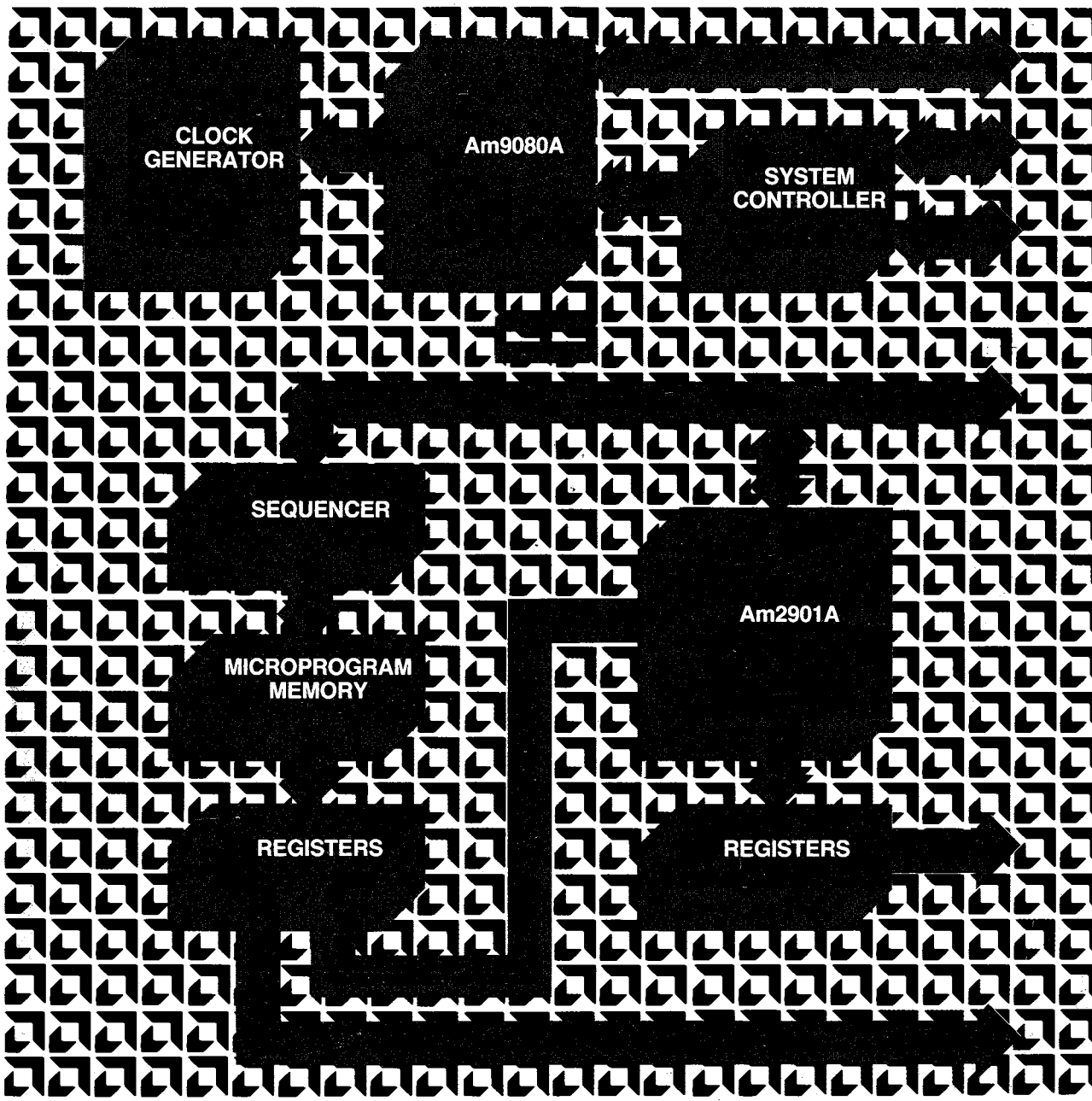
Advanced Micro Devices

An Emulation of the Am9080A

$5.00



Advanced Micro Devices

An Emulation of the Am9080A

An Example of A Microprogrammed Machine

By Moshe Shavit

AMPUB-064

TABLE OF CONTENTS

INTRODUCTION

The Am2901A Four-Bit Bipolar Microprocessor Slice and its associated support circuits have become the industry standard. However, even as this niche is already carved in the microprocessor industry, consideration must also be made to those users who either have not dealt with the speed and flexibility of the Am2900 Family or have utilized other types of microprocessing in their applications. Therefore, it is important that the needs of these users be met at least partially by allowing them to adapt to this new concept through the use of familiar tools.

This document describes the emulation of an Am9080A Eight-Bit MOS Microprocessor built around four Am2901A Bit Slices and three Am2909 Microprogram Sequencers. The primary purposes of this design are:

1. To demonstrate the use of the Am2901A, Am2909, and microprogramming in general, and
2. To emulate the Am9080A/Am8228 chip set in such a way as to better enable the user to make the transition to implementing the Am2900 Family in his system, particularly if he has previously been Am9080A-oriented.

The design was not intended to minimize component count or cycle times, but rather to demonstrate an emulation in the most straightforward manner. However, even though speed was not the first consideration in this design, the emulator needs, on the average, about 40% fewer clock cycles than does the Am9080A (see Table 1). Also, it can run with a maximum clock frequency of about 5MHz, which altogether is about a 4:1 speed improvement over the standard 2MHz Am9080A/Am8228 instruction set (see Table 7).

Furthermore, the emulator has the same input/output lines and controls as does the Am9080A/Am8228 set, i.e.:

1. A 16-bit wide Address Bus
2. An 8-bit wide Data Bus
3. $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{I/OR}}$, $\overline{\text{I/OW}}$, $\overline{\text{HLDA}}$, WAIT and $\overline{\text{INTA}}$ control outputs
4. HOLD, READY, INT, and RESET control inputs.

A complete list of the parts used in this design can be found at the end of the text.

GENERAL PHILOSOPHY

The Architecture

The heart of the emulator is a 16-bit wide ALU constructed with four Am2901A Four-Bit Bipolar Microprocessor Slices. Their internal registers are assigned as the Am9080A registers, including both the Program Counter and the Stack Pointer. The four Am2901A's are grouped in two pairs, a High Order Pair and a Low Order Pair. This scheme provides an easy access to the 16-bit register pairs (B-C, D-E, H-L) as well as to the 16-bit registers (PC, SP). The internal register allocations of the Am2901A's are depicted in Table 2. Note that for the High Order Pair (HOP), the Am2901A memory locations for each of the working registers are the same as the numbers used to identify these registers as source or destination registers in the Am9080A instructions. Thus, by properly decoding these bits in the Am9080A instructions and applying them to the A or B addresses of the Am2901A's, all of the Am9080A register-to-register instructions (approximately 25% of the total number of instructions) can be handled by only *one* set of microinstructions. This, of course, simplifies and shortens the microprogram.

Furthermore, addressing a register pair (B-C, D-E, H-L) is made easy by selecting bits 4 and 5 of the Am9080A instruction and applying them to the A or B address of the Am2901A's. This is true provided the register in the Low Order Pair (LOP) contains the same data as its corresponding register in the HOP. It can be accomplished merely by inverting the Least Significant Bit of the LOP register address whenever a single-byte (8-bit) operation is performed. For example, if the Am9080A instruction INR B (Increment Register B) is to be performed, its destination bits (000) are applied to the HOP B addresses, addressing Register 0. However, 001 is applied to the LOP B address, addressing Register 1, which again is B.

Executing double register operations (INX, DCX, LXI, etc.) does not require this Least Significant Bit (LSB) inversion, but for these instructions, the adjacent pairs of registers should always be updated. This can be accomplished by byte-swapping, e.g. taking Register H from the HOP (No. 4) and writing its contents into Register H of the LOP (No. 5), and at the same time taking Register L from the LOP (No. 4) and writing its contents into Register L of the HOP (No. 5). All of this can be done in one microcycle, and as the amount of such instructions is relatively small, the time consumption involved is negligible.

This system of register allocation uses only 3 bits of the microinstruction, but shortens by an appreciable amount both the microcode and the execution times. The registers adjacent to those allocated to A, PC, and SP are used as "scratchpads". The constants in Registers 12 and 13 are loaded in the initialization phase following a restart and are used in the restart instruction (RST) to properly mask the restart address bits from the Data Bus.

The Microprogram Control Unit

Figure 1 is a block diagram of the Microprogram Control portion of the emulator. This unit is a typical example of a computer control unit that uses three Am2909 Microprogram Sequencers. In an instruction fetch cycle, the Instruction Register latches the instruction appearing on the Data Bus and retains it during its execution. The output of this register is applied to the Mapping PROM (256 words by 12 bits) address inputs. This PROM maps the 8-bit instruction into the appropriate 12-bit microprogram memory address. The data ouputs of this Mapping PROM are connected to the D inputs of the Am2909's. Some Instruction Register bits also go to the ALU for register selection.

The Am2909 control lines are controlled by a Sequence PROM. Three of the Sequence PROM address inputs come from the Pipeline Register and a fourth is supplied by a sixteen-input Condition Code Multiplexer. Of its sixteen inputs, twelve are used: five for the Am9080A flags (CY, Z, S, P, AC), three for the micro-status bits (F=0, F_3, C_{n+4}), three for the INT, READY, and HOLD controls to the emulator, and one always TRUE input which the sequencer uses to execute an instruction unconditionally. The interrupt inputs are gated with one bit from the Sequence PROM and connected to the Am2909 OR inputs. This effectively allows the forcing of "1"'s into all of the OR inputs of the Am2909's when an interrupt is requested, while at the same time enabling the interrupt, forcing "1"'s into all of the microprogram address lines. At the highest available microprogram address, there is a JUMP to the interrupt handler.

Table 1. Clock Cycle Requirements for the Am9080A Emulation.

Op Code								Number Of Bytes	Clock Cycles		Assembly Mnemonic	Instruction Description
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		Am9080A	Am9080A Emulation*		
									DATA TRANSFER			
0	1	d	d	d	s	s	s	1	5	3	MOVr, r	Move register to register
0	1	1	1	0	s	s	s	1	7	5	MOVm, r	Move register to memory
0	1	d	d	d	1	1	0	1	7	5	MOVr, m	Move memory to register
0	0	d	d	d	1	1	0	2	7	5	MVI, r	Move to register, immediate
0	0	1	1	0	1	1	0	2	10	7	MVI, m	Move to memory, immediete
0	0	1	1	1	0	1	0	3	13	4	LDA	Load Acc, direct
0	0	0	0	1	0	1	0	1	7	5	LDAX B	Load Acc, indirect vie B & C
0	0	0	1	1	0	1	0	1	7	5	LDAX D	Load Acc, indirect via D & E
0	0	1	0	1	0	1	0	3	16	12	LHLD	Load H & L, direct
0	0	1	0	0	0	0	1	3	10	5	LXI H	Load H & L, immediate
0	0	0	1	0	0	1	1	3	10	5	LXI D	Load D & E, immediate
0	0	0	0	0	0	0	1	3	10	5	LXI B	Load B & C, immediate
0	0	1	1	0	0	0	1	3	10	5	LXI SP	Load steck pointer, immediate
0	0	1	0	0	0	1	0	3	16	11	SHLD	Store H & L, direct
0	0	1	1	0	0	1	0	3	13	9	STA	Store Acc, direct
0	0	0	0	0	0	1	0	1	7	5	STAX B	Store Acc, indirect via B & C
0	0	0	1	0	0	1	0	1	7	5	STAX D	Store Acc, indirect via D & E
1	1	1	1	1	0	0	1	1	5	3	SPHL	Transfer H & L to stack pointer
1	1	1	0	1	0	1	1	1	4	5	XCHG	Exchenge D & E with H & L
1	1	1	0	0	0	1	1	1	18	11	XTHL	Exchenge top of stack with H & L
1	1	0	1	1	0	1	1	2	10	6	IN	Input to Acc
1	1	0	1	0	0	1	1	2	10	5	OUT	Output from Acc
									CONTROL			
0	1	1	1	0	1	1	0	1	7	6	HLT	Halt and enter wait state
0	0	1	1	0	1	1	1	1	4	3	STC	Set carry flag
0	0	1	1	1	1	1	1	1	4	4-3	CMC	Complement carry flag
1	1	1	1	1	0	1	1	1	4	3	EI	Enable interrupts
1	1	1	1	0	0	1	1	1	4	3	DI	Disable interrupts
0	0	0	0	0	0	0	0	1	4	3	NOP	No operation
									BRANCHING			
1	1	0	0	0	0	1	1	3	10	7	JMP	Jump unconditionally
1	1	0	1	1	0	1	0	3	10	8-5	JC	Jump on carry
1	1	0	1	0	0	1	0	3	10	8-5	JNC	Jump on no carry
1	1	0	0	1	0	1	0	3	10	8-5	JZ	Jump on zero
1	1	0	0	0	0	1	0	3	10	8-5	JNZ	Jump on not zero
1	1	1	1	0	0	1	0	3	10	8-5	JP	Jump on positive
1	1	1	1	1	0	1	0	3	10	8-5	JM	Jump on minus
1	1	1	0	1	0	1	0	3	10	8-5	JPE	Jump on parity even
1	1	1	0	0	0	1	0	3	10	8-5	JPO	Jump on parity odd
1	1	0	0	1	1	0	1	3	17	11	CALL	Call unconditionally
1	1	0	1	1	1	0	0	3	17-11	12-5	CC	Call on carry
1	1	0	1	0	1	0	0	3	17-11	12-5	CNC	Call on no carry
1	1	0	0	1	1	0	0	3	17-11	12-5	CZ	Call on zero
1	1	0	0	0	1	0	0	3	17-11	12-5	CNZ	Call on not zero
1	1	1	1	0	1	0	0	3	17-11	12-5	CP	Call on positive
1	1	1	1	1	1	0	0	3	17-11	12-5	CM	Call on minus
1	1	1	0	1	1	0	0	3	17-11	12-5	CPE	Call on perity even
1	1	1	0	0	1	0	0	3	17-11	12-5	CPO	Call on perity odd
1	1	0	0	1	0	0	1	1	10	8	RET	Return unconditionally
1	1	0	1	1	0	0	0	1	11-5	9-3	RC	Return on carry
1	1	0	1	0	0	0	0	1	11-5	9-3	RNC	Return on no carry
1	1	0	0	1	0	0	0	1	11-5	9-3	RZ	Return on zero
1	1	0	0	0	0	0	0	1	11-5	9-3	RNZ	Return on not zero
1	1	1	1	0	0	0	0	1	11-5	9-3	RP	Return on positive
1	1	1	1	1	0	0	0	1	11-5	9-3	RM	Return on minus
1	1	1	0	1	0	0	0	1	11-5	9-3	RPE	Return on parity even
1	1	1	0	0	0	0	0	1	11-5	9-3	RPO	Return on perity odd
1	1	1	0	1	0	0	1	1	5	3	PCHL	Jump unconditionally, indirect via H & L
1	1	V	V	V	1	1	1	1	11	9	RST	Restart

*The number of clock cycles required to execute some instructions may vary, according to the conditions present at execution time. For example, a CMC (Complement Carry) is implemented in microcode, not in hardware. The carry is unconditionally reset while the previous status of this flag is examined. If it was already reset, a jump to the STC (Set Carry) routine is performed; otherwise, the next instruction is immedietely fetched.

Table 1. Clock Cycle Requirements for the Am9080A Emulation. (Cont.)

Op Code								Number Of Bytes	Clock Cycles		Assembly Mnemonic	Instruction Description
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		Am9080A	Am9080A Emulation		
									ARITHMETIC			
1	0	0	0	0	s	s	s	1	4	3	ADDr	Add register to Acc
1	0	0	0	1	s	s	s	1	4	4	ADCr	Add with carry register to Acc
1	0	0	0	0	1	1	0	1	7	5	ADDm	Add memory to Acc
1	0	0	0	1	1	1	0	1	7	5	ADCm	Add with carry memory to Acc
1	1	0	0	0	1	1	0	2	7	4	ADI	Add to Acc, immediate
1	1	0	0	1	1	1	0	2	7	5	ACI	Add with carry to Acc, immediate
0	0	0	0	1	0	0	1	1	10	4	DAD B	Double add B & C to H & L
0	0	0	1	1	0	0	1	1	10	4	DAD D	Double add D & E to H & L
0	0	1	0	1	0	0	1	1	10	4	DAD H	Double add H & L to H & L
0	0	1	1	1	0	0	1	1	10	4	DAD SP	Double add stack pointer to H & L
1	0	0	1	0	s	s	s	1	4	3	SUBr	Subtract register from Acc
1	0	0	1	1	s	s	s	1	4	4	SBBr	Subtract with borrow register from Acc
1	0	0	1	0	1	1	0	1	7	5	SUBm	Subtract memory from Acc
1	0	0	1	1	1	1	0	1	7	5	SBBm	Subtract with borrow memory from Acc
1	1	0	1	0	1	1	0	2	7	4	SUI	Subtract from Acc, immediate
1	1	0	1	1	1	1	0	2	7	6-5	SBI	Subtract with borrow from Acc, immediate
0	0	1	0	0	1	1	1	1	4	20-8	DAA	Decimal adjust Acc
									STACK OPERATIONS			
1	1	0	0	0	1	0	1	1	11	6	PUSH B	Push registers B & C on stack
1	1	0	1	0	1	0	1	1	11	6	PUSH D	Push registers D & E on stack
1	1	1	0	0	1	0	1	1	11	6	PUSH H	Push registers H & L on stack
1	1	1	1	0	1	0	1	1	11	8	PUSH PSW	Push Acc end flags on stack
1	1	0	0	0	0	0	1	1	10	8	POP B	Pop registers B & C off stack
1	1	0	1	0	0	0	1	1	10	8	POP D	Pop registers D & E off stack
1	1	1	0	0	0	0	1	1	10	8	POP H	Pop registers H & L off stack
1	1	1	1	0	0	0	1	1	10	8	POP PSW	Pop Acc and flags off stack
									LOGICAL			
1	0	1	0	0	s	s	s	1	4	3	ANA r	AND register with Acc
1	0	1	0	0	1	1	0	1	7	5	ANA m	AND memory with Acc
1	1	1	0	0	1	1	0	2	7	4	ANI	AND with Acc, immediate
1	0	1	0	1	s	s	s	1	4	3	XRA r	Exclusive OR register with Acc
1	0	1	0	1	1	1	0	1	7	5	XRA m	Exclusive OR memory with Acc
1	1	1	0	1	1	1	0	2	7	4	XRI	Exclusive OR with Acc, immediate
1	0	1	1	0	s	s	s	1	4	3	ORA r	Inclusive OR register with Acc
1	0	1	1	0	1	1	0	1	7	5	ORA m	Inclusive OR memory with Acc
1	1	1	1	0	1	1	0	2	7	5	ORI	Inclusive OR with Acc, immediate
1	0	1	1	1	s	s	s	1	4	3	CMP r	Compare register with Acc
1	0	1	1	1	1	1	0	1	7	5	CMP m	Compare memory with Acc
1	1	1	1	1	1	1	0	2	7	4	CPI	Compare with Acc, immediete
0	0	1	0	1	1	1	1	1	4	3	CMA	Complement Acc
0	0	0	0	0	1	1	1	1	4	5-4	RLC	Rotate Acc left
0	0	0	0	1	1	1	1	1	4	6-5	RRC	Rotate Acc right
0	0	0	1	0	1	1	1	1	4	5-4	RAL	Rotate Acc left through carry
0	0	0	1	1	1	1	1	1	4	6-5	RAR	Rotate Acc right through carry
									INCREMENT/DECREMENT			
0	0	d	d	d	1	0	0	1	5	3	INR r	Increment register
0	0	1	1	0	1	0	0	1	10	6	INR m	Increment memory
0	0	0	0	0	0	1	1	1	5	4	INX B	Increment extended B & C
0	0	0	1	0	0	1	1	1	5	4	INX D	Increment extended D & E
0	0	1	0	0	0	1	1	1	5	4	INX H	Increment extended H & L
0	0	1	1	0	0	1	1	1	5	4	INX SP	Increment stack pointer
0	0	d	d	d	1	0	1	1	5	3	DCR r	Decrement register
0	0	1	1	0	1	0	1	1	10	6	DCR m	Decrement memory
0	0	0	0	1	0	1	1	1	5	4	DCX B	Decrement extended B & C
0	0	0	1	1	0	1	1	1	5	4	DCX D	Decrement extended D & E
0	0	1	0	1	0	1	1	1	5	4	DCX H	Decrement extended H & L
0	0	1	1	1	0	1	1	1	5	4	DCX SP	Decrement stack pointer

