, the @A portion of the operand will address a location in the following page.

MOVP3 A,@A

Move Page 3 Data to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOVP3	A,@A	E3

Not applicable to 8021, 8022.

This instruction moves the content of the program memory location (within page 3) addressed by register A into register A.

The program counter will be restored after this operation.

Symbolic Operation: (PC BØ-7) <--- (A)
(PC B8-11) <--- ØØ11B
(A) <--- ((PC))

MOVX A,@R

Move Content from External memory to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOVX	A,@RØ	8Ø
	MOVX	A,@R1	81

Not applicable to 8021, 8022, 8041, 8741.

This instruction moves the content of external memory location addressed by the designated register into register A.

The content of the designated register (RØ or R1) is unaffected by this move.

Symbolic Operation: (A) <--- ((Rr)) where r = Ø-1

MOVX @R_,A

Move Content of Register A to External Memory

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	MOVX	@RØ,A	9Ø
	MOVX	@R1,A	91

Not applicable to 8021, 8022, 8041, 8741.

This instruction moves the content of register A to the external memory location addressed by the designated register.

The content of the designated register (RØ or R1) is unaffected by this move.

Symbolic Operation: ((Rr)) <--- (A) where r = Ø-1

NOP

No Operation

SYNTAX:

Label	Operation	Operand	Object Code (hex)
[symbol]	NOP		ØØ

This instruction performs no operation. The label field is optional; however, labels can be assigned to a NOP instruction for program branching. Operands are not permitted with a NOP instruction.

ORL A,_ _

Logical OR Register A with Register Mask

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ORL	A,RØ	48
	ORL	A,R1	49
	ORL	A,R2	4A
	ORL	A,R3	4B
	ORL	A,R4	4C
	ORL	A,R5	4D
	ORL	A,R6	4E
	ORL	A,R7	4F
	ORL	A,@RØ	4Ø
	ORL	A,@R1	41
	ORL	A,#data	43

This instruction performs a logical OR operation on the content of register A with the content of the designated register, memory location, or immediate data.

Symbolic Operation: (A) <--- (A) (Rr) where r = Ø-7
(A) <--- (A) ((Rr)) where r = Ø-1
(A) <--- (A) data

Logical OR BUS with Immediate Data

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ORL	BUS,#data	88

Not applicable to 8021, 8022, 8041, 8741.

This instruction performs a logical OR operation on the data on the BUS port with the immediately-specified data.

NOTE

This instruction assumes a prior specification of an 'OUTL BUS,A' instruction.

Symbolic Operation: (BUS) <--- (BUS) data

ORL P_,#data

Logical OR Port __ with Immediate Data

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ORL	P1,#data	89
	ORL	P2,#data	8A

Not applicable to 8021, 8022, 8041, 8741.

This instruction performs a logical OR operation on the data on the designated port with the immediately-specified data.

Symbolic Operation: (Pp) <--- (Pp) data where p = 1-2

ORLD P_,A

Logical OR Port_ with Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ORLD	P4,A	8C
	ORLD	P5,A	8D
	ORLD	P6,A	8E
	ORLD	P7,A	8F

This instruction performs a logical OR operation on the data at the designated port with bits Ø-3 of register A.

The result of the operation is placed on the designated port.

Symbolic Operation: (Pp) <--- (Pp) (A BØ-3) where p = 4-7

Output Register A Data to Data Bus Buffer

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	OUT	DBB,A	Ø2

Applicable to 8041, 8741 only.

This instruction transfers the data in register A to the data bus buffer.

OUTL BUS,A

Output Register A Data to BUS

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	OUTL	BUS,A	Ø2

Not applicable to 8021, 8022, 8041, 8741.

This instruction transfers the data in register A to the BUS port where it will be latched. The latched data will remain valid until changed by another OUTL instruction.

NOTE

Any other instruction requiring the use of the BUS port (except INS) will destroy the latched contents on the BUS port.

Logical AND and OR operations on BUS data assume that an "OUTL BUS,A" instruction has been executed previously.

Symbolic Operation: (BUS) <--- (A)

OUTL P_,A

Output Register A Data to Port_ _

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	OUTL	PØ,A	9Ø (8021 and 8022 only)
	OUTL	P1,A	39 (8048 and 8041 only)
	OUTL	P2,A	3A (8048 and 8041 only)

Applicable to: 8048, 8041, and 8021 microprocessors.

This instruction transfers the data in register A to the designated port where it will be latched.

Symbolic Operation: (Pp) <--- (A) where p = 1-2

RAD

Move Conversion Result to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RAD		8Ø

Applicable to 8022 only.

This instruction moves the contents of the A/D conversion result register to register A.

Symbolic Operation: (A) <--- (CRR)

RET

Return without PSW Restore

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RET		83

This instruction decrements the stack pointer (PSW bits Ø-2); Then the program counter is restored from the stack. Program status word (PSW) bits 4-7 are not restored.

Symbolic Operation: (SP) <--- (SP)-1
(PC) <--- ((SP))

RETI

Return from Interrupt

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RETI		93

Applicable to 8022 only.

This instruction decrements the stack pointer and restores the program counter from the stack. Interrupt input logic is re-enabled.

Symbolic Operation: (SP) <--- (SP)-1
(PC) <--- ((SP))+1

RETR

Return with PSW Restore

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RETR		93

Not applicable to 8021, 8022.

This instruction decrements the stack pointer (PSW bits Ø-2) and restores the program counter and bits 4-7 of the program status word (PSW) from the stack.

