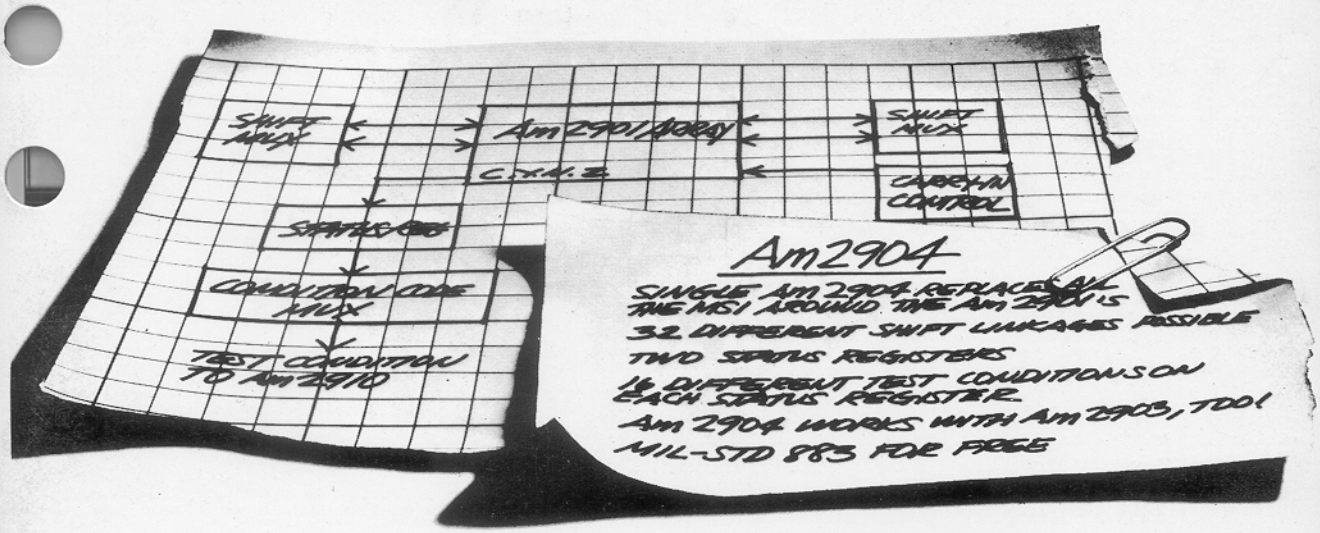


Build A Microcomputer

Chapter IV
The Data Path — Part II

Advanced Micro Devices

Advanced Micro Devices

Build A Microcomputer

Chapter IV
The Data Path – Part II

AM-PUB073-4

CHAPTER IV
THE DATA PATH

The previous CPU example (See Chapter III) utilized SSI and MSI components to accomplish the shift-linkage, carry control, and status register functions associated with the ALU. These functions can all be implemented with the Am2904 status and shift control unit.

The Am2904 is an LSI device that contains all the logic necessary to perform the shift and status control operations associated with the ALU portion of a microcomputer. These operations include storage for ALU status flags; carry-in generation and selection; data-path, carry bit linkage for shift/rotate instructions; and status condition code generation and selection. The ALU status flags: carry, zero, negative, and overflow; may be stored in either of two registers, a machine status register or a micro status register. The carry-in multiplexer can select the true or complement of the microstatus carry flag or machine status carry flag, as well as an external carry, a logical one, or a logical zero. The shift linkage multiplexers provide paths to rotate/shift single and double length words up, down, around the carry flag, and through the carry flag. The status condition code multiplexer provides tests on the true or complement of any status flag, as well as more complicated logical combinations of these flags to facilitate magnitude comparisons on unsigned and two's complement numbers, and normalization operations.

STATUS REGISTERS

The status registers contained in the Am2904 are shown in the upper portion of Figure 1. Each register is independently controlled by a combination of instruction signals and enable signals.

MICRO STATUS REGISTER (μSR)

The μSR is enabled when the $\overline{CE}\mu$ signal is low. When $\overline{CE}\mu$ is low the instruction present on I_5 through I_0 will be executed on the LOW to HIGH transition of the Clock input. These instructions fall into three main categories: Bit Operations, Register Operations and Load Operations.

The bit operations allow individual bits of the μSR to be set or reset. (See Table 1.1).