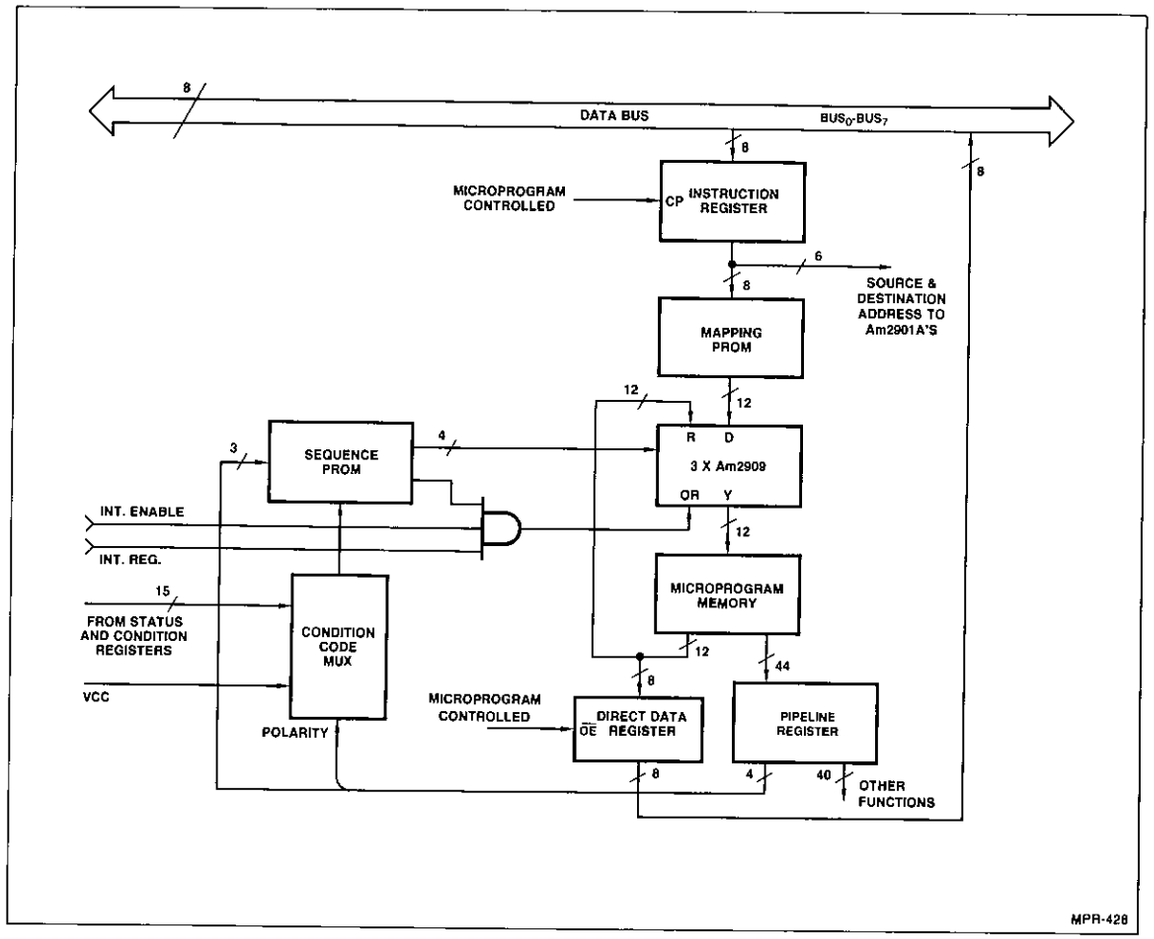


Figure 1. Microprogram Control Unit.

Table 2.

Am9080A Emulation Register Allocations.

Am2901A Register #	High Order Pair	Low Order Pair
0	B	C
1	C	B
2	D	E
3	E	D
4	H	L
5	L	H
6	not used	A
7	A	not used
8	not used	not used
9	Stack Pointer	Stack Pointer
10	scratch pad	scratch pad
11	scratch pad	scratch pad
12	00000000	00111000
13	00111000	00000000
14	not used	not used
15	Program Counter	Program Counter

The microprogram memory is organized as 56 bits wide and 352 words deep. Twelve of the fifty-six bits of the microprogram word are used to supply the address of the next microinstruction if a jump is needed in the microcode. The eight least significant bits of this field can also be applied (under microprogram control) to the Data Bus, thus allowing the Am2901A's to obtain data from the microinstruction. Three bits of the microprogram word are needed for the Sequence PROM and one bit is used to control the Condition Code Multiplexer polarity. The remaining forty bits are used in the ALU and in the other parts of the emulator.

Figure 2 is a block diagram showing the connections for the register, address, and data inputs of the Am2901A's. Data appearing on the Data Bus is latched into the Data Register. Supplying data to each pair of Am2901A's is a Data Multiplexer which can select either the output of the Data Register or, if a byte swap is to be performed, the output of the opposite pair of Am2901A's. The A and B address multiplexers choose between the register addresses supplied by the microinstruction and those extracted from the Am9080A instruction. Within the system, as was earlier described, LSB inversion circuitry allows single-or double-byte addressing on the LOP.

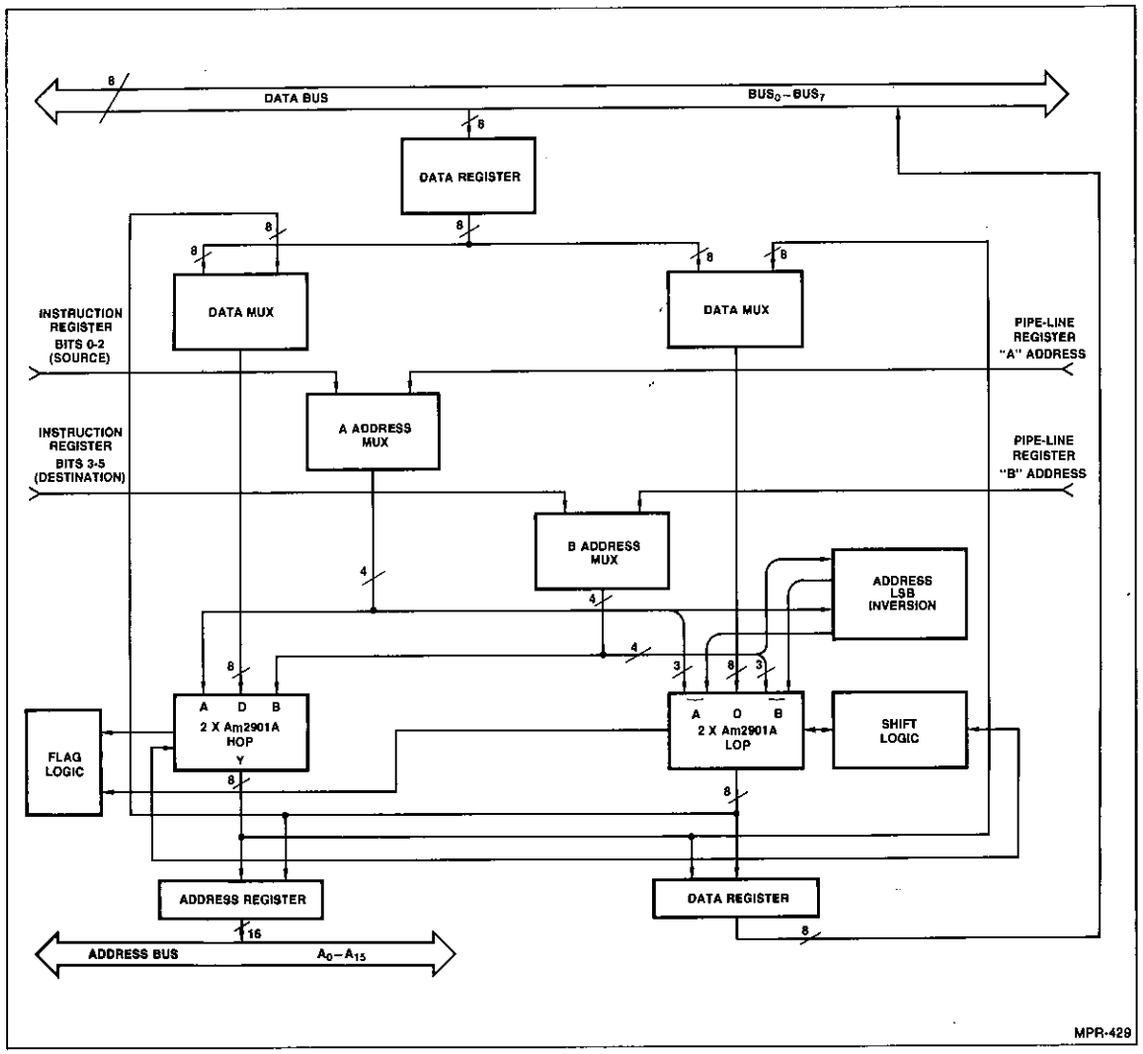


Figure 2. Data Processing Unit.

The outputs of the Am2901A's are latched into one of two pairs of registers that are under microprogram control. These registers serve to transfer the address onto the Address Bus or the data onto the Data Bus. In shifting operations, two multiplexers direct the data to the RAM and Q shift inputs of the Am2901A's. Shifting the CY bit into either end of the 8-bit word is done by hardware. On the other hand, shifting either end of the word into the CY bit, or shifting bit 0 to bit 7, or vice versa, is done by software. This method was used in order to demonstrate microprogramming.

The Am9080A flags are generated and stored using special circuitry. The Zero and Sign Flags pose no problem; the F=0 and F_3 outputs from the Am2901A's can be used directly. The Parity Flag necessitates the addition of a parity generator (an Am82S62). The AC Flag is the C_{n+4} output of the Least Significant Am2901A Slice and can almost be used directly. Generating the CY (Carry) Flag, however, is a different matter. When performing a subtraction, the normal convention dictates that if there is no borrow, the carry will be "1", and if there is a borrow, the carry will be "0". The carry in the Am9080A is the complement of this. Therefore, if one of the two subtract instructions of the Am2901A is executed, both instructions must be decoded, while the C_{n+4} output of the Most Significant Am2901A Slice must be complemented.

Furthermore, the flags in the Am9080A instructions are affected in a different manner. This design of the Am9080A emulation covers the six possible types of instructions as displayed in Table 3, implemented either with hardware or with software. The logical instructions are decoded by gates and a "0" is forced into the CY and AC bits of the Flag Register. Two bits in the microinstruction are used to control the updating ("affected"/"not affected") of the flags: one bit for CY and one for all of the other flags. (Also see Advanced Micro Devices' *8080A/9080A MOS Microprocessor Handbook*, Chapter 3.)

Table 3. Am9080A Flags.

Types of Instructions		CY	AC	Z, S, P
INR, DCR		NA	A	A
DAD, CMC, STC, all Rotates		A	NA	NA
ORA, ORI, XRA, XRI		C	C	A
ANA, ANI		C	ND	A
Other instructions	either	A	A	A
	or	NA	NA	NA

C = Cleared (Reset)
A = Affected
NA = Not Affected
ND = Not Defined

THE HARDWARE

Bus Interface and Interrupt Control

Figure 3 depicts in detail the sequencing portion of the Am9080A emulation. The Data Bus, comprised of BUS_0 through BUS_7, is connected to the inputs of an Am25LS377 eight-bit Register (U1516). Data appearing on the Data Bus is clocked into this register if its $\overline{E}$ input is LOW, which normally happens during an instruction fetch cycle. The instruction thus stored in this register is now available on the output of the register during the entire execution time, both for the Mapping PROM's (three Am29761: U11, U12, U13) and for the Register Address Multiplexers (two Am25LS157: U65, U66 in Figure 4). The Mapping PROM's supply addresses to the D inputs of the three Am2909 sequencers (U21, U22, U23), whose R inputs are connected to the microprogram memory output (bits μ42-μ53). Since the $\overline{RE}$ inputs of the sequencers are always LOW, their internal registers serve as a part of the Pipeline Register. They will contain the next microprogram address if a jump is anticipated. The S_0, S_1, $\overline{PUP}$, and $\overline{FE}$ control inputs of the sequencers are driven by a Sequence PROM, an Am29751 (U14), and their CPU inputs are driven by the system clock. The least significant C_n input is tied to HIGH, while the remaining C_n inputs are driven by the preceding C_{n+4}. The $\overline{ZERO}$ inputs are normally pulled HIGH, but a momentary push-button can force a LOW, thus steering the microprogram to Location 0 for initialization. All of the OR inputs of the sequencers are tied together and to the output of an AND gate (8/U73*). If the interrupt is enabled (INTE=HIGH), and there is an interrupt request (INT=HIGH), and the D_0 output of the Sequence PROM is also HIGH, then the highest available location of the microprogram memory will be addressed where a "Jump-to-Interrupt Handler" instruction is written. Using the O_0 output of the Sequence PROM to gate interrupts provides a simple means to assure that an interrupt can be executed only at macroinstruction boundaries. This is achieved by setting this bit LOW in every microinstruction except in the instruction fetch, where it is HIGH. The Y outputs of the sequencers, always enabled, supply the microcode memory address.

Microprogram Memory

The microprogram memory can be any memory array providing 56-bit wide words 352 words deep. Using seven Am29773 512 x 8 PROM's for this arrangement will provide a typical 45ns access time, although other configurations are possible. Table 4 is a summary of the microcode bit allocation. Some of the microprogram memory data are stored in the Pipeline Registers, made up of seven Am25LS374 eight-bit Registers (U3132, U3241, U4142, U51, U8182, U7172, U5161). The microinstructions are comprised of the outputs of these devices, each output being designated "PL". Five of these registers have their outputs constantly enabled. This allows the microinstruction to be delivered to the various parts of the circuit. Bit PL27 serves a dual purpose in this part of the circuit. Besides controlling the output of U7172, it is also inverted to serve as the HLDA output control line. This latter function allows the floating of all of the control outputs, except HLDA, when a Hold is acknowledged.

* This notation is used in the text to indicate a certain pin of the device being referenced. Hence, in this case, the text is talking about pin 8 of U73.

Table 4. Microcode Bit Allocation Summary.

Bit No.	No. of Bits	Description
0-2	3	ALU Source (I_0-I_2 of the Am2901A's)
3-5	3	ALU Function (I_3-I_5 of the Am2901A's)
6-8	3	ALU Destination (I_6-I_8 of the Am2901A's)
9-12	4	ALU "B" Address
13-16	4	ALU "A" Address
17	1	Single/Double Byte
18	1	C_n for least significant Am2901A slice
19	1	Rotate and Swap Control (formetted)
20-21	2	Update/keep flags
22	1	"A" Address Switch
23-24	2	Am2901A Output Steering Control
25-26	2	Date Bus Enable Control
27-32	6	HLDA, $\overline{MEMW}$, $\overline{MEMR}$, $\overline{I/OW}$, $\overline{I/OR}$, $\overline{INTA}$ Am9080A System Control Outputs
33	1	"B" Address Switch
34-37	4	Condition Code Select
38	1	Condition Code Polarity Control
39-41	3	Next Instruction Select
42-53	12	Numerical Field
54	1	Numerical Field to Deta Bus Control
55	1	Instruction Register Clock Enable

The output of U8182 is connected to the Data Bus and is enabled by PL54 only when the 8 least significant bits of the numerical field (bits μ42 through μ49 in the microinstruction) are needed as data. PL55 is the Clock Enable for the Instruction Register (1/U1516). It also serves as the Clock Enable of the Data Register U12324 (Figure 4) when it is inverted and delayed by one microcycle. This is needed to enable the Data Register to retain its contents while the instruction is fetched. Thus, it provides the possibility of executing the last step of an instruction while fetching the next one. The WAIT control output, obtained on pin 6 of U7172, is generated by an AND gate (6/U83), the mechanism of which will be explained later.

Two Am2922 Multiplexers (U8474, U8475 in Figure 3) are used to select one of sixteen conditions to generate the condition code. Their internal control registers serve as part of the Pipeline Register. Bit PL37 selects one of the two multiplexers, and bits μ34, μ35, and μ36 select one-of-eight on that multiplexer. Bit μ38 is the polarity control of the output. Thus, the output can either be HIGH for TRUE and LOW for FALSE or vice versa. This output is the LSB of the Sequence PROM address. Table 5 summarizes the condition inputs to the Am2922 multiplexers.

An Am29751 PROM (U14) is used as a Sequence PROM. This is a 32-word by 8-bit memory, but a 16-word by 5-bit configuration is all that is needed in this application. Address bit 0 is the condition code coming from the condition code multiplexers, while address bits 1, 2, and 3 come from the Pipeline Register. Data bit 0 serves as an interrupt internal enable and data bits 1, 2, 3, and 4 control the $\overline{FE}$, $\overline{PUP}$, S_1, and S_0 inputs of the Am2909 sequencers, respectively. Table 6 is the program for this PROM.

ALU and Working Registers

Figure 4 depicts the inputs to the four Am2901A four-bit slices. An Am25LS377 Eight-Bit Register (U12324) stores the Data Bus information. It is clocked in on every microcycle, except the one after the instruction fetch cycle. This is very useful for some instructions (e.g., Restart, where the instruction code itself contains data.) Two pairs of multiplexers (four Am25LS157 – U53, U54, U55, U56) select between the data of the Data Bus and the output of the opposite pair of Am2901A's, thus enabling byte-swapping. This selection is controlled by the SWAP signal, generated by another Am25LS157 (9/U64). PL17 is applied to the S input of this multiplexer, which performs the following functions:

PL17	1Y (C_n of HOP)	2Y (ROTATE)	3Y (SWAP)
LOW (single-byte operation)	PL18	PL19	LOW
HIGH (double-byte operation)	C_{n+4} of LOP	LOW	PL19

Table 5. Condition Code Truth Table.

Symbol	PL37	C	B	A*	Condition: Designation	Condition: Description
T0	0	0	0	0	Z	Am9080A Zero flag
T1	0	0	0	1	CY	Am9080A Cerry flag
T2	0	0	1	0	P	Am9080A Parity flag
T3	0	0	1	1	S	Am9080A Sign flag
T4	0	1	0	0	AC	Am9080A Auxiliary Carry flag
T5	0	1	0	1	–	not used
T6	0	1	1	0	–	not used
T7	0	1	1	1	–	not used
T8	1	0	0	0	INT	Am9080A INT request (control input)
T9	1	0	0	1	READY	Am9080A READY request (control input)
T10	1	0	1	0	HOLD	Am9080A HOLD request (control input)
T11	1	0	1	1	–	not used
T12	1	1	0	0	F_3	F_3 of most significant Am2901A slice
T13	1	1	0	1	F=0	F=0 outputs of all Am2901A's OR'ed together
T14	1	1	1	0	C_{n+4}	C_{n+4} from most significant Am2901A slice
T15	1	1	1	1	"1"	Constant HIGH ("unconditional")

*C, B, and A are the internal equivalents of PL36, PL35, and PL34, respectively.

Table 6. Sequence PROM Program.