Symbolic Operation: (SP) <--- (SP)-1
(PC) <--- ((SP))
(PSW B4-7) <--- ((SP))

RL

Rotate Left without Carry

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RL	A	E7

This instruction rotates the content of register A one bit position to the left. Bit 7 of the register will be rotated into the bit Ø position.

Symbolic Operation: A(B+1) <--- A(B) where B = Ø-6
A(BØ) <--- A(B7)

RLC

Rotate Left Through Carry

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RLC	A	F7

This instruction rotates the content of register A one bit position to the left. Bit 7 will replace the carry flag bit and the carry flag bit will be rotated into the bit Ø position of register A.

The carry flag bit is affected by the result of this operation.

Symbolic Operation: A(B+1) <--- A(B) where B = Ø-6
A(BØ) <--- (C)
(C) <--- A(B7)

RR

Rotate Right without Carry

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RR	A	77

This instruction rotates the content of register A one bit position to the right. Bit Ø of register A will be rotated into the bit 7 position.

Symbolic Operation: A(B) <--- A(B+1) where B = Ø-6
A(B7) <--- A(BØ)

RRC

Rotate Right Through Carry

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RRC	A	67

This instruction rotates the content of register A one bit position to the right. Bit Ø of register A will replace the carry flag bit and the carry flag bit will be rotated into the bit 7 position of register A.

The carry flag bit is affected by the result of this operation.

Symbolic Operation: A(B) <--- A(B+1) where B = Ø-6
A(B7) <--- (C)
(C) <--- A(BØ)

SEL AN_ _

Select Analog Input

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	SEL	ANØ	95
	SEL	AN1	85

Applicable to 8022 only.

This instruction selects either the zero or the One input to the A/D converter, and the conversion process is started. Restarting a sequence deletes the sequence in progress.

SEL MB_ _

Select Memory Bank

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	SEL	MBØ	E5
	SEL	MB1	F5

Not applicable to 8021, 8022, 8041, 8741.

The SEL MBØ instruction resets bit 11 of the program counter to zero causing all references to program memory to fall within the range Ø-2Ø47.

The SEL MB1 instruction sets bit 11 of the program counter to one causing all references to program memory to fall within the range 2Ø48-4Ø95.

Symbolic Operation: (PC B11) <--- Ø
(PC B11) <--- 1

Select Register Bank

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	SEL	RBØ	C5
	SEL	RB1	D5

Not applicable to 8021, 8022.

The SEL RBØ instruction resets bit 4 of the program status word to zero causing RAM locations Ø-7 to be selected for the working register.

The SEL RB1 instruction sets bit 4 of the program status word to one designating locations 24-31 as the working registers. This bank of registers can be used as an extension of the first bank, or can be reserved for use during interrupt service subroutines.

Symbolic Operation: (PSW B4) <--- Ø
(PSW B4) <--- 1

STOP TCNT

Stop Time/Event Counter

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	STOP	TCNT	65

This instruction stops both time accumulation and event counting.

STRT CNT

Start Event Counter

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	STRT	CNT	45

This instruction enables the test 1 (T1) pin for use as the event-counter input and then starts the counter. The event-counter register is incremented with each high-to-low transition detected at pin T1.

STRT T

Start Timer

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	STRT	T	55

This instruction initiates time accumulation in the timer register. The register will be incremented every 32 instruction cycles. The pre-scaler where the 32 cycles are counted will be cleared, but the time register will not.

SWAP

Swap 4-bit Data within Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	SWAP	A	47

This instruction exchanges bits Ø-3 and bits 4-7 in register A.

Symbolic Operation: (A B4-7) <---> (A BØ-3)

XCH A,_ _

Exchange Register-Register A Content

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	XCH	A,RØ	28
	XCH	A,R1	29
	XCH	A,R2	2A
	XCH	A,R3	2B
	XCH	A,R4	2C
	XCH	A,R5	2D
	XCH	A,R6	2E
	XCH	A,R7	2F
	XCH	A,@RØ	2Ø
	XCH	A,@R1	21

This instruction exchanges the content of the designated register or memory location and the content of register A.

Symbolic Operation: (A) <---> (Rr) where r = Ø-7
(A) <---> ((Rr)) where r = Ø-1

Exchange Register A and Data Memory 4-bit Data

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	XCHD	A,@RØ	3Ø
	XCHD	A,@R1	31

This instruction will exchange bits Ø-3 of register A with bits Ø-3 of the data memory location addressed by the designated register. Bits 4-7 of the data memory location, bits 4-7 of register A, and the content of the addressing register will be unaffected by this instruction.

Symbolic Operation: (A BØ-3) <---> ((Rr BØ-3)) where r = Ø-1

Logical Exclusive OR of Register A with Register Data

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	XRL	A,RØ	D8
	XRL	A,R1	D9
	XRL	A,R2	DA
	XRL	A,R3	DB
	XRL	A,R4	DC
	XRL	A,R5	DD
	XRL	A,R6	DE
	XRL	A,R7	DF
	XRL	A,@RØ	DØ
	XRL	A,@R1	D1
	XRL	A,data	D3

This instruction performs an exclusive OR operation on the content of register A and the content of the designated register, memory location, or immediate data.

Symbolic Operation: (A) <--- (A) (Rr) where r = Ø-7
(A) <--- (A) ((Rr)) where r = Ø-1
(A) <--- (A) data

64846-90902

PRINTED U.S.A.