The register operations allow the μSR to be loaded from the machine status register, to be set to all one's, reset to all zero's, or swapped with the machine status register. (See Table 1.2).

The load operations allow the μSR to be loaded from the I inputs directly, from the I inputs with I_C complemented, or from the I inputs with overflow retained, $I_{OVR} + \mu_{OVR} \rightarrow \mu_{OVR}$ (See Table 1.3). The load operation with I_C complemented can be used to emulate machines which use direct subtraction and thus need to complement the carry to obtain a borrow. The load with overflow retained allows a series of arithmetic instructions to be executed without the need for a check for overflow after each instruction. If an overflow occurred at any time during the series it will be "trapped." Thus a single test for overflow, at the end of the series, is all that is required.

MACHINE STATUS REGISTER (MSR)

The MSR is enabled when $\overline{CE}_M$ is low. If $\overline{CE}_M$ is low the instruction present on I_5 through I_0 will be executed on the LOW to HIGH transition of the Clock input. Additionally the individual bits of the MSR may be selectively enabled through the use of the Enable inputs $\overline{E}_Z$, $\overline{E}_C$, $\overline{E}_N$ and $\overline{E}_{OVR}$ (See Figure 1). This allows all possible combinations of the four status flags to be selectively operated on for maximum flexibility. Thus the instruction specified by I_5-I_0 only effect the enabled status flags.

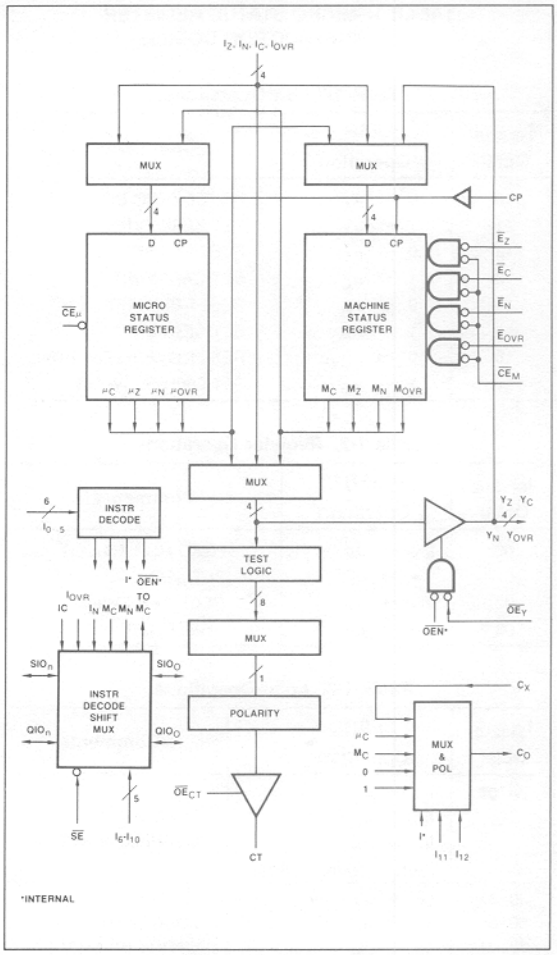


Figure 1. Am2904 Block Diagram.

The MSR instructions fall into two main categories: register operations and load operations (bit operations can be implemented through the use of the selective enable control lines).

The register operations allow the MSR to be loaded from the bi-directional Y port, or the μSR. Additionally the MSR may be set, reset, or complemented (See Table 2.1). These three instructions, combined with the selective enables, allow any combination of MSR bits to be set, reset, or complemented.

The load operations allow the MSR to be loaded directly from the I inputs, from the I inputs with I_C complemented, or from the I inputs for shift through overflow (See Table 2.2). The load with I_C complemented can be used to produce a borrow. The load for shift through overflow loads the zero flag and the negative flag from the I inputs while swapping the overflow and carry flags. This allows the shift through overflow operation to be easily implemented.

SHIFT LINKAGE MULTIPLEXERS

The shift linkage multiplexers control bi-directional shift lines SIOn, SIO_0 (RAM shifter on the Am2903) and QIOn, QIO_0 (Q register shifter on the Am2903). To enable the shift linkage multiplexers the shift enable line $\overline{SE}$ must be low. When $\overline{SE}$ is low the

TABLE 1. MICRO STATUS REGISTER INSTRUCTION CODES.