A_1	A_2	A_3	A_4	O_0	O_1 $\overline{FE}$	O_2 PUP	O_3 S1	O_4 S0	Mnemonic	Am2909 Function (Y=1)
0	0	0	0	0	1	X	0	0	C	Microprogram Counter
0	0	0	1	0	1	X	0	1	R	Register
0	0	1	0	0	1	X	1	1	D	"D" Inputs
0	0	1	1	0	1	X	0	1	R	Register
0	1	0	0	0	1	X	0	0	C	Microprogram Countar
0	1	0	1	0	0	1	0	1	SBR	Register (and PUSH μPCounter)
0	1	1	0	0	1	X	0	1	R	Register
0	1	1	1	0	0	0	1	0	RTN	STK0 (and POP)
1	0	0	0	1	1	X	1	0	F	STK0 (without POP)
1	0	0	1	0	0	1	0	1	SBR	Register (and PUSH μPCounter)
1	0	1	0	0	0	0	0	0	POP	μPCounter (and POP)
1	0	1	1	0	0	0	0	1	PR	Register (and POP)
1	1	0	0	0	1	X	0	1	R	Register
1	1	0	1	0	0	1	0	0	PUSH	μPCounter (and PUSH μPCounter)
1	1	1	0	0	1	X	0	1	R	Register
1	1	1	1	1	1	X	1	0	F	STK0 (without POP)

X = Don't Care C = Continue R = Register D = Direct F = File

By this arrangement, a single Control Bit (PL19) can be used for two different controls (SWAP, ROTATE), according to whether a single-byte or a double-byte operation is performed. The same device can also control the carry input (C_n) of the HOP Am2901A slices. In a double-byte operation, this C_n should be the carry-out (C_{n+4}) of the LOP Am2901A slices. However, in a single-byte operation, it should be determined by a control bit of the microcode (PL18), which is identical to the LOP C_n input. This is a simple example of microcode formatting, as defined in Advanced Micro Devices' *Microprogramming Handbook*.

The four A register (source register) address lines of the Am2901A's are fed from an Am25LS157 Multiplexer (U66). It can select as the source either bits PL13 through PL16 of the microprogram or bits 0 through 2 of the Data Bus, latched in U1516 (Figure 3). In the latter case, the Most Significant Bit is always LOW. Thus, bit PL22 ("A" Switch) controls whether the particular register is addressed by the microcode or the macrocode. The same is true for the B register (or destination register) address lines, this time using another Am25LS157 Multiplexer (U65) and bit PL33 ("B" Switch). Remember that the least significant address bit of both the source and destination register addresses is inverted in a single-byte operation. This is accomplished by an Am25LS153 Multiplexer (U63) and the "single/double" control bit PL17. However, in a double-byte operation, the macroinstruction register pair address is used, and occupies only two bits in the destination field. In that case, the LSB of the B address is set to LOW (13/U63).

Two Am25LS257 Multiplexers (U76, U77) take care of the I/O connections necessary to execute all Rotate instructions. Only one of these two multiplexers is enabled at any time, according to PL7, which is also I_7 of the Am2901A instruction. When shifting, I_7=HIGH will cause an up-shift (rotate left in Am9080A terminology) and I_7=LOW a down-shift (rotate right). The rotate control signal, generated by 7/U64, selects the source of the shifted-in bit.

U77 participates in the rotate right instructions. The 1Y output feeds RAM_3 of the HOP and the 2Y output feeds RAM_3 of the LOP. The shifted-in bit will be either the shifted-out bit of the corresponding pair of slices or the CY Flag. This is determined by whether an "around carry" (ROTATE=LOW) or a "through carry" (ROTATE=HIGH), respectively, is required. The 3Y output of U77 feeds Q_3 of the HOP and the 4Y output feeds Q_3 of the LOP. When ROTATE=LOW, the shifted-out Q_0 bit of each pair will be shifted in, but when ROTATE=HIGH, the shifted-out Q_0 bit of the *opposite* pair will be shifted in, thus performing a double-byte rotate. This capability is one that is not contained in the Am9080A itself, but is a very useful feature in the emulation.

U76 performs a similar function in a rotate left instruction. As can be seen, the shifting-in of the Carry Flag in a "through carry" rotate instruction is implemented in hardware. Every rotate instruction requires that the Carry Flag be set according to the shifted-out bit. Although this could also be accomplished by using hardware, a software method was chosen for demonstrational purposes. The bottom of Figure 5 shows the system outputs, the flag generation, and some control functions.

Instruction Decode

The nine I inputs of each Am2901A slice are fed in parallel by bits PL0 through PL8 of the microprogram word. The Y outputs of the low order microprocessor pair are routed to two Am2920 registers, U9394 (the Data register from the LOP, referred to in the microcode as "DL"), whose outputs are tied to the Data Bus, and U9596 (the Address register from the LOP, referred to as "AL"), whose outputs are tied to the eight least significant bits of the Address Bus. Similarly, the Y outputs of the high order microprocessor pair are routed to two additional Am2920 registers, U10304 (the Data register from the HOP, called "DH" in the microcode) and U10506 (the Address register from the HOP, called "AH"). They are also connected to an Am82S62 (U97), which generates the even Parity Flag.

The Output Enable's of both address registers (U10506 and U9596) are controlled by PL27, which is actually the inverted HLDA. Thus, the Address Bus is placed in the high-impedance state when a hold is acknowledged. The Data Bus has three sources: U9394, U10304, and the Flag registers U101 and U102. One-half of an Am25LS139 Decoder/Demultiplexer (U131), using PL25 and PL26 of the microinstructions as control inputs, assures that no more than one output is enabled at any time. Since the 1Y0 output of this decoder is not connected, none of the register outputs are enabled when both PL25 and PL26 are LOW, and the Data Bus is free for other communication purposes (e.g., main memory read). The other half of the same decoder (U131) directs the Am2901A output to one of three sets of registars by controlling the Register Enable's: the two data registers (U9394, U10304) are controlled by the OCL 1 line, the two address registers (U9596, U10506) by the OCL 2 line, and the INTE register (U111) by the OCL 3 line.

Status Bits

In order to shorten the execution time by interleaving several different operations, it is necessary to delay the updating of the address registers when the READY input is LOW. This is required until the I/O or main memory completes its read or write cycle (and sets READY to HIGH). If the 2Y2 output of U131 is selected (LOW) by PL23 and PL24, and READY (T9) is HIGH, then 13/U113 will go HIGH and the OCL 2 line LOW, thus allowing the address registers to latch their input. However, if at the same time READY is LOW there is an I/O or memory reference operation (PL28 or PL29 or PL30 or PL31 is LOW), then 6/U73 will be LOW. This will bring both 10/U113 and OCL 2 HIGH, thus blocking the data coming in to the address registers.

The INTE (Interrupt Enable) control output of the Am9080A is a single-bit, software-controlled flag. An Am2920 (U111) is used to store its state. The flag is generated by using software to manipulate the MSB's of the HOP Am2901A slices, using the most significant F_3 output. PL23 and PL24, through U131 and the OCL 3 control line, determine whether or not to update this status. The INTE output is also used in the interrupt recognition circuit (9/U73, Fig. 3), as described above.

The five Am9080A flags are stored in two Am2918 registers; U102 stores the CY Flag and U101 the others. The Y Outputs of these registers are connected to the Data Bus and are controlled by DB3, which is generated by U131. This enables PUSH PSW, where the contents of the A register are written into the stack. The input of U101 is fed from an Am25LS157 Multiplexer (U91). The B inputs of this multiplexer come from an additional Am25LS157 (U115). Usually, U115 will pass whatever data is appearing on the Q outputs of U101 to the B inputs of U91. Bit PL20, when HIGH, will pass this data back to U101, thus storing the previous contents of U101. This is the case when the flags are not affected. The 1A input of U91 (the Z flag) comes from the F=0 outputs of the Am2901A slices, all of which are tied together and pulled up to V_{CC} by a 1KΩ resistor. The 2A input of U91 (the P flag) comes from the Am82S62 Parity Generator (U97). The 3A input of U91 (the S flag) comes from the F_3 output of U34. The 4A input of U91 (the AC flag) comes from the C_{n+4} output of U33, through an AND gate. Thus, when PL20 is LOW, the new flags will be latched in U101. This is the case when the flags are affected.

As mentioned earlier, the case for the CY flag is somewhat different. PL3, PL4, and PL5 are the I_3, I_4, and I_5 instruction bits of the Am2901A slices, respectively. The logic operations are decoded from these bits to cause 1/U113 to go LOW. This resets, using two AND gates, both the CY flag coming from the C_{n+4} output of U34 (which is applied to the 1C0 input of an Am25LS153 Multiplexer – U92) and the AC flag (as mentioned before). The two subtract operations are also decoded and they control the A control input of U92, selecting the complement of the carry as the CY flag, through the 1C1 input. If PL21 is HIGH, the previous carry is directed back to U102 through U125.

Both U125 and U115 (two Am25LS157 multiplexers) effect the POP PSW instruction. Here, the flags are read from the main memory and are stored in the flag registers U101 and U102. The NAND function of the Am2901A's (when I_{345} = 101) is used for this purpose and this purpose only. This function is decoded and applied to the S input of both multiplexers (1/U115 and 1/U125), thus moving the flags latched in the data register U12324 (Figure 3) to their proper registers (U101, U102). This is an example of microprogram formatting, since the I_{345} bits are used for two different purposes.

The Q outputs of the Flag Registers are also applied to the appropriate inputs of the condition code multiplexers U8474 and U8475 (Figure 3 and Table 5).

An Am25LS374 register (U121) is used to latch both the incoming INT, READY, and HOLD control signals and the C_{n+4}, F=0, and F_3 micro-flags, and then to apply them to the condition code multiplexers.

Timing Considerations

Two worst case speed paths are calculated in Table 7: The Control Path and the Data Path. In each case, the maximum and typical values are shown using Am25LS commercial parts (V_{CC} = 5V, T_A = 25°C). Two additional columns are shown: maximum and typical values if "S" type parts are used, when applicable. The calculation point is from the LOW-to-HIGH transition of the clock pulse until the next LOW-to-HIGH transition of the clock (in ns).

Table 7. Speed Path Calculations.

Designation	Device Type	Path	Am25LS Typ.	Am25LS Max.	"S" Typ.	"S" Max.	Remarks
		Control Path					
U5161, U121	Am25LS374	CP → Y	22	37	11.5	17	Note 1
U62	74LS04	In → Out	10	15	not needed		
U8475	Am2922	$\overline{OE}$ → Y	10	17	10	17	
U14	Am29751	A → O	32	50	32	50	
U21	Am2909	S_0S_1 → C_{n+4}	50	50	50	50	
U22	Am2909	C_n → C_{n+4}	18	18	18	18	
Microprogram Memory	Am29773	A → D	35	50	35	50	
Pipeline Register	Am25LS374	Set-up	20	20	5	5	
		Total	197	257	161.5	207	
		Data Path					
Pipeline Register	Am25LS374	CP → Y	22	37	11.5	17	
U65, U66	Am25LS157	S → Y	15	23	12	18	Note 2
U62	74LS04	In → Out	10	15	3	5	
U63	Am25LS153	C → Y	10	18	not needed		
U43	Am2901A	A, B → C_{n+4}	75	75	75	75	
U44	Am2901A	C_n → C_{n+4}	20	20	20	20	
U64	Am25LS157	A → Y	8	12	4.5	6.5	
U33	Am2901A	C_n → C_{n+4}	20	20	20	20	
U34	Am2901A	C_n → F=0	50	50	50	50	
U91	Am25LS157	A → Y	8	12	4.5	6.5	
U101	Am2918	Set-up	5	5	5	5	
		Total	242	285	205.5	223	

Notes: 1. This path was calculated using an Am74S175 to drive the $\overline{OE}$'s of U8474 and U8475 from its Q and $\overline{Q}$ outputs simultaneously.
2. This path was calculated using an Am74S151 to multiplex the LSB of the A, B addresses.

THE MICROCODE

The 56-bit wide microprogram was written and processed using AMDASM™/TS. AMDASM™ is very versatile and different approaches are possible. One way would be such that only variables are substituted in the Assembly Phase. The disadvantage of this scheme is that the variables must be entered in a predefined order. If there are a great number of variables (as would be expected in a 56-bit wide word), the counting of the commas, which are the variable delimiters, is a tedious task. This can lead to a wider margin for human error. The other extreme is to use only definitions and no variables. The disadvantage of this choice is that there can be no default values. Since a large percentage of the number of microinstruction bits have control functions, the bits cannot be left at an arbitrary value. Thus, in the Assembly Phase, all microprogram bits must be explicitly accounted for.

Consequently, a scheme that is midway between the two extremes was adopted. Several fields are defined in the Definition Phase, each having a small number of variables, i.e., anywhere from two to four. These fields group the functional bits together, and each control bit has its default value such that when it is not defined in the Assembly Phase, the default value will be selected so as *not* to bring about an undesired function which can harm the program, the contents of the registers, or the hardware.

Appendix I is the print-out of the microprogram Definition File.* The listing, containing many comments, is self-explanatory. Note that for the Control Bus bits, an all-definitions policy was adopted, thus assuring that no mutually exclusive bits can be active. At the end of the file there are some "wide-field" definitions. These join together, with variables, some commonly-used definitions, allowing shorter assembly statements.

A print-out of the assembly statements and the object code in an interleaved format is given in Appendix II. One can follow the various microinstructions either by studying the definitions and using the mnemonics, or by using the Bit Definition Table (Table 4) and the bit pattern of the object code. Some of the features of the microprogram will be described here.

Locations 000 to 003: This is the Initialization Phase. It zeroes the PC, disables the interrupt, and loads constants into Registers 12 and 13. Executing microinstruction 3 causes the microprogram counter to be pushed onto the Am2909 sequencers' stacks. The top of the stack then contains 004_{16}, which is the Instruction Fetch routine location. Coming back to this routine is accomplished by referring to this stack (and without the use of the POP).

Locations 004 to 005: This is the Instruction Fetch. Two steps are necessary. One is to latch the incoming instruction into the Instruction Register U12324 (Figure 4). This supplies the address for the Mapping PROM. The other step is to apply the data in the Mapping PROM to the address lines of the microprogram memory and latch this data in the Pipeline Register.

Locations 00A to 013: These subroutines and handlers deal with the WAIT and HOLD states. Although it is easy to implement the WAIT state and HOLD state simulation by simply stopping the clock until READY goes HIGH or HOLD goes LOW, a software-oriented method was adopted here to demonstrate microcode subroutining.

In every memory or I/O reference microinstruction, the status of the READY line is tested. If it is HIGH, the microprogram will continue. However, if it is LOW, the program must wait, keeping the Address Bus stable, until the READY line goes HIGH. This is easy to implement. For example, at Location 004, the microprogram will repeat itself (NUM, $) until it senses a HIGH at the READY line (IF CR, . . . by default). Since the PC should point to the *next* address, the majority of the memory read or write microinstructions increment the PC, and at the *same* microinstruction. Repeating the same instruction several times would cause the PC to continue endlessly. To prevent this, the MMR and MMW "wide-field" definitions cause the microcode to subroutine to MMRSB (Location 00D) or MMWSB (Location 00E). Here, the updated internal PC is repeatedly fetched from its registers until READY goes HIGH. This enables the Address Register to clock in the new PC and return from the subroutine. If the last step in executing a macroinstruction is a memory read or write, there will be no subroutine call for READY=LOW. Instead, a jump will be performed to MMRF (Location 010) or MMWF (Location 00F), which ends with a file reference (as opposed to a return-from-subroutine). This will direct the microprogram to the Instruction Fetch routine (Location 004). Three more microinstructions handle the cases where the Stack Pointer controls the System Address Bus (Locations 012 and 013).

The HOLD line is checked in every microinstruction where a DMA cannot disturb the normal operation of the program (and where, of course, the condition code multiplexers are not occupied by other tests). This treatment differs from that for the WAIT state in one respect. All of the data, address, and control lines, with the exception of the HLDA control output, are put in the high impedance state. Location A serves as the subroutine and Location B ends with a file reference, but Location 00C can be accessed only by Location 005.

Locations 014 through 15F contain the microprogram necessary to execute all of the Am9080A instructions. They can easily be recognized since mnemonics similar to theirs were used at the starting addresses. At the highest location of the microprogram memory (in this case, 3FF), a jump is written. This directs the program to the interrupt handler (Location 084), which is a part of the HLT instruction.

The software for the Mapping PROM was written using the free form capability of AMDASM™. After the microprogram was assembled, the entry addresses for the macroinstructions were extracted by printing out the cross-reference table. Since AMDASM™ provides consecutive addresses when assembling (which are identified by the "PC" values), the printout of the Am9080A instruction code values was easily generated. First of all, a simple definition file was written (Figure 6). It stated only that the Mapping PROM word width was twelve, equal to the maximum address width of the Microprogram Memory. Then, an Am9080A instruction list, in which the instructions were listed in the ascending order of the values of their opcode (starting with NOP and ending with RST 7) was used to create an Assembly File, using free-format statements. For each statement, the corresponding microprogram entry address was entered. Wherever an undefined Am9080A instruction was encountered, a DON'T CARE (12X) word was written.**

*The three computer print-outs that are in Appendices I through III are also available via timesharing from Computer Sciences Corporation. At the beginning of each Appendix is given the file name for that respective print-out.

**Appendix III contains a print-out of the Mapping PROM assembly statements. However, the "holes" in the instruction set can still be of service in that the user may here insert his own instructions, e.g., to effect the microprogramming of multiplication and division.


```
!
TITLE 8080EMULATOR MAPPING PROM DEFINITIONS
WORD 12
END
^
```

Figure 6. Mapping PROM Definition File.

Since all register reference instructions use the same microcode, the AMDASM™ DUP statement was used to shorten the file. After the assembly is run, the output (Appendix III) lists the Am9080A instruction code as address and the appropriate microprogram entry point as data. This data can be further processed by AMPROM™ to punch a paper tape to program the Am29761 PROM's.

SUMMARY

The particular design used in this application note is by no means a unique method to emulate the Am9080A/Am8228 chip set. It is intended to serve as an example of such an emulation and, in doing so, to bring about some reduction in execution times. Variations are possible, both to alter the number of devices used and to reduce the number of cycles necessary to execute the instructions. Also, additional performance improvement can be achieved by adding architectural enhancements, such as overlap fetch of the next machine instruction.

PARTS LIST

Device	Description	Qty.
Am2901A	Four-Bit Bipolar Microprocessor Slice	4
Am2909	Microprogram Sequencer	3
Am2918	Four-Bit Register with Standard and Three-State Outputs	2
Am2920	Eight-Bit Register with Three-State Outputs and Clear and Enable	5
Am2922	Eight-Input Multiplexer with Control Storage	2
Am29751	32 x 8 PROM with Three-State Outputs	1
Am29761	256 x 4 PROM with Three-State Outputs	3
Am29773	512 x 8 PROM with Three-State Outputs	7
Am25LS139	Dual One-of-Four Decoder/Demultiplexer	1
Am25LS153	Dual Four-Input Multiplexer	2
Am25LS157	Quad Two-Input Multiplexer, Non-Inverting	10
Am25LS257	Quad Two-Input Multiplexer with Three-State Outputs, Non-Inverting	2
Am25LS374	Octal D-Register with Three-State Outputs	8
Am25LS377	Octal D-Register with Common Enable	2
Am82S62	Schottky Nine-Input Parity Checker/Generator	1
74S08	Quad Two-Input AND Gate	2
74LS02	Quad Two-Input NOR Gate	1
74LS04	Hex Inverter	2
74LS21	Dual Four-Input AND Gate	1
	Total	59

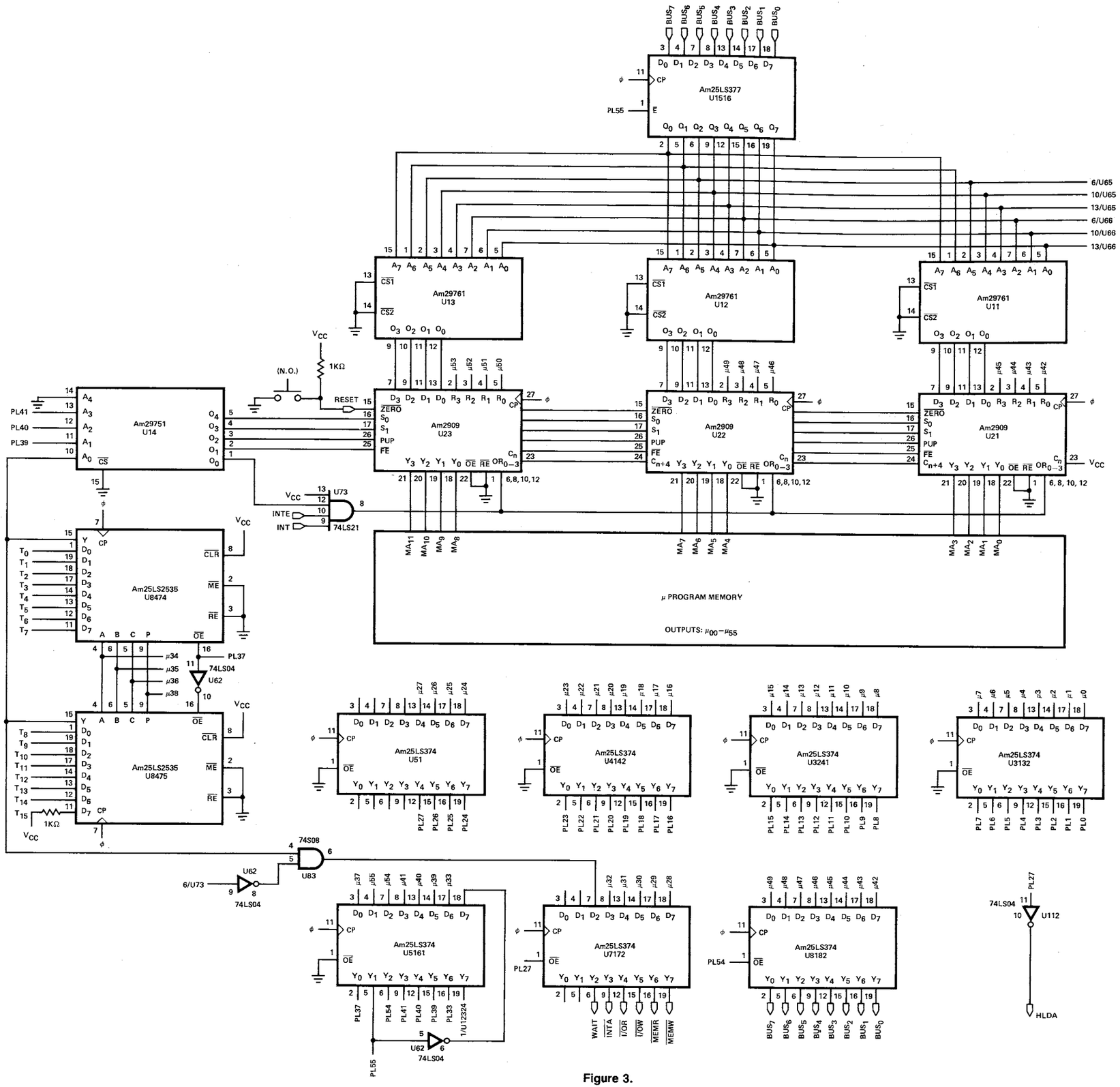

Am25LS377
U1516
Am29761
U13
Am29761
U12
Am29761
U11
Am29751
U14
Am2909
U23
Am2909
U22
Am2909
U21
Am25LS2535
U8474
Am25LS2535
U8475
Am25LS374
U51
Am25LS374
U4142
Am25LS374
U3241
Am25LS374
U3132
Am25LS374
U5161
Am25LS374
U7172
Am25LS374
U8182
74LS21
74LS04
74LS08
μ PROGRAM MEMORY
OUTPUTS: μ00−μ55
6/U65
10/U65
13/U65
6/U66
10/U66
13/U66
HLDA

Figure 3.

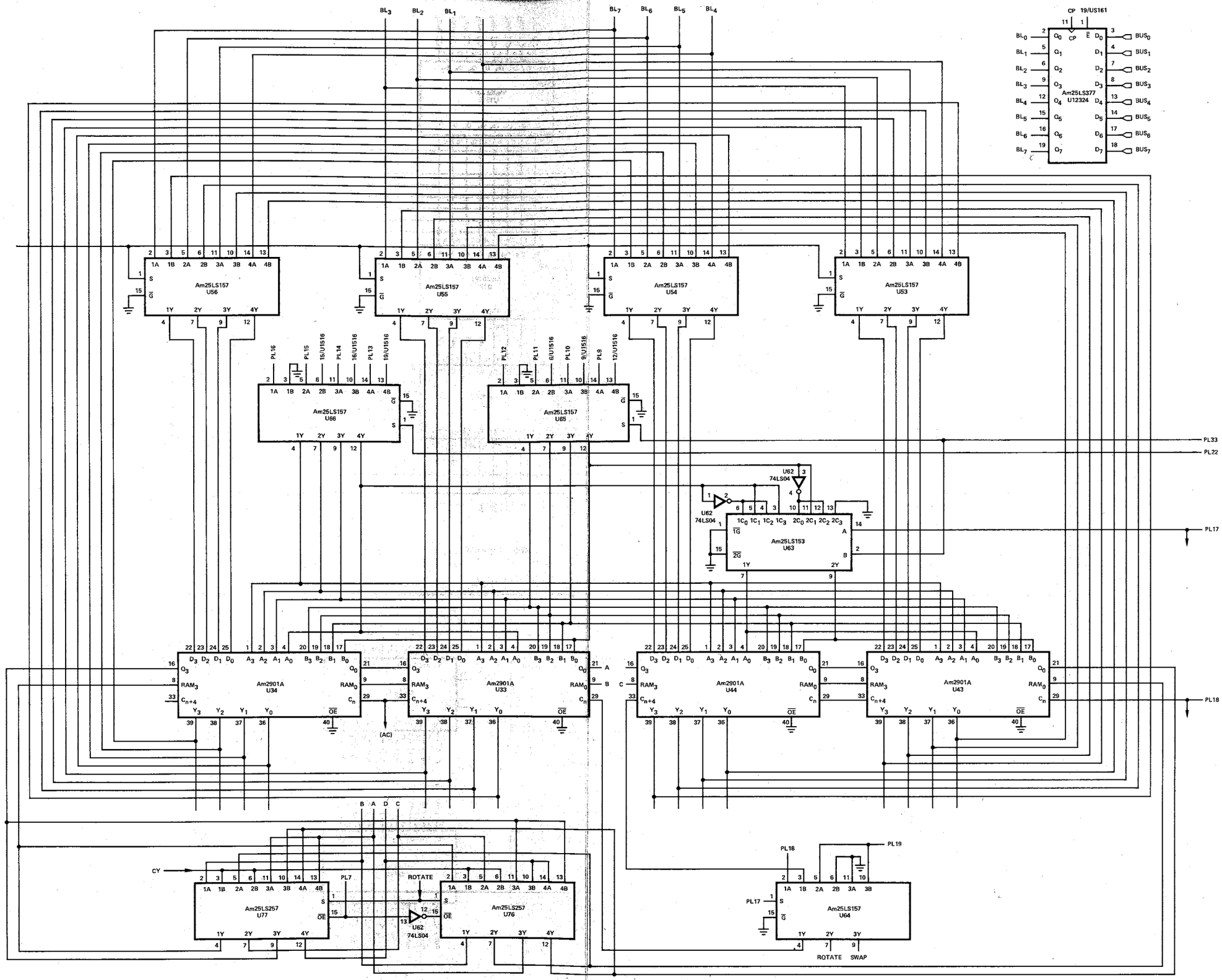

CP 19/US161
Am25LS377
U12324
Am25LS157
U56
U55
U54
U53
U66
U65
U62
74LS04
Am25LS153
U63
Am2901A
U34
U33
U44
U43
Am25LS257
U77
U76
Am25LS157
U64
PL33
PL22
PL17
PL18
PL19
(AC)
CY
ROTATE
SWAP

Figure 4.

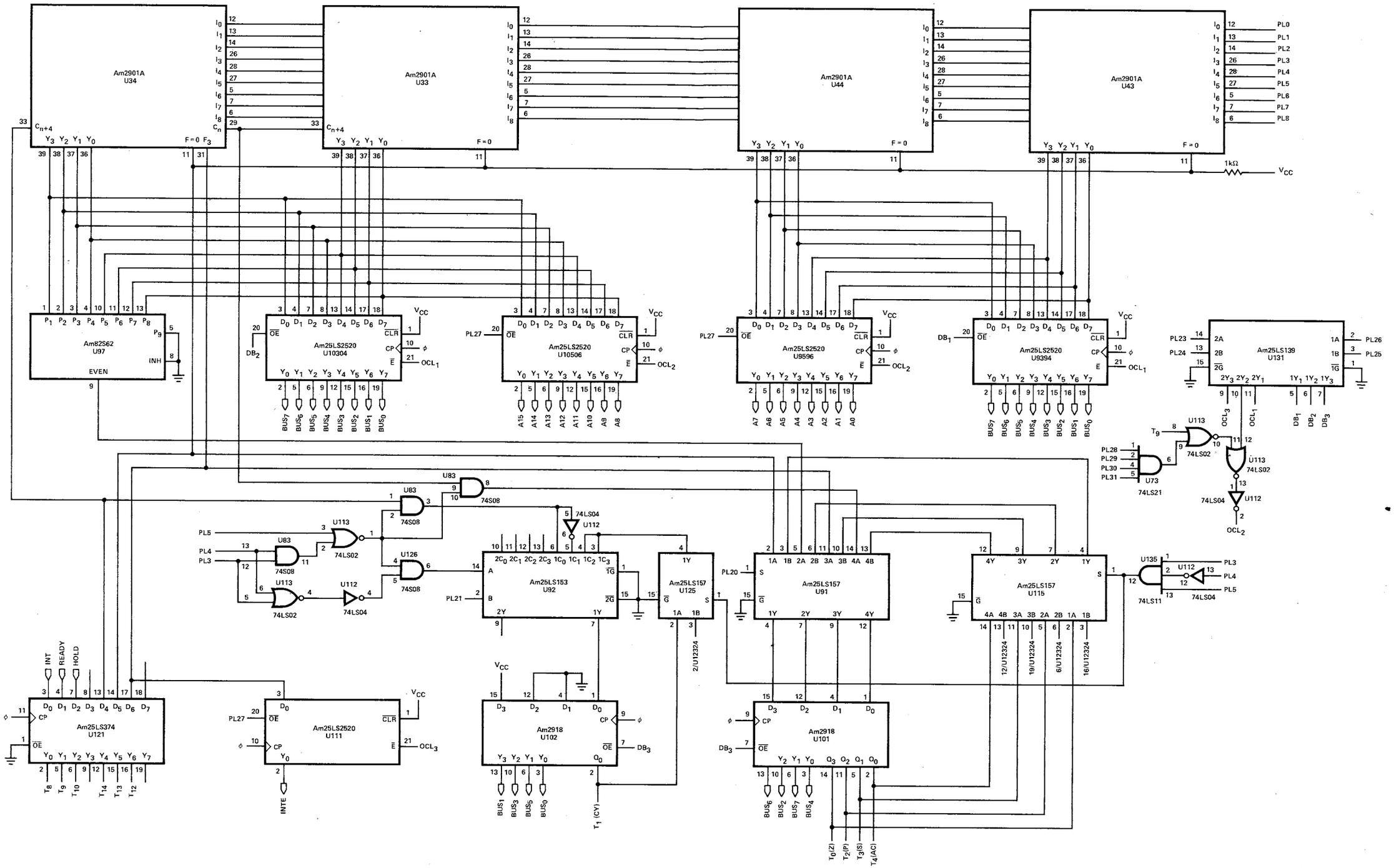


Figure 5.

MPR-454

APPENDIX III

Mapping PROM AMDASM Output
CSC File Name: AMMAPO

Note: A listing of the Mapping PROM Source Code (Assembly File) only may be obtained using CSC File Name AMMAPA.

```
 PC   MICROWORD IN HEX                    SOURCE CODE
0000  086          NOP:      FF H#086
0001  022          LXIB:     FF H#022
0002  0DF          STAXB:    FF H#0DF
0003  06D          INXB:     FF H#06D
0004  0AB          INRB:     FF H#0AB
0005  0AA          DCRB:     FF H#0AA
0006  01B          MVIB:     FF H#01B
0007  05A          RLC:      FF H#05A
0008  000                    FF 12X
0009  071          DADB:     FF H#071
000A  0DC          LDAXB:    FF H#0DC
000B  06F          DCXB:     FF H#06F
000C  0AB          INRC:     FF H#0AB
000D  0AA          DCRC:     FF H#0AA
000E  01B          MVIC:     FF H#01B
000F  05D          RRC:      FF H#05D
0010  000                    FF 12X
0011  0E5          LXID:     FF H#0E5
0012  0DF          STAXD:    FF H#0DF
0013  0F1          INXD:     FF H#0F1
0014  0AB          INRD:     FF H#0AB
0015  0AA          DCRD:     FF H#0AA
0016  01B          MVID:     FF H#01B
0017  05F          RAL:      FF H#05F
0018  000                    FF 12X
0019  073          DADD:     FF H#073
001A  156          LDAXD:    FF H#156
001B  0F6          DCXD:     FF H#0F6
001C  0AB          INRE:     FF H#0AB
001D  0AA          DCRE:     FF H#0AA
001E  01B          MVIE:     FF H#01B
001F  060          RAR:      FF H#060
0020  000                    FF 12X
0021  0E9          LXIH:     FF H#0E9
0022  0D3          SHLD:     FF H#0D3
0023  0F3          INXH:     FF H#0F3
0024  0AB          INRH:     FF H#0AB
0025  0AA          DCRH:     FF H#0AA
0026  01B          MVIH:     FF H#01B
0027  13F          DAA:      FF H#13F
0028  000                    FF 12X
0029  074          DADH:     FF H#074
002A  0C9          LHLD:     FF H#0C9
002B  0F8          DCXH:     FF H#0F8
002C  0AB          INRL:     FF H#0AB
002D  0AA          DCRL:     FF H#0AA
002E  01B          MVIL:     FF H#01B
002F  0C8          CMA:      FF H#0C8
0030  000                    FF 12X
0031  0ED          LXISP:    FF H#0ED
0032  02D          STA:      FF H#02D
0033  0F5          INXSP:    FF H#0F5
0034  0A6          INRM:     FF H#0A6
0035  0A2          DCRM:     FF H#0A2
0036  01E          MVIM:     FF H#01E
0037  09C          STC:      FF H#09C
C038  000                    FF 12X
0039  075          DADSP:    FF H#075
003A  026          LDA:      FF H#026
003B  0FA          DCXSP:    FF H#0FA
003C  0AB          INRA:     FF H#0AB
003D  0AA          DCRA:     FF H#0AA
003E  01B          MVIA:     FF H#01B
```



```
 PC   MICROWORD IN HEX                    SOURCE CODE
003F  09D          CMC:      FF H#09D
0040  014          MOVB.R:   FF H#014
0041  014          MOVB.R:   FF H#014
0042  014          MOVB.R:   FF H#014
0043  014          MOVB.R:   FF H#014
0044  014          MOVB.R:   FF H#014
0045  014          MOVB.R:   FF H#014
0046  018          MOVB.M:   FF H#018
0047  014          MOVB.A:   FF H#014
0048  014          MOVC.R:   FF H#014
0049  014          MOVC.R:   FF H#014
004A  014          MOVC.R:   FF H#014
004B  014          MOVC.R:   FF H#014
004C  014          MOVC.R:   FF H#014
004D  014          MOVC.R:   FF H#014
004E  018          MOVC.M:   FF H#018
004F  014          MOVX.R:   FF H#014
0050  014          MOVX.R:   FF H#014
0051  014          MOVX.R:   FF H#014
0052  014          MOVX.R:   FF H#014
0053  014          MOVX.R:   FF H#014
0054  014          MOVX.R:   FF H#014
0055  014          MOVX.R:   FF H#014
0056  018          MOVD.M:   FF H#018
0057  014          MOVY.R:   FF H#014
0058  014          MOVY.R:   FF H#014
0059  014          MOVY.R:   FF H#014
005A  014          MOVY.R:   FF H#014
005B  014          MOVY.R:   FF H#014
005C  014          MOVY.R:   FF H#014
005D  014          MOVY.R:   FF H#014
005E  018          MOVE.M:   FF H#018
005F  014          MOVZ.R:   FF H#014
0060  014          MOVZ.R:   FF H#014
0061  014          MOVZ.R:   FF H#014
0062  014          MOVZ.R:   FF H#014
0063  014          MOVZ.R:   FF H#014
0064  014          MOVZ.R:   FF H#014
0065  014          MOVZ.R:   FF H#014
0066  018          MOVH.M:   FF H#018
0067  014          MOVW.R:   FF H#014
0068  014          MOVW.R:   FF H#014
0069  014          MOVW.R:   FF H#014
006A  014          MOVW.R:   FF H#014
006B  014          MOVW.R:   FF H#014
006C  014          MOVW.R:   FF H#014
006D  014          MOVW.R:   FF H#014
006E  018          MOVL.M:   FF H#018
006F  014          MOVL.A:   FF H#014
0070  015          MOVM.R:   FF H#015
0071  015          MOVM.R:   FF H#015
0072  015          MOVM.R:   FF H#015
0073  015          MOVM.R:   FF H#015
0074  015          MOVM.R:   FF H#015
0075  015          MOVM.R:   FF H#015
0076  082          HLT:      FF H#082
0077  015          MOVM.A:   FF H#015
0078  014          MOVA.R:   FF H#014
0079  014          MOVA.R:   FF H#014
007A  014          MOVA.R:   FF H#014
007B  014          MOVA.R:   FF H#014
007C  014          MOVA.R:   FF H#014
```

```
 PC   MICROWORD IN HEX                    SOURCE CODE
007D  014                 MOVA.R:   FF H#014
007E  018                 MOVA.M:   FF H#018
007F  014                 MOVA.A:   FF H#014
0080  034                 ADDR:     FF H#034
0081  034                 ADDR:     FF H#034
0082  034                 ADDR:     FF H#034
0083  034                 ADDR:     FF H#034
0084  034                 ADDR:     FF H#034
0085  034                 ADDR:     FF H#034
0086  035                 ADDM:     FF H#035
0087  034                 ADDA:     FF H#034
0088  03A                 ADCR:     FF H#03A
0089  03A                 ADCR:     FF H#03A
008A  03A                 ADCR:     FF H#03A
008B  03A                 ADCR:     FF H#03A
008C  03A                 ADCR:     FF H#03A
008D  03A                 ADCR:     FF H#03A
008E  03C                 ADCM:     FF H#03C
008F  03A                 ADCA:     FF H#03A
0090  0AC                 SUBR:     FF H#0AC
0091  0AC                 SUBR:     FF H#0AC
0092  0AC                 SUBR:     FF H#0AC
0093  0AC                 SUBR:     FF H#0AC
0094  0AC                 SUBR:     FF H#0AC
0095  0AC                 SUBR:     FF H#0AC
0096  0AD                 SUBM:     FF H#0AD
0097  0AC                 SUBA:     FF H#0AC
0098  0B2                 SBBR:     FF H#0B2
0099  0B2                 SBBR:     FF H#0B2
009A  0B2                 SBBR:     FF H#0B2
009B  0B2                 SBBR:     FF H#0B2
009C  0B2                 SBBR:     FF H#0B2
009D  0B2                 SBBR:     FF H#0B2
009E  0B4                 SBBM:     FF H#0B4
009F  0B2                 SBBA:     FF H#0B2
00A0  09F                 ANAR:     FF H#09F
00A1  09F                 ANAR:     FF H#09F
00A2  09F                 ANAR:     FF H#09F
00A3  09F                 ANAR:     FF H#09F
00A4  09F                 ANAR:     FF H#09F
00A5  09F                 ANAR:     FF H#09F
00A6  0B9                 ANAM:     FF H#0B9
00A7  09F                 ANAA:     FF H#09F
00A8  0A0                 XRAR:     FF H#0A0
00A9  0A0                 XRAR:     FF H#0A0
00AA  0A0                 XRAR:     FF H#0A0
00AB  0A0                 XRAR:     FF H#0A0
00AC  0A0                 XRAR:     FF H#0A0
00AD  0A0                 XRAR:     FF H#0A0
00AE  0BC                 XRAM:     FF H#0BC
00AF  0A0                 XRAA:     FF H#0A0
00B0  0A1                 ORAR:     FF H#0A1
00B1  0A1                 ORAR:     FF H#0A1
00B2  0A1                 ORAR:     FF H#0A1
00B3  0A1                 ORAR:     FF H#0A1
00B4  0A1                 ORAR:     FF H#0A1
00B5  0A1                 ORAR:     FF H#0A1
00B6  0BF                 ORAM:     FF H#0BF
00B7  0A1                 ORAA:     FF H#0A1
00B8  076                 CMPR:     FF H#076
00B9  076                 CMPR:     FF H#076
00BA  076                 CMPR:     FF H#076
```



```
 PC   MICROWORD IN HEX                    SOURCE CODE
00BB  076                 CMPR:     FF H#076
00BC  076                 CMPR:     FF H#076
00BD  076                 CMPR:     FF H#076
00BE  079                 CMPM:     FF H#079
00BF  076                 CMPA:     FF H#076
00C0  10D                 RNZ:      FF H#10D
00C1  07C                 POPB:     FF H#07C
00C2  107                 JNZ:      FF H#107
00C3  042                 JMP:      FF H#042
00C4  10A                 CNZ:      FF H#10A
00C5  064                 PUSHB:    FF H#064
00C6  038                 ADI:      FF H#038
00C7  055                 RST0:     FF H#055
00C8  114                 RZ:       FF H#114
00C9  050                 RET:      FF H#050
00CA  10E                 JZ:       FF H#10E
00CB  000                           FF 12X
00CC  111                 CZ:       FF H#111
00CD  047                 CALL:     FF H#047
00CE  03F                 ACI:      FF H#03F
00CF  055                 RST1:     FF H#055
00D0  11B                 RNC:      FF H#11B
00D1  0FB                 POPD:     FF H#0FB
00D2  115                 JNC:      FF H#115
00D3  08B                 OUT.:     FF H#08B
00D4  118                 CNC:      FF H#118
00D5  064                 PUSHD:    FF H#064
00D6  0B0                 SUI:      FF H#0B0
00D7  055                 RST2:     FF H#055
00D8  122                 RC:       FF H#122
00D9  000                           FF 12X
00DA  11C                 JC:       FF H#11C
00DB  087                 IN.:      FF H#087
00DC  11F                 CC:       FF H#11F
00DD  000                           FF 12X
00DE  0B7                 SBI:      FF H#0B7
00DF  055                 RST3:     FF H#055
00E0  129                 RPO:      FF H#129
00E1  101                 POPH:     FF H#101
00E2  123                 JPO:      FF H#123
00E3  092                 XTHL:     FF H#092
00E4  126                 CPO:      FF H#126
00E5  064                 PUSHH:    FF H#064
00E6  0C2                 ANI:      FF H#0C2
00E7  055                 RST4:     FF H#055
00E8  130                 RPE:      FF H#130
00E9  09B                 PCHL:     FF H#09B
00EA  12A                 JPE:      FF H#12A
00EB  159                 XCHG:     FF H#159
00EC  12D                 CPE:      FF H#12D
00ED  000                           FF 12X
00EE  0C4                 XRI:      FF H#0C4
00EF  055                 RST5:     FF H#055
00F0  137                 RP:       FF H#137
00F1  150                 POPPSW:   FF H#150
00F2  131                 JP:       FF H#131
00F3  090                 DI:       FF H#090
00F4  134                 CP:       FF H#134
00F5  068                 PUSHPSW:  FF H#068
00F6  0C6                 ORI:      FF H#0C6
00F7  055                 RST6:     FF H#055
00F8  13E                 RM:       FF H#13E
```

```
 PC   MICROWORD IN HEX                    SOURCE CODE
00F9  091                  SPHL:    FF H#091
00FA  138                  JM:      FF H#138
00FB  08F                  EI:      FF H#08F
00FC  13B                  CM:      FF H#13B
00FD  000                           FF 12X
00FE  077                  CPI:     FF H#077
00FF  055                  RST7:    FF H#055
                           END
```