Table 1-1. Bit Operations.

I_{543210} Octal	μSR Operation	Comments
10	$0 \rightarrow \mu_Z$	RESET ZERO BIT
11	$1 \rightarrow \mu_Z$	SET ZERO BIT
12	$0 \rightarrow \mu_C$	RESET CARRY BIT
13	$1 \rightarrow \mu_C$	SET CARRY BIT
14	$0 \rightarrow \mu_N$	RESET SIGN BIT
15	$1 \rightarrow \mu_N$	SET SIGN BIT
16	$0 \rightarrow \mu_{OVR}$	RESET OVERFLOW BIT
17	$1 \rightarrow \mu_{OVR}$	SET OVERFLOW BIT

Table 1-2. Register Operations.

I_{543210} Octal	μSR Operation	Comments
00	$M_X \rightarrow \mu_X$	LOAD MSR TO μSR
01	$1 \rightarrow \mu_X$	SET μSR
02	$M_X \rightarrow \mu_X$	REGISTER SWAP
03	$0 \rightarrow \mu_X$	RESET μSR

Table 1-3. Load Operations.

I_{543210} Octal	μSR Operation	Comments
06, 07	$I_Z \rightarrow \mu_Z$ $I_C \rightarrow \mu_C$ $I_N \rightarrow \mu_N$ $I_{OVR} + \mu_{OVR} \rightarrow \mu_{OVR}$	LOAD WITH OVERFLOW RETAIN
30, 31 50, 51 70, 71	$I_Z \rightarrow \mu_Z$ $I_C \rightarrow \mu_C$ $I_N \rightarrow \mu_N$ $I_{OVR} \rightarrow \mu_{OVR}$	LOAD WITH CARRY INVERT
04, 05 20-27 32-47 52-67 72-77	$I_Z \rightarrow \mu_Z$ $I_C \rightarrow \mu_C$ $I_N \rightarrow \mu_N$ $I_{OVR} \rightarrow \mu_{OVR}$	LOAD DIRECTLY FROM I_Z, I_C, I_N, I_{OVR}

Note: The above tables assume $\overline{CE}$ is LOW.

shift linkage data path will be set-up depending on the state of instruction lines I_{10} through I_6 (See Table 3). These instructions allow single length or double length shifts/rotates either up, or down. Additionally shifts/rotates may be done through or around the MSR carry and negative flag. Special operations exist to provide support for add and shift (multiply) instructions. These instructions select the present carry I_C (for unsigned multiply), or the Exclusive-OR of the sign flag I_n with the overflow flag I_{OVR} (for two's complement multiplication).

CONDITION CODE MULTIPLEXER

The condition code multiplier selects one of sixteen possible logical combinations of the μSR, MSR or I inputs, depending on the state of the I_5-I_0 input lines. These combinations include the true or complement form of any individual bit in the μSR, MSR or I inputs. Additionally several more complicated logical operations may be performed to provide magnitude tests on both two's complement numbers and unsigned numbers. Table 5 lists the conditional test outputs (CT) corresponding to the state of the I_5-I_0 instruction lines. Table 6 lists the possible relations between two unsigned or two's complement numbers and the corresponding status and instruction codes. The three-state conditional test output CT is active only if $\overline{OE}_{CT}$ is low.

CARRY IN MULTIPLEXER

The Carry output can be selected from one of seven different sources depending on the state of instruction input lines. The seven possible sources are: logical zero, logical one, the μSR carry flag, the complement of the μSR carry flag, the MSR carry flag, the complement of the MSR carry flag, or the external carry input C_X (See Table 4).

TABLE 2. MACHINE STATUS REGISTER INSTRUCTION CODES.

Table 2-1. Register Operations.

I_{543210} Octal	MSR Operation	Comments
00	$Y_X \rightarrow M_X$	LOAD Y_Z, Y_C, Y_N, Y_{OVR} TO MSR
01	$1 \rightarrow M_X$	SET MSR
02	$\mu_X \rightarrow M_X$	REGISTER SWAP
03	$0 \rightarrow M_X$	RESET MSR
05	$\overline{M_X} \rightarrow M_X$	INVERT MSR

Table 2-2. Load Operations.