APPENDIX I

Microcode Definition File
CSC File Name: AMDEF


```
TITLE 8080 EMULATOR DEFINITIONS (REV C, 10-20-76)
WORD 56
[
 ;ALU RELATED FIELDS:
;
;ALU: DEF SINGLE/DOUBLE, A ADDR, B ADDR, DEST
;
ALU:       DEF 38X, 1VB#0, 4VH#A, 4VH#A, 3VQ#1, 6X
;
;DEFAULTS ARE:
;SINGLE, REG9,REG9,NOLOAD
;
;VARIABLES TO BE USED IN THE ABOVE FIELDS:
;
;FIRST VARIABLE:
DOUBLE:     EQU B#1
;
;SECOND AND THIRD VARIABLES:
A:          EQU H#7
B:          EQU H#0
C:          EQU H#1
D:          EQU H#2
E:          EQU H#3
H:          EQU H#4
L:          EQU H#5
SP:         EQU H#8
PC:         EQU H#F
;
;FOURTH VARIABLE:
FTOQ:       EQU Q#0
NOLOAD:     EQU Q#1
FTOB.A:     EQU Q#2
FTOB.F:     EQU Q#3
DROT.FQ:    EQU Q#4
DROT.F:     EQU Q#5
UROT.FQ:    EQU Q#6
UROT.F:     EQU Q#7
;
;ALU FUNCTION FIELD DEFINITIONS
PLUS:     DEF 50X, Q#0, 3X
SUNIM:    DEF 50X, Q#1, 3X
MINUS:    DEF 50X, Q#2, 3X
OR:       DEF 50X, Q#3, 3X
AND:      DEF 50X, Q#4, 3X
NAND:     DEF 50X, Q#5, 3X
XOR:      DEF 50X, Q#6, 3X
NXOR:     DEF 50X, Q#7, 3X
;
;ALU SOURCE FIELD DEFINITIONS:
AQ:       DEF 53X, Q#0
AB:       DEF 53X, Q#1
ZQ:       DEF 53X, Q#2
ZB:       DEF 53X, Q#3
ZA:       DEF 53X, Q#4
DA:       DEF 53X, Q#5
DQ:       DEF 53X, Q#6
DZ:       DEF 53X, Q#7
;
;ALU RELATED CONTROL FIELDS
;
;ALUC DEF AB ADDRESS, FLAGS UPDATE/KEEP, SWAP(ROT), CN
;
;
```

```
ALUC:      DEF 34X, 2VB#11, 1VB#0, 1VB#1, 18X
;
;DEFAULTS ARE: KEEP,KEEP,NOSWAP,HIGH
;
;VARIABLES TO BE USED
UPDTCY:    EQU B#01
UPDTFL:    EQU B#10
UPDTALL:   EQU B#00
SWAP:      EQU B#1
CNL:       EQU B#0
;
;N O T E: WHEN SINGLE IS DEFINED IN ALU'S FIRST VARIABLE
;         'SWAP' WILL CAUSE ROTATION WITH CY SHIFTED IN;
;         AND NOSWAP WILL CAUSE WRAP-AROUND ROTATION!
;
;A AND B ADDRESS SWITCH:
BASW:      DEF 22X, 1VB#0, 10X, 1VB#0, 22X
SW:        EQU B#1
;
;
;I/O CONTROLS:
;
;IOC: DEF DATA IN LATCH,  DATA-BUS, 2901 OUTPUT LATCH
;
IOC:       DEF 1VB#1, 28X, 2VB#00, 2VB#00, 23X
;
;VARIABLE FO THE FIRST FIELD♦
IN:      EQU B#0
;
;VARIABLES FOR THE SECOND FIELD:
DL:        EQU B#01
DH:        EQU B#10
FLAGS:     EQU B#11
;
;VARIABLES FOR THE THIRD FIELD:
TD.D:      EQU B#01
TD.A:      EQU B#10
TD.INTE:   EQU B#11
;
;CONTROL-BUS FIELD DEFINITIONS
;
NOC:       DEF 23X, B#111110, 27X
MEMW:      DEF 23X, B#111100, 27X
MEMR:      DEF 23X, B#111010, 27X
IOW:       DEF 23X, B#110110, 27X
IOR:       DEF 23X, B#101110, 27X
INTA:      DEF 23X, B#011110, 27X
HLDA:      DEF 23X, B#111111, 27X
;
;TEST SELECT, POLARITY AND NEXT INSTRUCTION FIELDS:
;
;IF DEF NEXT INSTR., POL, TEST
IF:        DEF 14X, 3VQ#0, 1VB#1, 4VH#F,34X
;
;NEXT INSTRUCTION VARIABLES:
;
;(FALSE.TRUE)
C.R:       EQU Q#0
D.R:       EQU Q#1
C.SBR:     EQU Q#2
R.RTN:     EQU Q#3
F.SBR:     EQU Q#4
POP.PR:    EQU Q#5
```



```
R.PUSH:    EQU Q#6
R.F:       EQU Q#7
;
;C=CONTINUE; R=REGISTER; D=DIRECT(MAP);
;SBR=SUBROUTINE REGISTER; RTN=RETURN FROM SUBROUTINE;
;F=FILE REFERENCE; PUSH=PUSH AND CONTINUE
;POP=POP AND CONTINUE; PR=POP AND BRANCH REGISTER
;
INV:       EQU B#0
;
;TEST VARIABLES:
Z:         EQU H#0
CY:        EQU H#1
P:         EQU H#2
S:         EQU H#3
AC:        EQU H#4
INT:       EQU H#8
READY:     EQU H#9
HOLD:      EQU H#A
F3:        EQU H#C
F0:        EQU H#D
CN.4:      EQU H#E
;
;THE NUMERICAL FIELD IS 12 BIT WIDE. THE 8 LS BITS CAN BE
;USED FOR DIRECT DATA BY LETTING THE FIRST VARIABLE TO 'DBUS'
;
; NUM DEF DB, NUMBER IN HEXA.
NUM:       DEF 1X, 1VB#1, 12V%H#000, 42X
;
;VARIABLE TO BE USED:
DBUS:      EQU B#0
;
;WIDE FIELD DEFINITIONS:
MMR:       DEF 1X, B#1, 12D#13%, Q#2, B#0, H#9, 1X, Q#72, 27X
MMW:       DEF 1X, B#1, 12D#14%, Q#2, B#0, H#9, 1X, Q#74, 27X
HLD:       DEF 1X, B#1, 12D#10%, Q#2, B#1, H#A, 1X, Q#76, 27X
INCPC:     DEF B#1, 21X, B#0, 6X, B#0010011011, H#FF,Q#304
FMMR:      DEF B#11, 12V%, H#0976, H#0C551, 6X
FMMW:      DEF B#11, 12V%, H#0979, H#0C551, 6X
NALU:   DEF 22X, B#0, 10X, B#0, H#D, B#0, H#AA, Q#1, 6X
END
END
~
```

APPENDIX II

Microcode Source Code and Object Code Output In Interleaved Format CSC File Name: AM80O

Note: A listing of the Source Code (Assembly File) only may be obtained using CSC File Name AMASSY.


```
EMULATOR ASSEMBLY (MARCH 1977)
07/06/77

PC    SOURCE AND/OR OBJECT CODE.   X = DON'T CARE

0000  ;INITIALIZATION:

0000  RESET:   ALU DOUBLE, PC,PC,FTOB.F & AND & ZA & ALUC & BASW &
               /IOC,,TO.INTE & IF,INV, & NUM DBUS,H#38 & NOC
0000 1000000011100000 0011110111110001 1011011111111110 11100100

0001           ALU,,H#D,FTOB.F & OR & DZ & ALUC & BASW & IOC &
               /NOC & IF,INV & NUM
0001 1100000000000000 0011110111110000 0011010101011010 11011111

0002           ALU,,H#C,FTOB.F & AND & ZA & ALUC & BASW & IOC &
               /NOC & IF,INV & NUM
0002 1100000000000000 0011110111110000 0011010101011000 11100100

0003           ALU,,,FTOB.F & AND & ZA & ALUC & BASW & IOC,,TO.A &
               /NOC & IF R.PUSH & NUM
0003 1100000000000011 0111110111110001 0011010101010100 11100100

0004  FETCH:   ALU DOUBLE,PC,PC,FTOB.F & OR & ZA & ALUC & BASW &
               /IOC IN,,TO.A & MEMR & IF ,INV,READY & NUM, $
0004 0100000000010000 0010010111010001 0011011111111110 11011100

0005           INCPC & IF D.R, ,HOLD & NUM,HLDD & NOC
0005 1100000000110000 1110100111110001 0011011111111110 11000100

0000  ;

0000  ;HOLD AND MEMORY REFERENCE SUBROUTINES AND HANDLERS:

0000  ;

0000  ORG 10

000A  HLDSB:   NALU & IOC & HLDA & IF R.RTN, INV, HOLD & NUM, $
000A 1100000000101001 1010100111111000 0011010101010100 01XXXXXX

000B  HLDF:      NALU & IOC & HLDA & IF R.F,INV,HOLD & NUM, $
000B 1100000000101111 1010100111111000 0011010101010100 01XXXXXX

000C  HLDD:      NALU & IOC & HLDA & IF D.R,,HOLD & NUM,  $
000C 1100000000110000 1110100111111000 0011010101010100 01XXXXXX

000D  MMRSB:     ALU DOUBLE,PC,PC,FTOB.F & OR & ZA & IOC,,TO.A &
                 /ALUC & MEMR & IF R.RTN, ,READY & NUM , $ & BASW
000D 1100000000110101 1110010111010001 0011011111111110 11011100

000E  MMWSB:     ALU DOUBLE,PC,PC,FTOB.F & OR & ZA & ALUC &
                 /IOC, DH, TO.A & NUM , $ & IF R.RTN,,READY & BASW & ME
MW

000E 1100000000111001 1110010111100101 0011011111111110 11011100

000F  MMWF:    ALU DOUBLE,PC,PC,FTOB.F & OR & ZA & ALUC & BASW &
               /MEMW & IF R.F,,READY & IOC, DH,TO.A & NUM, $
000F 1100000000111111 1110010111100101 0011011111111110 11011100

0010  MMRF:    ALU DOUBLE,PC,PC,FTOB.F & OR & ZA & ALUC & BASW & MEMR &
```

```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE
              /IOC,,TD.A & IF R.F ,,READY & NUM , $
0010 1100000001000011 1110010111010001 0011011111111110 11011100

0011  MMRSP:    ALU DOUBLE,SP,SP,FTOB.F & OR & ZA & IOC,,TD.A & BASW &
              /ALUC & MEMR & IF R.RTN,,READY & NUM, $
0011 1100000001000101 1110010111010001 0011011100010000 11011100

0012  MMWSPH: ALU DOUBLE,SP,SP,FTOB.F & OR & ZA & ALUC & BASW &
              /IOC,DH,TD.A & MEMW & IF R.RTN,,READY & NUM, $
0012 1100000001001001 1110010111100101 0011011100010000 11011100

0013  MMWSPL: ALU DOUBLE,SP,SP,FTOB.F & OR & ZA & ALUC & BASW &
              /IOC,DL,TD.A & MEMW & IF R.RTN,,READY & NUM, $
0013 1100000001001101 1110010111100011 0011011100010000 11011100

0000  ;

0000  ;MACROCODES:

0000  ;

0014  MOVRR:    ALU,,,FTOB.F & ALUC & BASW SW,SW & OR & ZA & IOC &
              /IF R.F, INV,HOLD & NUM, HLDF & NOC
0014 1100000000101111 1010101111110000 0111010101010100 11011100

0015  MOVMR:    ALU DOUBLE , H & ALUC & OR & ZA & BASW & IOC,,TD.A & HLD


0015 1100000000101001 0110100111110001 0011011010010100 01011100

0016           ALU & OR & ZA & BASW, SW & ALUC & IOC,,TD.D & HLD
0016 1100000000101001 0110100111110000 1111010101010100 01011100

0017           ALU DOUBLE,PC,PC,FTOB.F & ALUC & OR & ZA & BASW &
              /MEMW & IF R.F,,READY & NUM, $ &  IOC, DH, TD.A
0017 1100000001011111 1110010111100101 0011011111111110 11011100

0018  MOVRM:    ALU DOUBLE, H & ALUC & OR & ZA & BASW & IOC,,TD.A & HLD


0018 1100000000101001 0110100111110001 0011011010010100 01011100

0019       ALU DOUBLE,PC,PC,FTOB.F & OR & ZA & ALUC & BASW & MMR &
              /IOC,,TD.A
0019 1100000000110101 0010010111010001 0011011111111110 11011100

001A           ALU,,,FTOB.F & OR & DZ & BASW SW & ALUC & IOC &
              /NOC & IF R.F, INV, HOLD & NUM, HLDF
001A 1100000000101111 1010101111110000 0011010101010100 11011111

001B  MVIR:     INCPC & MMR
001B 1100000000110101 0010010111010001 0011011111111110 11000100

001C           ALU,,,FTOB.F & OR & DZ & ALUC & BASW SW & IOC
              / & NOC & IF R.F, INV, HOLD & NUM, HLDF
001C 1100000000101111 1010101111110000 0011010101010100 11011111

001D          NALU
001D XXXXXXXXXXXXXXXX XXXXXX0XXXXXXXXX X0110101010101100 01XXXXXX

001E  MVIM:     NALU & MMR & IOC
```



```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE
001E 1100000000110101 0010010111010000 0011010101010100 01XXXXXX

001F           ALU,,,FTOB.F & BASW & OR & DZ & ALUC & IOC ,,TD.D & HLD
001F 1100000000101001 0110100111110000 1011010101010100 11011111

0020           ALU DOUBLE,H & OR & ZA & ALUC & BASW & IOC,,TD.A & HLD
0020 1100000000101001 0110100111110001 0011011010010100 01011100

0021         ALU DOUBLE,PC,PC,FTOB.F & PLUS & ZA & ALUC & BASW &
              /MEMW & IF R.F,,READY & NUM, MMWF & IOC,DH,TD.A
0021 1100000000111111 1110010111100101 0011011111111110 11000100

0022  LXIB:     INCPC & MMR
0022 1100000000110101 0010010111010001 0011011111111110 11000100

0023           ALU,,C,FTOB.F & OR & DZ & ALUC & BASW & HLD & IOC
0023 1100000000101001 0110100111110000 0011010101000010 11011111

0024           INCPC & MMR
0024 1100000000110101 0010010111010001 0011011111111110 11000100

0025           ALU,,B,FTOB.F, & OR & DZ & ALUC & BASW & NOC &
              /IF R.F,INV,HOLD & NUM,HLDF & IOC
0025 1100000000101111 1010100111110000 0011010101000000 11011111

0026  LDA:     INCPC & MMR
0026 1100000000110101 0010010111010001 0011011111111110 11000100

0027           ALU DOUBLE,,;FTOB.F & OR & DZ & ALUC & BASW & HLD & IOC
0027 1100000000101001 0110100111110000 0011011101010100 11011111

0028           INCPC & MMR
0028 1100000000110101 0010010111010001 0011011111111110 11000100

0028           INCPC & MMR
0028 1100000000110101 0010010111010001 0011011111111110 11000100

0029           ALU,,,FTOB.F & OR & DZ & ALUC & BASW & HLD & IOC
0029 1100000000101001 0110100111110000 0011010101010100 11011111

002A           ALU DOUBLE,, & OR & ZA & ALUC & BASW & IOC,,TD.A & HLD
002A 1100000000101001 0110100111110001 0011011101010100 01011100

002B            ALU DOUBLE,PC,PC,FTOB.F & OR & ZA & ALUC & BASW &
              /IOC,,TD.A & MMR
002B 1100000000110101 0010010111010001 0011011111111110 11011100

002C            ALU,A,A,FTOB.F & OR & DZ & ALUC & BASW & IOC &
              /NOC & IF R.F,INV,HOLD & NUM, HLDF
002C 1100000000101111 1010100111110000 0011010011101110 11011111

002D  STA:     INCPC & MMR
002D 1100000000110101 0010010111010001 0011011111111110 11000100

002E           ALU DOUBLE,,,FTOB.F & OR & DZ & ALUC & BASW & IOC & HLD
002E 1100000000101001 0110100111110000 0011011101010100 11011111

002F           INCPC & MMR
002F 1100000000110101 0010010111010001 0011011111111110 11000100

0030           ALU,,,FTOB.F & OR & DZ & ALUC & BASW & IOC & HLD
0030 1100000000101001 0110100111110000 0011010101010100 11011111

0031           ALU DOUBLE,,,FTOB.F & OR & ZA & ALUC & BASW & IOC,,TD.A &
              /HLD
```

```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE
0031 1100000000101001 0110100111110001 0011011101010100 11011100

0032          ALU,A,A,FTOB.F & OR & ZA & ALUC & BASW & IOC,,TO.D & HLD


0032 1100000000101001 0110100111110000 1011010011101110 11011100

0033          ALU DOUBLE,PC,PC,FTOB.F & OR & ZA & ALUC & BASW &
              /IOC,DH,TO.A & MEMW & IF R.F,,READY & NUM, MMWF
0033 1100000000111111 1110010111100101 0011011111111110 11011100

0034  ADDR:     ALU,,A,FTOB.F & PLUS & AB & BASW,SW & ALUC UPDTALL,,CNL


             &/IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0034 1100000000101111 1010100111110000 0100000101001110 11000001

0035  ADDM:     ALU DOUBLE,H & OR & ZA & ALUC & BASW & IOC,,TO.A & HLD
0035 1100000000101001 0110100111110001 0011011010010100 01011100

0036  ADDM1:          ALU DOUBLE,PC,PC,FTOB.F &OR & ZA & ALUC & BASW &
              /IOC ,,TO.A & MMR
0036 1100000000110101 0010010111010001 0011011111111110 11011100

0037          ALU,A,A,FTOB.F & PLUS & DA & ALUC UPDTALL,,CNL &
              /BASW & IOC & NOC & IF R.F,INV,HOLD & NUM, HLDF
0037 1100000000101111 1010100111110000 0000000011101110 11000101

0038  ADI:      INCPC & MMR
0038 1100000000110101 0010010111010001 0011011111111110 11000100

0039          ALU,A,A,FTOB.F & PLUS & DA & ALUC UPDTALL,,CNL &
              /BASW & IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0039 1100000000101111 1010100111110000 0000000011101110 11000101

003A  ADCR:     NALU & IOC & NOC & IF,INV,CY & NUM,ADDR
003A 1100000011010000 0000010111110000 0011010101010100 01XXXXXX

003B          ALU,,A,FTOB.F & PLUS & AB & BASW,SW & ALUC UPDTALL &
              /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
003B 1100000000101111 1010100111110000 0100010101001110 11000001

003C  ADCM:  ALU DOUBLE,H & OR & ZA & ALUC & BASW & IOC,,TO.A &
             /NOC & IF,INV,CY & NUM,ADDM1
003C 1100000011011000 0000010111110001 0011011010010100 01011100


003D        ALU DOUBLE ,PC,PC,FTOB.F & OR & ZA & ALUC & BASW &
            /IOC,,TO.A & MMR
003D 1100000000110101 0010010111010001 0011011111111110 11011100

003E        ALU,A,A,FTOB.F & PLUS & DA & ALUC UPDTALL & BASW &
            /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
003E 1100000000101111 1010100111110000 0000010011101110 11000101

003F  ACI:  NALU & IOC & NOC & IF,INV,CY & NUM,ADI
003F 1100000011100000 0000010111110000 0011010101010100 01XXXXXX


0040        INCPC & MMR
0040 1100000000110101 0010010111010001 0011011111111110 11000100
```



```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE


0041        ALU,A,A,FTOB.F & PLUS & DA & ALUC UPDTALL & BASW &
            /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0041 1100000000101111 1010100111110000 0000010011101110 11000101

0042  JMP:     INCPC & MMR
0042 1100000000110101 0010010111010001 0011011111111110 11000100

0043         ALU DOUBLE,,,FTOB.F & OR & DZ & ALUC & BASW &
             /IOC & HLD
0043 1100000000101001 0110100111110000 0011011101010100 11011111

0044         INCPC & MMR
0044 1100000000110101 0010010111010001 0011011111111110 11000100

0045         ALU,,,FTOB.F & OR & DZ & ALUC & BASW & IOC & HLD
0045 1100000000101001 0110100111110000 0011010101010100 11011111

0046        ALU DOUBLE,,PC,FTOB.A & OR & ZA & ALUC & BASW & IOC,,TO.A &


            / NOC & IF R.F,INV,HOLD & NUM,HLDF
0046 1100000000101111 1010100111110001 0011011101011110 10011100

0047  CALL:    INCPC & MMR
0047 1100000000110101 0010010111010001 0011011111111110 11000100

0048          ALU DOUBLE,,,FTOB.F & OR & DZ & ALUC & BASW & IOC &
              /HLD
0048 1100000000101001 0110100111110000 0011011101010100 11011111

0049          INCPC & MMR
0049 1100000000110101 0010010111010001 0011011111111110 11000100

004A          ALU,,,FTOB.F & OR & DZ & ALUC & BASW & IOC & HLD
004A 1100000000101001 0110100111110000 0011010101010100 11011111

004B          ALU DOUBLE,SP,SP,FTOB.F & SUNIM & ZA & ALUC,,CNL &
              /BASW & IOC,,TO.A & HLD
004B 1100000000101001 0110100111110001 0011001100010000 11001100

004C          ALU DOUBLE,PC & OR & ZA & ALUC & BASW & IOC,,TO.D & HLD
004C 1100000000101001 0110100111110000 1011011111110100 01011100

004D          ALU DOUBLE,SP,SP,FTOB.F & SUNIM & ZA & ALUC,,CNL &
              /IOC,DH,TO.A & BASW & MEMW & IF C.SBR,INV,READY &
              /NUM, MMWSPH
004D 1100000001001001 0010010111100101 0011001100010000 11001100

004E          ALU DOUBLE,,PC,FTOB.F & OR & ZA & ALUC & BASW &
              /IOC,DL,TO.A & MEMW & IF R.F,,READY & NUM, $
004E 1100000100111011 1110010111100011 0011011101011110 11011100

004F            NALU
004F XXXXXXXXXXXXXXXX XXXXXX0XXXXXXXXX X011010101010100 01XXXXXX

0050  RET:     ALU DOUBLE,SP,SP,FTOB.A & PLUS & ZA & BASW & ALUC &
              /IOC,,TO.A & NOC & IF C.SBR & NUM,MMRSP
0050 1100000001000101 0111110111110001 0011011100010000 10000100
```

```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE

0051          ALU DOUBLE,PC,PC,FTOB.F & OR & DZ & BASW & ALUC &
              /IOC & HLD
0051 1100000000101001 0110100111110000 0011011111111110 11011111

0052          ALU DOUBLE,SP,SP,FTOB.A & PLUS & ZA & BASW & ALUC &
              /IOC,,TO.A & MEMR & IF C.SBR,INV,READY & NUM,MMRSP
0052 1100000001000101 0010010111010001 0011011100010000 10000100

0053          ALU,PC,PC,FTOB.F & OR & DZ & ALUC & BASW & IOC & HLD
0053 1100000000101001 0110100111110000 0011010111111110 11011111

0054         ALU DOUBLE,PC,PC, & OR & ZA & ALUC & BASW & IOC,,TO.A &
             /NOC & IF R.F,INV,HOLD & NUM, HLDF
0054 1100000000101111 1010100111110001 0011011111111110 01011100

0055  RST:       ALU DOUBLE,H#C,,FTOB.F & AND & DA & ALUC & BASW &
              /IOC & HLD
0055 1100000000101001 0110100111110000 0011011110010100 11100101

0056   RST1:    ALU DOUBLE,PC & OR &  ZA & ALUC & BASW & IOC,,TO.D & HLD


0056 1100000000101001 0110100111110000 1011011111110100 01011100

0057          ALU DOUBLE,SP,SP,FTOB.F & SUNIM & ZA & ALUC,,CNL &
              /BASW & IOC,,TO.A & NOC & IF C.SBR & NUM, MMWSPH
0057 1100000001001001 0111110111110001 0011001100010000 11001100

0058          ALU DOUBLE, SP,SP,FTOB.F & SUNIM & ZA & ALUC,,CNL &
              /BASW & IOC,,TO.A & HLD
0058 1100000000101001 0110100111110001 0011001100010000 11001100

0059          ALU DOUBLE, ,PC,FTOB.F & ZA & OR & ALUC & BASW & MEMW &


              /IOC,DL,TO.A & IF R.F,,READY & NUM, $
0059 1100000101100111 1110010111100011 0011011101011110 11011100

005A  RLC:       ALU,A,A,UROT.F & OR & ZA & BASW & IOC &
              /ALUC UPDTCY & HLD
005A 1100000000101001 0110100111110000 0001010011101111 11011100

005B  RLC1:      NALU & IOC & NOC & IF R.F,INV,F3 & NUM,STC
005B 1100001001110011 1011000111110000 0011010101010100 01XXXXXX

005C          ALU & OR & ALUC UPDTCY & BASW & IOC &
              /NOC & IF R.F,INV,HOLD & NUM,HLDF
005C 1100000000101111 1010100111110000 0001010101010100 01011XXX

005D  RRC:       ALU,A,A,DROT.F & OR & ZA & BASW & IOC & ALUC UPDTCY &
              /HLD
005D 1100000000101001 0110100111110000 0001010011101111 01011100

005E          ALU,A  & OR & ZA & BASW & IOC & ALUC & NOC & IF & NUM,RL
C1

005E 1100000101101100 0111110111110000 0011010011110100 01011100

005F  RAL:       ALU,A,A,UROT.F & OR & ZA & BASW & IOC & ALUC UPDTCY,SWAP
&
```



```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE
              /NOC & IF & NUM,RLC1
005F 1100000101101100 0111110111110000 0001110011101111 11011100

0060  RAR:       ALU,A,A,DROT.F & OR & ZA & BASW & IOC & ALUC & HLD
0060 1100000000101001 0110100111110000 0011010011101111 01011100

0061          ALU,A,A,UROT.F & OR & ZA & BASW & IOC & ALUC & HLD
0061 1100000000101001 0110100111110000 0011010011101111 11011100

0062          ALU,A,A,DROT.F & OR & ZA & BASW & IOC & ALUC UPDTCY,
              /SWAP & NOC & IF R.F,INV,F3 & NUM,STC
0062 1100001001110011 1011000111110000 0001110011101111 01011100

0063          ALU & OR & ALUC UPDTCY & BASW & IOC & NOC &
              /IF R.F,INV,HOLD & NUM,HLDF
0063 1100000000101111 1010100111110000 0001010101010100 01011XXX

0064  PUSHRP:  ALU DOUBLE & OR & ZB & ALUC & IOC,,TO.D & BASW SW & HLD


0064 1100000000101001 0110101111110000 1011011101010100 01011011

0065          ALU DOUBLE,SP,SP,FTOB.F & SUNIM & ZA & ALUC,,CNL &
              /IOC,,TO.A & BASW & HLD
0065 1100000000101001 0110100111110001 0011001100010000 11001100

0066          ALU DOUBLE,SP,SP,FTOB.F & SUNIM & ZA & ALUC,,CNL &
              /BASW & IOC,DH,TO.A & MEMW & IF C.SBR,INV,READY &
              /NUM,MMWSPH
0066 1100000001001001 0010010111100101 0011001100010000 11001100

0067          ALU DOUBLE,PC,PC,FTOB.F & OR & ZA & ALUC & BASW &
             /IOC,DL,TO.A & MEMW & IF R.F,,READY & NUM, $
0067 1100000110011111 1110010111100011 0011011111111110 11011100

0068  PUSHPSW: ALU,A & OR & ZA & ALUC & IOC,,TO.D & BASW & HLD
0068 1100000000101001 0110100111110000 1011010011110100 01011100

0069          ALU DOUBLE,SP,SP,FTOB.F & SUNIM & ZA & ALUC,,CNL &
              /BASW & IOC,,TO.A & NOC & IF C.SBR & NUM,MMWSPH
0069 1100000001001001 0111110111110001 0011001100010000 11001100

006A          NALU & IOC,FLAGS & MEMW & IF,INV,READY & NUM,$
006A 1100000110101000 0010010111100110 0011010101010100 01XXXXXX

006B          ALU DOUBLE,SP,SP,FTOB.F & SUNIM & ZA & ALUC,,CNL & BASW


             &/IOC & HLD
006B 1100000000101001 0110100111110000 0011001100010000 11001100

006C          ALU DOUBLE,PC,PC,FTOB.F & OR & ZA & ALUC & BASW & IOC,,T


        O.A &/NOC & IF R.F,INV,HOLD & NUM,HLDF
006C 1100000000101111 1010100111110001 0011011111111110 11011100

006D  INXB:      ALU DOUBLE,,B,FTOB.F & PLUS & ZB & ALUC & BASW & IOC & H
              LD

006D 1100000000101001 0110100111110000 0011011101000000 11000011
```

```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE

006E            ALU DOUBLE,B,C,FTOB.A & OR & DZ & ALUC,SWAP & BASW &
                /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
006E 1100000000101111 1010100111110000 0011111000000010 10011111

006F  DCXB:     ALU DOUBLE,,B,FTOB.F & SUNIM & ZB & ALUC,,CNL &
                /BASW & IOC & HLD
006F 1100000000101001 0110100111110000 0011001101000000 11001011

0070            ALU DOUBLE,B,C,FTOB.A & OR & DZ & ALUC,SWAP & IOC &
                /NOC & IF R.F,INV,HOLD & NUM,HLDF
0070 1100000000101111 101010X111110000 0X11111000000010 10011111

0071  DAD.B:    ALU DOUBLE,B,H,FTOB.F & PLUS & AB & ALUC UPDTCY,,CNL & I

        OC &/BASW & HLD
0071 1100000000101001 0110100111110000 0001001000001000 11000001

0072  DAD1:     ALU DOUBLE,H,L,FTOB.A & OR & DZ & ALUC,SWAP & BASW &
                /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0072 1100000000101111 1010100111110000 0011111010001010 10011111