I_{543210} Octal	MSR Operation	Comments
04	$I_Z \rightarrow M_Z$ $M_{OVR} \rightarrow M_C$ $I_N \rightarrow M_N$ $M_C \rightarrow M_{OVR}$	LOAD FOR SHIFT THROUGH OVERFLOW OPERATION
10, 11 30, 31 50, 51 70, 71	$I_Z \rightarrow M_Z$ $I_C \rightarrow M_C$ $I_N \rightarrow M_N$ $I_{OVR} \rightarrow M_{OVR}$	LOAD WITH CARRY INVERT
06, 07 12-17 20-27 32-37 40-47 52-67 72-77	$I_Z \rightarrow M_Z$ $I_C \rightarrow M_C$ $I_N \rightarrow M_N$ $I_{OVR} \rightarrow M_{OVR}$	LOAD DIRECTLY FROM I_Z, I_C I_N, I_{OVR}

Note: 1. The above tables assume $\overline{CE}_M$, $\overline{E}_Z$, $\overline{E}_C$, $\overline{E}_N$, $\overline{E}_{OVR}$ are LOW.

Y INPUT/OUTPUT LINES

The bi-directional Y data lines may be used for extra data input lines when the Y output buffer is disabled ($\overline{OE}_Y$ high). Additionally, when I_5-I_0 are low, the Y buffer is disabled, irrespective of the $\overline{OE}_Y$ signal. When the Y buffer is enabled ($\overline{OE}_Y$ is low) the Y data lines are selected from the MSR, μSR, or I input lines depending on the state of instruction lines I_5 and I_4 (See Table 7).

TABLE 3. SHIFT LINKAGE MULTIPLEXER INSTRUCTION CODES.

I_{10}	I_9	I_8	I_7	I_6	M_C	RAM	Q	SIO_o	SIO_n	QIO_o	QIO_n	Loaded into M_C
0	0	0	0	0		MSB LSB	MSB LSB	Z	0	Z	0	
0	0	0	0	1				Z	1	Z	1	
0	0	0	1	0				Z	0	Z	M_N	SIO_o
0	0	0	1	1				Z	1	Z	SIO_o	
0	0	1	0	0				Z	M_C	Z	SIO_o	
0	0	1	0	1				Z	M_N	Z	SIO_o	
0	0	1	1	0				Z	0	Z	SIO_o	
0	0	1	1	1				Z	0	Z	SIO_o	QIO_o
0	1	0	0	0				Z	SIO_o	Z	QIO_o	SIO_o
0	1	0	0	1				Z	M_C	Z	QIO_o	SIO_o
0	1	0	1	0				Z	SIO_o	Z	QIO_o	
0	1	0	1	1				Z	I_C	Z	SIO_o	
0	1	1	0	0				Z	M_C	Z	SIO_o	QIO_o
0	1	1	0	1				Z	QIO_o	Z	SIO_o	QIO_o
0	1	1	1	0				Z	$I_N \oplus I_{OVR}$	Z	SIO_o	
0	1	1	1	1				Z	QIO_o	Z	SIO_o	
1	0	0	0	0		MSB LSB	MSB LSB	0	Z	0	Z	SIO_n
1	0	0	0	1				1	Z	1	Z	SIO_n
1	0	0	1	0				0	Z	0	Z	
1	0	0	1	1				1	Z	1	Z	
1	0	1	0	0				QIO_n	Z	0	Z	SIO_n
1	0	1	0	1				QIO_n	Z	1	Z	SIO_n
1	0	1	1	0				QIO_n	Z	0	Z	
1	0	1	1	1				QIO_n	Z	1	Z	
1	1	0	0	0				SIO_n	Z	QIO_n	Z	SIO_n
1	1	0	0	1				M_C	Z	QIO_n	Z	SIO_n
1	1	0	1	0				SIO_n	Z	QIO_n	Z	
1	1	0	1	1				M_C	Z	0	Z	
1	1	1	0	0				Q!O_n	Z	M_C	Z	SIO_n
1	1	1	0	1				QIO_n	Z	SIO_n	Z	SIO_n
1	1	1	1	0				QIO_n	Z	M_C	Z	
1	1	1	1	1				QIO_n	Z	SIO_n	Z	

Notes: 1. Z = High impedance (outputs off) state.
2. Outputs enabled and M_C loaded only if $\overline{SE}$ is LOW.
3. Loading of M_C from I_{10-6} overrides control from I_{5-0}, $\overline{CE}_M$, $\overline{E}_C$.