0073  DAD.D:    ALU DOUBLE,D,H,FTOB.F & PLUS & AB & ALUC UPDTCY,,CNL & I

        OC &/BASW & NOC & IF & NUM,DAD1
0073 1100000111001000 0111110111110000 0001001001001000 11000001

0074  DAD.H:    ALU DOUBLE,H,H,FTOB.F & PLUS & AB & ALUC UPDTCY,,CNL & I

        OC &/BASW & NOC & IF & NUM,DAD1
0074 1100000111001000 0111110111110000 0001001010001000 11000001

0075  DAD.SP:   ALU DOUBLE,SP,H,FTOB.F & PLUS & AB & ALUC UPDTCY,,CNL &

        IOC &/BASW & NOC & IF & NUM,DAD1
0075 1100000111001000 0111110111110000 0001001100001000 11000001

0076  CMPR:     ALU,,A, & BASW,SW & AB & SUNIM & ALUC UPDTALL & IOC &
                /NOC & IF R.F,INV,HOLD & NUM,HLDF
0076 1100000000101111 1010100111110000 0100010101001110 01001001

0077  CPI:      INCPC & MMR
0077 1100000000110101 0010010111010001 0011011111111110 11000100

0078            ALU,A & BASW & DA & SUNIM & ALUC UPDTALL & IOC &
                /NOC & IF R.F,INV,HOLD & NUM,HLDF
0078 1100000000101111 1010100111110000 0000010011110100 01001101

0079  CMPM:     ALU DOUBLE,H & OR & ZA & IOC,,TO.A & BASW & ALUC & HLD
0079 1100000000101001 0110100111110001 0011011010010100 01011100

007A            ALU DOUBLE,PC & OR & ZA & IOC,,TO.A & BASW & ALUC & MMR

007A 1100000000110101 0010010111010001 0011011111110100 01011100

007B            ALU,A & BASW & DA & SUNIM & ALUC UPDTALL & IOC &
```



```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE

                /NOC & IF R.F,INV,HOLD & NUM,HLDF
007B 1100000000101111 1010100111110000 0000010011110100 01001101

007C  POP.B:    ALU DOUBLE,SP,SP,FTOB.A & OR & ZA & ALUC & BASW &
                /IOC,,TO.A & NOC & IF C.SBR,,HOLD & NUM,HLDSB
007C 1100000000101001 0110100111110001 0011011100010000 10011100

007D            ALU DOUBLE,SP,SP,FTOB.F & PLUS & ZA & ALUC & BASW &
                / IOC,,TO.A & MEMR & IF C.SBR,INV,READY & NUM,MMRSP
007D 1100000001000101 0010010111010001 0011011100010000 11000100

007E            ALU,,C,FTOB.F & OR & DZ & ALUC & IOC & BASW & HLD
007E 1100000000101001 0110100111110000 0011010101000010 11011111

007F            ALU DOUBLE,SP,SP,FTOB.F & PLUS& ZA & ALUC & BASW &
                /IOC,,TO.A & MEMR & IF C.SBR,INV,READY & NUM,MMRSP
007F 1100000001000101 0010010111010001 0011011100010000 11000100

0080            ALU,,B,FTOB.F & OR & DZ & ALUC & IOC & BASW & HLD
0080 1100000000101001 0110100111110000 0011010101000000 11011111

0081            ALU DOUBLE,PC & OR & ZA & ALUC & IOC,,TO.A & BASW &
                /NOC & IF R.F,INV,HOLD & NUM,HLDF
0081 1100000000101111 1010100111110001 0011011111110100 01011100

0082  HLT:      NALU & IOC'& HLD
0082 1100000000101001 0110100111110000 0011010101010100 01XXXXXX

0083            NALU & IOC & NOC & IF ,INV,INT & NUM,HLT
0083 1100001000001000 0010000111110000 0011010101010100 01XXXXXX

0084  INTHNDL:  NALU & IOC & INTA & IF,INV & NUM
0084 1100000000000000 0011110011110000 0011010101010100 01XXXXXX

0085            NALU & IOC & INTA & IF & NUM,RST1
0085 1100000101011000 0111110011110000 0011010101010100 01XXXXXX

0086  NOP:      NALU & IOC & IF R.F,INV,HOLD & NUM,HLDF & NOC
0086 1100000000101111 1010100111110000 0011010101010100 01XXXXXX

0087  IN.:      INCPC & MMR
0087 1100000000110101 0010010111010001 0011011111111110 11000100

0088            ALU & OR & DZ & ALUC & IOC,,TO.A & BASW & HLD
0088 1100000000101001 0110100111110001 0011010101010100 01011111

0089            ALU DOUBLE,PC & OR & ZA & BASW & ALUC & IOC,,TO.A &
                /IOR & IF,INV,READY & NUM,$
0089 1100001000100100 0010010101110001 0011011111110100 01011100

008A            ALU,,A,FTOB.F & OR & DZ & ALUC & IOC & BASW & NOC &
                /IF R.F,INV,HOLD & NUM,HLDF
008A 1100000000101111 1010100111110000 0011010101001110 11011111

008B  OUT.:     ALU,A & OR & ZA & IOC,,TO.D & BASW & ALUC & HLD
008B 1100000000101001 0110100111110000 1011010011110100 01011100

008C            INCPC & MMR
008C 1100000000110101 0010010111010001 0011011111111110 11000100
```

```
PC     SOURCE AND/OR OBJECT CODE.    X = DON'T CARE

008D              ALU & OR & DZ & ALUC & IOC,,TO.A & BASW & HLD
008D 1100000000101001 0110100111110001 0011010101010100 01011111

008E              ALU DOUBLE,PC & OR & ZA & BASW & ALUC & IOC,DH,TO.A &
                  /IOW & IF R.F,,READY & NUM,$
008E 1100001000111011 1110010110110101 0011011111110100 01011100

008F  EI:        ALU,H#C & NXOR & ZA & IOC,,TO.INTE & BASW & ALUC &
                  /NOC & IF R.F,INV,HOLD & NUM,HLDF
008F 1100000000101111 1010100111110001 1011010110010100 01111100

0090  DI:         ALU & AND & ZA & IOC,,TO.INTE & BASW & ALUC & NOC &
                  /IF R.F,INV,HOLD & NUM,HLDF
0090 1100000000101111 1010100111110001 1011010101010100 01100100

0091  SPHL:       ALU DOUBLE,H,SP,FTOB.F & OR & ZA & ALUC & BASW & IOC &


                  /NOC & IF R.F,INV,HOLD & NUM,HLDF
0091 1100000000101111 1010100111110000 0011011010010000 11011100

0092  XTHL:       ALU DOUBLE,H & OR & ZA & ALUC & BASW & IOC,,TO.D & HLD


0092 1100000000101001 0110100111110000 1011011010010100 01011100

0093              ALU DOUBLE,SP & OR & ZA & ALUC & BASW & IOC,,TO.A & HLD

0093 1100000000101001 0110100111110001 0011011100010100 01011100

0094              NALU & IOC & MEMR & IF,INV,READY & NUM,$
0094 1100001001010000 0010010111010000 0011010101010100 01XXXXXX

0095              ALU,,L,FTOB.F & OR & DZ & ALUC & IOC & BASW & HLD
0095 1100000000101001 0110100111110000 0011010101001010 11011111

0096              ALU DOUBLE,SP,SP,FTOB.F & PLUS & ZA & ALUC & BASW &
                  /IOC,DL,TO.A & MEMW & IF C.SBR,INV,READY & NUM,MMWSPL
0096 1100000001001101 0010010111100011 0011011100010000 11000100

0097              NALU & IOC & MEMR & IF,INV,READY & NUM,$
0097 1100001001011100 0010010111010000 0011010101010100 01XXXXXX

0098              ALU,,H,FTOB.F & OR & DZ & ALUC & IOC & BASW & HLD
0098 1100000000101001 0110100111110000 0011010101001000 11011111

0099              ALU DOUBLE,PC & OR & ZA & ALUC & BASW & IOC,DH,TO.A &
                  /MEMW & IF,INV,READY & NUM,$
0099 1100001001100100 0010010111100101 0011011111110100 01011100

009A              ALU DOUBLE,SP,SP,FTOB.F & ZA & SUNIM & ALUC,,CNL &
                  /BASW & IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
009A 1100000000101111 1010100111110000 0011001100010000 11001100

009B  PCHL:       ALU DOUBLE,H,PC,FTOB.F & OR & ZA & ALUC & IOC,,TO.A &
                  /BASW & NOC & IF R.F,INV,HOLD & NUM,HLDF
009B 1100000000101111 1010100111110001 0011011010011110 11011100

009C  STC:        ALU , H#C & SUNIM & ZA & ALUC UPDTCY,,CNL & BASW & IOC
```



```
PC     SOURCE AND/OR OBJECT CODE.    X = DON'T CARE
          & NOC & /IF R.F,INV,HOLD & NUM,HLDF
009C 1100000000101111 1010100111110000 0001000110010100 01001100

009D  CMC:        ALU & AND & ZA & ALUC UPDTCY & BASW & IOC & NOC &
                  /IF R.F,,CY & NUM, STC
009D 1100001001110011 1100010111110000 0001010101010100 01100100

009E              ALU & AND & ZA & ALUC UPDTCY,,CNL & BASW & IOC &
                  /NOC & IF R.F,INV,HOLD & NUM,HLDF
009E 1100000000101111 1010100111110000 0001000101010100 01100100

009F  ANAR:       ALU,,A,FTOB.F & AND & AB & ALUC UPDTALL & BASW ,SW &
                  /IOC & NOC & IF R.F,INV,HOLD &NUM,HLDF
009F 1100000000101111 1010100111110000 0100010101001110 11100001

00A0  XRAR:       ALU,,A,FTOB.F & XOR & AB & ALUC UPDTALL & BASW,SW &
                  /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00A0 1100000000101111 1010100111110000 0100010101001110 11110001

00A1  ORAR:       ALU,,A,FTOB.F & OR & AB & ALUC UPDTALL & BASW ,SW &
                  /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00A1 1100000000101111 1010100111110000 0100010101001110 11011001

00A2  DCRM:      ALU DOUBLE,H & OR & ZA & ALUC & IOC,,TO.A & BASW & HLD
00A2 1100000000101001 0110100111110001 0011011010010100 01011100

00A3              NALU & IOC & MEMR & IF,INV,READY & NUM,$
00A3 1100001010001100 0010010111010000 0011010101010100 01XXXXXX

00A4              ALU & DZ & MINUS & ALUC UPDTFL,,CNL & BASW &
                  /IOC,,TO.D & HLD
00A4 1100000000101001 0110100111110000 1010000101010100 01010111

00A5              ALU DOUBLE,PC & OR & ZA & ALUC & BASW & IOC,DH,TO.A &
                  /MEMW & IF R.F,,READY & NUM,$
00A5 1100001010010111 1110010111100101 0011011111110100 01011100

00A6  INRM:       ALU DOUBLE,H & OR & ZA & ALUC & IOC,,TO.A & BASW & HLD
00A6 1100000000101001 0110100111110001 0011011010010100 01011100

00A7              NALU & IOC & MEMR & IF,INV,READY & NUM,$
00A7 1100001010011100 0010010111010000 0011010101010100 01XXXXXX

00A8              ALU & DZ & PLUS & ALUC UPDTFL & BASW & IOC,,TO.D & HLD
00A8 1100000000101001 0110100111110000 1010010101010100 01000111

00A9              ALU DOUBLE,PC & OR & ZA & ALUC & BASW & IOC,DH,TO.A &
                  /MEMW & IF R.F,,READY & NUM,$
00A9 1100001010100111 1110010111100101 0011011111110100 01011100

00AA  DCRR:       ALU,,,FTOB.F & ZB & SUNIM & ALUC UPDTFL,,CNL &
                  /BASW SW & IOC & IF R.F,INV,HOLD & NUM,HLDF & NOC
00AA 1100000000101111 1010101111110000 0010000101010100 11001011

00AB  INRR:       ALU,,,FTOB.F & ZB & PLUS & ALUC UPDTFL & BASW SW & IOC &


                  /NOC & IF R.F,INV,HOLD & NUM,HLDF
00AB 1100000000101111 1010101111110000 0010010101010100 11000011

00AC  SUBR:       ALU,,A,FTOB.F & AB & SUNIM & ALUC UPDTALL & BASW , SW &
```

```
PC     SOURCE AND/OR OBJECT CODE.    X = DON'T CARE
                 /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00AC 1100000000101111 1010100111110000 0100010101001110 11001001

00AD  SUBM:      ALU DOUBLE,H & OR & ZA & ALUC & BASW & IOC,,TO.A & HLD
00AD 1100000000101001 0110100111110001 0011011010010100 01011100

00AE             ALU DOUBLE,PC & OR & ZA & BASW & ALUC & IOC,,TO.A & MMR


00AE 1100000000110101 0010010111010001 0011011111110100 01011100

00AF             ALU,A,A,FTOB.F & DA & SUNIM & ALUC UPDTALL &
                 /BASW & IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00AF 1100000000101111 1010100111110000 0000010011101110 11001101

00B0  SUI:       INCPC & MMR
00B0 1100000000110101 0010010111010001 0011011111111110 11000100

00B1             ALU,A,A,FTOB.F & DA & SUNIM & ALUC UPDTALL &
                 /BASW & IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00B1 1100000000101111 1010100111110000 0000010011101110 11001101

00B2  SBBR:      NALU & IOC & NOC & IF,INV,CY & NUM,SUBR
00B2 1100001010110000 0000010111110000 0011010101010100 01XXXXXX

00B3             ALU,,A,FTOB.F & AB & SUNIM & BASW,SW & ALUC UPDTALL,,CN


           L &/IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00B3 1100000000101111 1010100111110000 0100000101001110 11001001

00B4  SBBM:      ALU DOUBLE,H & OR & ZA & IOC,,TO.A & BASW & ALUC &
                 /NOC & IF ,INV,CY & NUM,SUBM+1
00B4 1100001010111000 0000010111110001 0011011010010100 01011100

00B5             ALU DOUBLE, PC & OR & ZA & IOC,,TO.A & BASW & ALUC & MM
                 R

00B5 1100000000110101 0010010111010001 0011011111110100 01011100

00B6             ALU,A,A,FTOB.F & DA & SUNIM & ALUC UPDTALL,,CNL &
                 /BASW & IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00B6 1100000000101111 1010100111110000 0000000011101110 11001101

00B7  SBI:      NALU & IOC & NOC & IF,INV,CY & NUM,SUI
00B7 1100001011000000 0000010111110000 0011010101010100 01XXXXXX

00B8           NALU & IOC & NOC & IF & NUM,SBI1
00B8 1100010101110100 0111110111110000 0011010101010100 01XXXXXX

00B9  ANAM:      ALU DOUBLE,H & OR & ZA & IOC,,TO.A & ALUC & BASW & HLD


00B9 1100000000101001 0110100111110001 0011011010010100 01011100

00BA             ALU DOUBLE,PC & OR & ZA & IOC,,TO.A & ALUC & BASW & MMR

00BA 1100000000110101 0010010111010001 0011011111110100 01011100

00BB             ALU,A,A,FTOB.F & AND & DA & ALUC UPDTALL & BASW &
                 /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
```



```
PC     SOURCE AND/OR OBJECT CODE.    X = DON'T CARE
00BB 1100000000101111 1010100111110000 0000010011101110 11100101

00BC  XRAM:      ALU DOUBLE,H & OR & ZA & IOC,,TO.A & ALUC & BASW & HLD


00BC 1100000000101001 0110100111110001 0011011010010100 01011100

00BD             ALU DOUBLE,PC & OR & ZA & IOC,,TO.A & ALUC & BASW & MMR


00BD 1100000000110101 0010010111010001 0011011111110100 01011100

00BE             ALU,A,A,FTOB.F & XOR & DA &  ALUC UPDTALL & BASW &
                 /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00BE 1100000000101111 1010100111110000 0000010011101110 11110101

00BF  DRAM:      ALU DOUBLE,H & OR & ZA & IOC,,TO.A & ALUC & BASW & HLD


00BF 1100000000101001 0110100111110001 0011011010010100 01011100

00C0             ALU DOUBLE,PC & OR & ZA & IOC,,TO.A & ALUC & BASW & MMR


00C0 1100000000110101 0010010111010001 0011011111110100 01011100

00C1             ALU,A,A,FTOB.F & OR & DA & ALUC UPDTALL & BASW &
                 /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00C1 1100000000101111 1010100111110000 0000010011101110 11011101

00C2  ANI:       INCPC & MMR
00C2 1100000000110101 0010010111010001 0011011111111110 11000100

00C3             ALU,A,A,FTOB.F & AND & DA & ALUC UPDTALL & BASW & IOC &


                 /NOC & IF R.F,INV,HOLD & NUM,HLDF
00C3 1100000000101111 1010100111110000 0000010011101110 11100101

00C4  XRI:       INCPC & MMR
00C4 1100000000110101 0010010111010001 0011011111111110 11000100

00C5             ALU,A,A,FTOB.F & XOR & DA & ALUC UPDTALL & BASW & IOC &


                 /NOC & IF R.F,INV,HOLD & NUM,HLDF
00C5 1100000000101111 1010100111110000 0000010011101110 11110101

00C6  ORI:       INCPC & MMR
00C6 1100000000110101 0010010111010001 0011011111111110 11000100

00C7             ALU,A,A,FTOB.F & OR & DA & ALUC UPDTALL & BASW & IOC &
                 /NOC & IF R.F,INV,HOLD & NUM,HLDF
00C7 1100000000101111 1010100111110000 0000010011101110 11011101

00C8  CMA:       ALU,A,A,FTOB.F & NXOR & ZA & IOC & ALUC & BASW &
                 /NOC & IF R.F,INV,HOLD & NUM,HLDF
00C8 1100000000101111 1010100111110000 0011010011101110 11111100

00C9  LHLD:      INCPC & MMR
```

```
PC    SOURCE AND/OR OBJECT CODE.   X = DON'T CARE
00C9 1100000000110101 0010010111010001 0011011111111110 11000100

00CA            ALU DOUBLE,,,FTOB.F & OR & DZ & ALUC & BASW & IOC & HLD


00CA 1100000000101001 0110100111110000 0011011101010100 11011111

00CB            INCPC & MMR
00CB 1100000000110101 0010010111010001 0011011111111110 11000100

00CC            ALU,,,FTOB.F & OR & DZ & ALUC & BASW & IOC & HLD
00CC 1100000000101001 0110100111110000 0011010101010100 11011111

00CD            ALU DOUBLE,,,FTOB.F & OR & ZA & ALUC & IOC,,TO.A &
                /BASW & HLD
00CD 1100000000101001 0110100111110001 0011011101010100 11011100

00CE            NALU & IOC & MEMR & IF,INV,READY & NUM, $
00CE 1100001100111000 0010010111010000 0011010101010100 01XXXXXX

00CF            ALU,,L,FTOB.F & OR & DZ & IOC & BASW & ALUC & HLD
00CF 1100000000101001 0110100111110000 0011010101001010 11011111

00D0            ALU DOUBLE,,,FTOB.F & PLUS & ALUC & BASW & IOC,,TO.A & H


      LD/ & ZA
00D0 1100000000101001 0110100111110001 0011011101010100 11000100

00D1            ALU DOUBLE,PC & OR & ZA & IOC,,TO.A & BASW & ALUC &
                /MEMR & IF,INV,READY & NUM , $
00D1 1100001101000100 0010010111010001 0011011111110100 01011100

00D2            ALU,,H,FTOB.F & OR & DZ & IOC & BASW & ALUC &
                /NOC & IF R.F,INV,HOLD & NUM,HLDF
00D2 1100000000101111 1010100111110000 0011010101001000 11011111

00D3  SHLD:     INCPC & MMR
00D3 1100000000110101 0010010111010001 0011011111111110 11000100

00D4            ALU DOUBLE,,,FTOB.F & OR & DZ & ALUC & BASW & IOC & HLD


00D4 1100000000101001 0110100111110000 0011011101010100 11011111

00D5            INCPC & MMR
00D5 1100000000110101 0010010111010001 0011011111111110 11000100

00D6            ALU,,,FTOB.F & OR & DZ & ALUC & BASW & IOC & HLD
00D6 1100000000101001 0110100111110000 0011010101010100 11011111

00D7            ALU DOUBLE,,,FTOB.F & OR & ZA & ALUC & IOC,,TO.A &
                /BASW & HLD
00D7 1100000000101001 0110100111110001 0011011101010100 11011100

00D8            ALU DOUBLE,H & OR & ZA & IOC,,TO.D & BASW & ALUC & HLD
00D8 1100000000101001 0110100111110000 1011011010010100 01011100

00D9            NALU & IOC,DL & MEMW & IF,INV,READY & NUM, $
00D9 1100001101100100 0010010111100010 0011010101010100 01XXXXXX
```



```
PC    SOURCE AND/OR OBJECT CODE.   X = DON'T CARE

00DA            ALU DOUBLE,,,FTOB.F & PLUS & ZA & ALUC & BASW &
                /IOC,,TO.A & HLD
00DA 1100000000101001 0110100111110001 0011011101010100 11000100

00DB            ALU DOUBLE ,PC & OR & ZA & ALUC & IOC,DH,TO.A & BASW &
                /MEMW & IF R.F,,READY & NUM, $
00DB 1100001101101111 1110010111100101 0011011111110100 01011100

00DC  LDAX.B: ALU DOUBLE,B & BASW & ZA & OR & IOC,,TO.A & ALUC & HLD
00DC 1100000000101001 0110100111110001 0011011000010100 01011100
00DD            ALU DOUBLE,PC & OR & ZA & IOC,,TO.A & ALUC & BASW &
                /MEMR & IF,INV,READY & NUM, $
00DD 1100001101110100 0010010111010001 0011011111110100 01011100

00DE            ALU,,A,FTOB.F & OR & DZ & ALUC & BASW & IOC &
                /NOC & IF R.F,INV,HOLD & NUM,HLDF
00DE 1100000000101111 1010100111110000 0011010101001110 11011111

00DF  STAX:     ALU DOUBLE & BASW SW & ZB & OR & IOC,,TO.A & ALUC & HLD


00DF 1100000000101001 0110101111110001 0011011101010100 01011011

00E0            ALU,A & OR & ZA & IOC,,TO.D & BASW & ALUC & HLD
00E0 1100000000101001 0110100111110000 1011010011110100 01011100

00E1            ALU DOUBLE,PC & OR & ZA & IOC,DH,TO.A & BASW & ALUC &
                /MEMW & IF R.F,INV,HOLD & NUM, HLDF
00E1 1100000000101111 1010100111100101 0011011111110100 01011100

00E2  WXCHG:    ALU DOUBLE,D,,FTOB.F & OR & ZA & ALUC & BASW & IOC & HL
                D

00E2 1100000000101001 0110100111110000 0011011001010100 11011100

00E3            ALU DOUBLE,H,D,FTOB.F & OR & ZA & ALUC & BASW & IOC & HL
                D

00E3 1100000000101001 0110100111110000 0011011010000100 11011100

00E4            ALU DOUBLE,,H,FTOB.F & OR & ZA & ALUC & BASW & IOC &
                /NOC & IF R.F,INV,HOLD & NUM,HLDF
00E4 1100000000101111 1010100111110000 0011011101001000 11011100

00E5  LXID:     INCPC & MMR
00E5 1100000000110101 0010010111010001 0011011111111110 11000100

00E6            ALU,,E,FTOB.F & OR & DZ & ALUC & BASW & HLD & IOC
00E6 1100000000101001 0110100111110000 0011010101000110 11011111

00E7            INCPC & MMR
00E7 1100000000110101 0010010111010001 0011011111111110 11000100

00E8            ALU,,D,FTOB.F, & OR & DZ & ALUC & BASW & NOC &
                /IF R.F,INV,HOLD & NUM,HLDF & IOC
00E8 1100000000101111 1010100111110000 0011010101000100 11011111

00E9  LXIH:     INCPC & MMR
00E9 1100000000110101 0010010111010001 0011011111111110 11000100
```

```
PC     SOURCE AND/OR OBJECT CODE.    X = DON'T CARE

00EA             ALU,,L,FTOB.F & OR & DZ & ALUC & BASW & HLD & IOC
00EA 1100000000101001 0110100111110000 0011010101001010 11011111

00EB             INCPC & MMR
00EB 1100000000110101 0010010111010001 0011011111111110 11000100

00EC             ALU,,H,FTOB.F, & OR & DZ & ALUC & BASW & NOC &
                /IF R.F,INV,HOLD & NUM,HLDF & IOC
00EC 1100000000101111 1010100111110000 0011010101001000 11011111

00ED  LXISP:     INCPC & MMR
00ED 1100000000110101 0010010111010001 0011011111111110 11000100

00EE             ALU DOUBLE,,SP,FTOB.F & OR & DZ & ALUC & BASW & HLD & IO
                 C

00EE 1100000000101001 0110100111110000 0011011101010000 11011111

00EF             INCPC & MMR
00EF 1100000000110101 0010010111010001 0011011111111110 11000100

00F0             ALU,,SP,FTOB.F, & OR & DZ & ALUC & BASW & NOC &
                /IF R.F,INV,HOLD & NUM,HLDF & IOC
00F0 1100000000101111 1010100111110000 0011010101010000 11011111

00F1  INXD:      ALU DOUBLE,,D,FTOB.F & PLUS & ZB & ALUC & BASW & IOC & H
                 LD

00F1 1100000000101001 0110100111110000 0011011101000100 11000011

00F2             ALU DOUBLE,D,E,FTOB.A & OR & DZ & ALUC,SWAP & BASW &
                 /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00F2 1100000000101111 1010100111110000 0011111001000110 10011111

00F3  INXH:      ALU DOUBLE,,H,FTOB.F & PLUS & ZB & ALUC & BASW & IOC & H
                 LD

00F3 1100000000101001 0110100111110000 0011011101001000 11000011

00F4             ALU DOUBLE,H,L,FTOB.A & OR & DZ & ALUC,SWAP & BASW &
                 /IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00F4 1100000000101111 1010100111110000 0011111010001010 10011111

00F5  INXSP:      ALU DOUBLE,,SP,FTOB.F & PLUS & ZB & ALUC & BASW & IOC &


          /NOC & IF R.F,INV,HOLD & NUM,HLDF
00F5 1100000000101111 1010100111110000 0011011101010000 11000011

00F6  DCXD:      ALU DOUBLE,,D,FTOB.F & SUNIM & ZB & ALUC,,CNL &
                 /BASW & IOC & HLD
00F6 1100000000101001 0110100111110000 0011001101000100 11001011

00F7             ALU DOUBLE,D,E,FTOB.A & OR & DZ & ALUC,SWAP & IOC &
                 /NOC & IF R.F,INV,HOLD & NUM,HLDF
00F7 1100000000101111 101010X111110000 0X11111001000110 10011111

00F8  DCXH:      ALU DOUBLE,,H,FTOB.F & SUNIM & ZB & ALUC,,CNL &
                 /BASW & IOC & HLD
```



```
PC     SOURCE AND/OR OBJECT CODE.    X = DON'T CARE

00F8 1100000000101001 0110100111110000 0011001101001000 11001011

00F9             ALU DOUBLE,H,L,FTOB.A & OR & DZ & ALUC,SWAP & IOC &
                 /NOC & IF R.F,INV,HOLD & NUM,HLDF
00F9 1100000000101111 101010X111110000 0X11111010001010 10011111

00FA  DCXSP:     ALU DOUBLE,,SP,FTOB.F & SUNIM & ZB & ALUC,,CNL &
                 /BASW & IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
00FA 1100000000101111 1010100111110000 0011001101010000 11001011

00FB  POP.D:     ALU DOUBLE,SP,SP,FTOB.A & OR & ZA & ALUC & BASW &
                 /IOC,,TO.A & NOC & IF C.SBR,,HOLD & NUM,HLDSB
00FB 1100000000101001 0110100111110001 0011011100010000 10011100

00FC             ALU DOUBLE,SP,SP,FTOB.F & PLUS & ZA & ALUC & BASW &
                 / IOC,,TO.A & MEMR & IF C.SBR,INV,READY & NUM,MMRSP
00FC 1100000001000101 0010010111010001 0011011100010000 11000100

00FD             ALU,,E,FTOB.F & OR & DZ & ALUC & IOC & BASW & HLD
00FD 1100000000101001 0110100111110000 0011010101000110 11011111

00FE             ALU DOUBLE,SP,SP,FTOB.F & PLUS& ZA & ALUC & BASW &
                 /IOC,,TO.A & MEMR & IF C.SBR,INV,READY & NUM,MMRSP
00FE 1100000001000101 0010010111010001 0011011100010000 11000100

00FF             ALU,,D,FTOB.F & OR & DZ & ALUC & IOC & BASW & HLD
00FF 1100000000101001 0110100111110000 0011010101000100 11011111

0100             ALU DOUBLE,PC & OR & ZA & ALUC & IOC,,TO.A & BASW &
                 /NOC & IF R.F,INV,HOLD & NUM,HLDF
0100 1100000000101111 1010100111110001 0011011111110100 01011100

0101  POP.H:     ALU DOUBLE,SP,SP,FTOB.A & OR & ZA & ALUC & BASW &
                 /IOC,,TO.A & NOC & IF C.SBR,,HOLD & NUM,HLDSB
0101 1100000000101001 0110100111110001 0011011100010000 10011100

0102             ALU DOUBLE,SP,SP,FTOB.F & PLUS & ZA & ALUC & BASW &
                 / IOC,,TO.A & MEMR & IF C.SBR,INV,READY & NUM,MMRSP
0102 1100000001000101 0010010111010001 0011011100010000 11000100

0103             ALU,,L,FTOB.F & OR & DZ & ALUC & IOC & BASW & HLD
0103 1100000000101001 0110100111110000 0011010101001010 11011111

0104             ALU DOUBLE,SP,SP,FTOB.F & PLUS& ZA & ALUC & BASW &
                 /IOC,,TO.A & MEMR & IF C.SBR,INV,READY & NUM,MMRSP
0104 1100000001000101 0010010111010001 0011011100010000 11000100

0105             ALU,,H,FTOB.F & OR & DZ & ALUC & IOC & BASW & HLD
0105 1100000000101001 0110100111110000 0011010101001000 11011111

0106             ALU DOUBLE,PC & OR & ZA & ALUC & IOC,,TO.A & BASW &
                 /NOC & IF R.F,INV,HOLD & NUM,HLDF
0106 1100000000101111 1010100111110001 0011011111110100 01011100

0107  JNZ:       NALU & IOC & NOC & IF ,INV,Z & NUM,JMP
0107 1100000100001000 0000000111110000 0011010101010100 01XXXXXX


0108             INCPC & HLD
```

```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE
0108 1100000000101001 0110100111110001 0011011111111110 11000100

0109              INCPC & IF R.F,INV,HOLD & NUM,HLDF & NOC
0109 1100000000101111 1010100111110001 0011011111111110 11000100

010A  CNZ:      NALU & IOC & NOC & IF ,INV,Z & NUM,CALL
010A 1100000100011100 0000000111110000 0011010101010100 01XXXXXX

010B                INCPC & HLD
010B 1100000000101001 0110100111110001 0011011111111110 11000100

010C                INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
010C 1100000000101111 1010100111110001 0011011111111110 11000100

010D  RNZ:      NALU & IOC & NOC & IF R.F,,Z & NUM,RET
010D 1100000101000011 1100000111110000 0011010101010100 01XXXXXX

010E  JZ:       NALU & IOC & NOC & IF , ,Z & NUM,JMP
010E 1100000100001000 0100000111110000 0011010101010100 01XXXXXX

010F                INCPC & HLD
010F 1100000000101001 0110100111110001 0011011111111110 11000100

0110                INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0110 1100000000101111 1010100111110001 0011011111111110 11000100

0111  CZ:       NALU & IOC & NOC & IF , ,Z & NUM,CALL
0111 1100000100011100 0100000111110000 0011010101010100 01XXXXXX

0112                INCPC & HLD
0112 1100000000101001 0110100111110001 0011011111111110 11000100

0113                INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0113 1100000000101111 1010100111110001 0011011111111110 11000100

0114  RZ:       NALU & IOC & NOC & IF R.F,INV,Z & NUM,RET
0114 1100000101000011 1000000111110000 0011010101010100 01XXXXXX

0115  JNC:      NALU & IOC & NOC & IF ,INV,CY & NUM,JMP
0115 1100000100001000 0000010111110000 0011010101010100 01XXXXXX

0116              INCPC &HLD
0116 1100000000101001 0110100111110001 0011011111111110 11000100

0117              INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0117 1100000000101111 1010100111110001 0011011111111110 11000100

0118  CNC:      NALU & IOC & NOC & IF ,INV,CY & NUM,CALL
0118 1100000100011100 0000010111110000 0011010101010100 01XXXXXX

0119              INCPC & HLD
0119 1100000000101001 0110100111110001 0011011111111110 11000100

011A              INCPC & NOC & IF R.F ,INV,HOLD & NUM,HLDF
011A 1100000000101111 1010100111110001 0011011111111110 11000100

011B  RNC:      NALU & IOC & NOC & IF R.F,,CY & NUM,RET
011B 1100000101000011 1100010111110000 0011010101010100 01XXXXXX

011C  JC:       NALU & IOC & NOC & IF ,,CY & NUM,JMP
```



```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE
011C 1100000100001000 0100010111110000 0011010101010100 01XXXXXX

011D              INCPC & HLD
011D 1100000000101001 0110100111110001 0011011111111110 11000100

011E              INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
011E 1100000000101111 1010100111110001 0011011111111110 11000100

011F  CC:       NALU & IOC & NOC & IF ,,CY & NUM,CALL
011F 1100000100011100 0100010111110000 0011010101010100 01XXXXXX

0120              INCPC & HLD
0120 1100000000101001 0110100111110001 0011011111111110 11000100

0121              INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0121 1100000000101111 1010100111110001 0011011111111110 11000100

0122  RC:       NALU & IOC & NOC & IF R.F,INV,CY & NUM,RET
0122 1100000101000011 1000010111110000 0011010101010100 01XXXXXX

0123  JPO:      NALU & IOC & NOC & IF ,INV,P & NUM,JMP
0123 1100000100001000 0000100111110000 0011010101010100 01XXXXXX

0124              INCPC & HLD
0124 1100000000101001 0110100111110001 0011011111111110 11000100

0125              INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0125 1100000000101111 1010100111110001 0011011111111110 11000100

0126  CPO:      NALU & IOC & NOC & IF ,INV,P & NUM,CALL
0126 1100000100011100 0000100111110000 0011010101010100 01XXXXXX

0127              INCPC & HLD
0127 1100000000101001 0110100111110001 0011011111111110 11000100

0128              INCPC & NOC & IF R.F,INV,HOLD& NUM,HLDF
0128 1100000000101111 1010100111110001 0011011111111110 11000100

0129  RPO:      NALU & IOC & NOC & IF R.F,,P & NUM,RET
0129 1100000101000011 1100100111110000 0011010101010100 01XXXXXX

012A  JPE:      NALU & IOC & NOC & IF ,,P & NUM,JMP
012A 1100000100001000 0100100111110000 0011010101010100 01XXXXXX

012B              INCPC & HLD
012B 1100000000101001 0110100111110001 0011011111111110 11000100

012C              INCPC & NOC &IF R.F,INV,HOLD & NUM,HLDF
012C 1100000000101111 1010100111110001 0011011111111110 11000100

012D  CPE:      NALU & IOC & NOC & IF ,,P & NUM,CALL
012D 1100000100011100 0100100111110000 0011010101010100 01XXXXXX

012E              INCPC & HLD
012E 1100000000101001 0110100111110001 0011011111111110 11000100

012F              INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
012F 1100000000101111 1010100111110001 0011011111111110 11000100

0130  RPE:      NALU & IOC & NOC & IF R.F,INV,P & NUM,RET
```

```
PC    SOURCE AND/OR OBJECT CODE.   X = DON'T CARE

0130 1100000101000011 1000100111110000 0011010101010100 01XXXXXX

0131  JP:      NALU & IOC & NOC & IF ,INV,S & NUM,JMP
0131 1100000100001000 0000110111110000 0011010101010100 01XXXXXX

0132            INCPC & HLD
0132 1100000000101001 0110100111110001 0011011111111110 11000100

0133            INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0133 1100000000101111 1010100111110001 0011011111111110 11000100

0134  CP:      NALU & IOC & NOC & IF R.F,,S & NUM,CALL
0134 1100000100011111 1100110111110000 0011010101010100 01XXXXXX

0135            INCPC & HLD
0135 1100000000101001 0110100111110001 0011011111111110 11000100

0136            INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
0136 1100000000101111 1010100111110001 0011011111111110 11000100

0137  RP:      NALU & IOC & NOC & IF ,INV,S & NUM,RET
0137 1100000101000000 0000110111110000 0011010101010100 01XXXXXX

0138  JM:      NALU & IOC & NOC & IF ,,S & NUM,JMP
0138 1100000100001000 0100110111110000 0011010101010100 01XXXXXX

0139            INCPC & HLD
0139 1100000000101001 0110100111110001 0011011111111110 11000100

013A             INCPC & NOC & IF R.F,INV,HOLD & NUM,HLDF
013A 1100000000101111 1010100111110001 0011011111111110 11000100

013B  CM:      NALU & IOC & NOC & IF ,,S & NUM,CALL
013B 1100000100011100 0100110111110000 0011010101010100 01XXXXXX

013C       INCPC & HLD
013C 1100000000101001 0110100111110001 0011011111111110 11000100

013D            INCPC  & NOC & IF R.F,INV,HOLD & NUM,HLDF
013D 1100000000101111 1010100111110001 0011011111111110 11000100

013E  RM:      NALU & IOC & NOC & IF R.F,INV,S & NUM,RET
013E 1100000101000011 1000110111110000 0011010101010100 01XXXXXX

013F  DAA:    NALU & IOC & HLD
013F 1100000000101001 0110100111110000 0011010101010100 01XXXXXX

0140            NALU & IOC & IF,INV & NUM DBUS,006 & NOC
0140 1000000000011000 0011110111110000 0011010101010100 01XXXXXX

0141            ALU,,,FTOB.F & OR & DZ & ALUC & BASW & IOC &
                /NOC & IF,,AC & NUM,DAA1
0141 1100010100011000 0101000111110000 0011010101010100 11011111

0142            NALU & IOC & IF,INV & NUM DBUS,00F & NOC
0142 1000000000111100 0011110111110000 0011010101010100 01XXXXXX

0143            ALU,A,,FTOQ & AND & DA & ALUC & BASW & IOC &
                /NOC & IF,INV & NUM DBUS,00A
0143 1000000000101000 0011110111110000 0011010011110100 00100101
```



```
PC    SOURCE AND/OR OBJECT CODE.   X = DON'T CARE

0144            ALU & DQ & SUNIM & ALUC & BASW & IOC & HLD
0144 1100000000101001 0110100111110000 0011010101010100 01001110

0145            NALU & IOC & NOC & IF,INV,CN.4 & NUM,$+2
0145 1100010100011100 0011100111110000 0011010101010100 01XXXXXX

0146  DAA1:     ALU,,A,FTOB.F & PLUS & AB & ALUC UPDTALL,,CNL &
                /IOC & BASW & NOC & IF C.SBR,,CY & NUM,DAA4
0146 1100010101111101 0100010111110000 0000000101001110 11000001

0147            NALU & IOC & IF,INV & NUM DBUS,060 & NOC
0147 1000000110000000 0011110111110000 0011010101010100 01XXXXXX


0148            ALU,,,FTOB.F & OR & DZ & ALUC & BASW & IOC &
                /NOC & IF,,CY & NUM,DAA3
0148 1100010100111100 0100010111110000 0011010101010100 11011111

0149            NALU & IOC & IF,INV & NUM DBUS,0F0 & NOC
0149 1000001111000000 0011110111110000 0011010101010100 01XXXXXX

014A            ALU,A,,FTOQ & AND & DA & ALUC & BASW & IOC &
                /NOC & IF,INV & NUM DBUS,0A0
014A 1000001010000000 0011110111110000 0011010011110100 00100101

014B            ALU & DQ & SUNIM & ALUC & BASW & IOC & HLD
014B 1100000000101001 0110100111110000 0011010101010100 01001110

014C            NALU & IOC & NOC & IF,INV,CN.4 & NUM,$+2
014C 1100010100111000 0011100111110000 0011010101010100 01XXXXXX

014D  DAA2:     ALU,,A,FTOB.F & PLUS & AB & ALUC UPDTALL ,,CNL &
                /BASW & IOC & HLD
014D 1100000000101001 0110100111110000 0000000101001110 11000001

014E            NALU & IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
014E 1100000000101111 1010100111110000 0011010101010100 01XXXXXX

014F  DAA3:    ALU,,A,FTOB.F & PLUS & AB & ALUC UPDTFL,,CNL &
               /BASW & IOC & NOC & IF R.F,INV,HOLD & NUM,HLDF
014F 1100000000101111 1010100111110000 0010000101001110 11000001

0150  POP.PSW: ALU DOUBLE,SP,SP,FTOB.A & OR & ZA & ALUC & BASW & IOC,,T
          O.A & /NOC & IF C.SBR,,HOLD & NUM,HLDSB
0150 1100000000101001 0110100111110001 0011011100010000 10011100

0151            ALU DOUBLE,SP,SP,FTOB.F & PLUS & ZA & ALUC & BASW & MEMR


              & /IOC,,TO.A & IF C.SBR,INV,READY & NUM,MMRSP
0151 1100000001000101 0010010111010001 0011011100010000 11000100

0152            ALU & NAND & ALUC & IOC & BASW & HLD
0152 1100000000101001 0110100111110000 0011010101010100 01101XXX

0153            ALU DOUBLE,SP,SP,FTOB.F & PLUS & ZA & ALUC & BASW &
               /IOC,,TO.A & MEMR & IF C.SBR,INV,READY & NUM,MMRSP
0153 1100000001000101 0010010111010001 0011011100010000 11000100

0154          ALU,,A,FTOB.F & OR & DZ & ALUC & BASW & IOC &  HLD
```

```
PC    SOURCE AND/OR OBJECT CODE.    X = DON'T CARE
0154 1100000000101001 0110100111110000 0011010101001110 11011111

0155          ALU DOUBLE,PC & DR & ZA & ALUC & IOC,,TO.A & BASW & NOC &


              /IF R.F,INV,HOLD & NUM,HLDF
0155 1100000000101111 1010100111110001 0011011111110100 01011100

0156  LDAX.D: ALU DOUBLE,D & BASW & ZA & DR & IOC,,TO.A & ALUC & HLD
0156 1100000000101001 0110100111110001 0011011001010100 01011100

0157            ALU DOUBLE,PC & DR & ZA & IOC,,TO.A & ALUC & BASW &
                /MEMR & IF,INV,READY & NUM, $
0157 1100010101011100 0010010111010001 0011011111110100 01011100

0158            ALU,,A,FTOB.F & DR & DZ & ALUC & BASW & IOC &
                /NOC & IF R.F,INV,HOLD & NUM,HLDF
0158 1100000000101111 1010100111110000 0011010101001110 11011111

0159  XCHG:     ALU DOUBLE,D,L,FTOB.A & DR & DZ & ALUC,SWAP & BASW & IOC

    & HLD
0159 1100000000101001 0110100111110000 0011111001001010 10011111

015A            ALU DOUBLE,H,D,FTOB.A & DR & ZA & ALUC & BASW & IOC & HL
                D

015A 1100000000101001 0110100111110000 0011011010000100 10011100

015B            ALU DOUBLE,L,H,FTOB.A & DR & DZ & ALUC,SWAP & BASW & IOC

    & HLD
015B 1100000000101001 0110100111110000 0011111010101000 10011111

015C      ALU DOUBLE,D,E,FTOB.A & ALUC,SWAP & BASW & IOC &
                /NOC & IF R.F,INV,HOLD & NUM,HLDF & DR & DZ
015C 1100000000101111 1010100111110000 0011111001000110 10011111

015D  SBI1:     INCPC & MMR
015D 1100000000110101 0010010111010001 0011011111111110 11000100

015E            ALU,A,A,FTOB.F & DA & SUNIM & ALUC UPDTALL,,CNL & IOC &


                /BASW & NOC & IF R.F,INV,HOLD & NUM,HLDF
015E 1100000000101111 1010100111110000 0000000011101110 11001101

015F  DAA4:     ALU,H#C & SUNIM & ZA & ALUC UPDTCY,,CNL & BASW &
                /IOC & NOC & IF R.RTN & NUM
015F 1100000000000001 1111110111110000 0001000110010100 01001100

0000  ORG H#3FF

03FF  INTRPT:  NALU & IOC & NOC & IF & NUM,INTHNDL
03FF 1100001000010000 0111110111110000 0011010101010100 01XXXXXX

0400  END
```

ADVANCED
MICRO
DEVICES, INC.
901 Thompson Place
Sunnyvale
California 94086
(408) 732-2400
TWX: 910-339-9280
TELEX: 34-6306
TOLL FREE
(800) 538-8450