TABLE 4. CARRY-IN CONTROL MULTIPLEXER INSTRUCTION CODES.

I_{12}	I_{11}	I_5	I_3	I_2	I_1	C_0
0	0	X	X	X	X	0
0	1	X	X	X	X	1
1	0	X	X	X	X	C_X
1	1	0	0	X	X	μ_C
1	1	0	X	1	X	μ_C
1	1	0	X	X	1	μ_C
1	1	0	1	0	0	$\overline{\mu}_C$
1	1	1	0	X	X	M_C
1	1	1	X	1	X	M_C
1	1	1	X	X	1	M_C
1	1	1	1	0	0	$\overline{M}_C$

TABLE 5. CONDITION CODE OUTPUT (CT) INSTRUCTION CODES.

I_{3-0} HEX	I_3	I_2	I_1	I_0	$I_5 = I_4 = 0$	$I_5 = 0, I_4 = 1$	$I_5 = 1, I_4 = 0$	$I_5 = I_4 = 1$
0	0	0	0	0	$(\mu_N \oplus \mu_{OVR}) + \mu_Z$	$(\mu_N \oplus \mu_{OVR}) + \mu_Z$	$(M_N \oplus M_{OVR}) + M_Z$	$(I_N \oplus I_{OVR}) + I_Z$
1	0	0	0	1	$(\mu_N \odot \mu_{OVR}) \cdot \overline{\mu}_Z$	$(\mu_N \odot \mu_{OVR}) \cdot \overline{\mu}_Z$	$(M_N \odot M_{OVR}) \cdot \overline{M}_Z$	$(I_N \odot I_{OVR}) \cdot \overline{I}_Z$
2	0	0	1	0	$\mu_N \oplus \mu_{OVR}$	$\mu_N \oplus \mu_{OVR}$	$M_N \oplus M_{OVR}$	$I_N \oplus I_{OVR}$
3	0	0	1	1	$\mu_N \odot \mu_{OVR}$	$\mu_N \odot \mu_{OVR}$	$M_N \odot M_{OVR}$	$I_N \odot I_{OVR}$
4	0	1	0	0	μ_Z	μ_Z	M_Z	I_Z
5	0	1	0	1	$\overline{\mu}_Z$	$\overline{\mu}_Z$	$\overline{M}_Z$	$\overline{I}_Z$
6	0	1	1	0	μ_{OVR}	μ_{OVR}	M_{OVR}	I_{OVR}
7	0	1	1	1	$\overline{\mu}_{OVR}$	$\overline{\mu}_{OVR}$	$\overline{M}_{OVR}$	$\overline{I}_{OVR}$
8	1	0	0	0	$\mu_C + \mu_Z$	$\mu_C + \mu_Z$	$M_C + M_Z$	$\overline{I}_C + I_Z$
9	1	0	0	1	$\overline{\mu}_C \cdot \overline{\mu}_Z$	$\overline{\mu}_C \cdot \overline{\mu}_Z$	$\overline{M}_C \cdot \overline{M}_Z$	$I_C \cdot \overline{I}_Z$
A	1	0	1	0	μ_C	μ_C	M_C	I_C
B	1	0	1	1	$\overline{\mu}_C$	$\overline{\mu}_C$	$\overline{M}_C$	$\overline{I}_C$
C	1	1	0	0	$\overline{\mu}_C + \mu_Z$	$\overline{\mu}_C + \mu_Z$	$\overline{M}_C + M_Z$	$\overline{I}_C + I_Z$
D	1	1	0	1	$\mu_C \cdot \overline{\mu}_Z$	$\mu_C \cdot \overline{\mu}_Z$	$M_C \cdot \overline{M}_Z$	$I_C \cdot \overline{I}_Z$
E	1	1	1	0	$I_N \oplus M_N$	μ_N	M_N	I_N
F	1	1	1	1	$I_N \odot M_N$	$\overline{\mu}_N$	$\overline{M}_N$	$\overline{I}_N$

Notes: 1. $\oplus$ Represents EXCLUSIVE-OR $\odot$ Represents EXCLUSIVE-NOR or coincidence.

TABLE 6. CRITERIA FOR COMPARING TWO NUMBERS FOLLOWING "A MINUS B" OPERATIONS.

	For Unsigned Numbers			For 2's Complement Numbers		
		I_{3-0}			I_{3-0}	
Relation	Status	CT = H	CT = L	Status	CT = H	CT = L
A = B	Z = 1	4	5	Z = 1	4	5
A = B	Z = 0	5	4	Z = 0	5	4
A $\geq$ B	C = 1	A	B	N $\odot$ OVR = 1	3	2
A < B	C = 0	B	A	N $\oplus$ OVR = 1	2	3
A > B	C $\cdot \overline{Z}$ = 1	D	C	(N $\odot$ OVR) $\cdot \overline{Z}$ = 1	1	0
A $\leq$ B	$\overline{C}$ + Z = 1	C	D	(N $\oplus$ OVR) + Z = 1	0	1

$\oplus$ = Exclusive OR H = HIGH Note: For Am2910, the CC input is active LOW, so use I_{3-0} code to produce
$\odot$ = Exclusive NOR L = LOW CT = L for the desired test.

TABLE 7. Y OUTPUT INSTRUCTION CODES.

$\overline{OE_Y}$	I_5	I_4	Y Output	Comment
1	X	X	Z	Output Off High Impedance
O	O	X	$\mu_i \rightarrow Y_i$	See Note 1
O	1	O	$M_i \rightarrow Y_i$	
O	1	1	$I_i \rightarrow Y_i$	

Notes: 1. For the conditions:
I_5, I_4, I_3, I_2, I_1, I_0 are LOW, Y is an input.
$\overline{OE_Y}$ is "Don't Care" for this condition.
2. X is "Don't Care" condition.

TIMING ANALYSIS

In the previous chapter a timing analysis was presented with the shift-linkage, carry-control, and status registers implemented in SSI and MSI. This timing analysis will be repeated with the SSI and MSI logic replaced with the Am2904. Tables 8.1, 8.2, 8.4 and 8.5 list the typical AC characteristics of the registers, Am2902A, Am2901A, Am2903, and Am2904 used in these calculations. Table 8.3 lists the assumed AC characteristics for the set-up time of the Am2904.

Figure 2 illustrates the timing analysis for an Am2901A based design. The analysis begins with the LOW to HIGH transition of the system clock. All signals must be valid for the next LOW to HIGH transition of the system clock, i.e. one-microcycle later.

Figure 3 illustrates a similar timing analysis for the Am2903. The results of both analysis are listed in Table 9.

USING THE Am2904 IN A 16-BIT DESIGN

Perhaps the best technique for understanding the Am2904 is to simply compare 16-bit ALU designs with and without the Am2904. The first design, Figure 4a, is an example of a 16-bit CPU design using SSI/MSI parts instead of the Am2904. In Figure 4b, the second 16-bit CPU design, the Am2904 is shown replacing the SSI/MSI. The Am2904 substitutes for the appropriate shift matrix control and status registers. A more detailed comparison may be obtained by referring to the 16-bit ALU designs in Chapter III and the one in Appendix C of this chapter. To understand the Am2904 further, the usage of the Am2904 is described through the microprogram bits in the microprogram structure and shown later in the actual microprograms.

TABLE 8-1. STANDARD DEVICE SCHOTTKY SPEEDS.

Device and Path	Min.	Typ.	Max.
S-REGISTER			
Clock to Output		9	15
$\overline{OE}$ to Output		13	20
Set-up	5	2	
Am2902A			
Cn to Cn+x, Y, Z		7	11
G, P to G, P		7	10
G, P to Cn+x, Y, Z		5	7

TABLE 8-2.
PRELIMINARY SWITCHING CHARACTERISTICS.

Combinational Delays (ns)

From (Input)	To (Output)	t_{pd}
I_Z I_C I_N I_{OVR}	Y_Z Y_C Y_N Y_{OVR}	20
CP	Y_Z, Y_C, Y_N, Y_{OVR}	30
I_4, I_5	Y_Z, Y_C, Y_N, Y_{OVR}	23
I_Z, I_C, I_N, I_{OVR}	CT	30
CP	CT	30
I_0-I_5	CT	30
C_X	C_O	12
CP	C_O	20
$I_{1,2,3,5,11,12}$	C_O	24
SIO_n, QIO_n	SIO_o	16
SIO_o, QIO_o	SIO_n	16
I_C, I_N, I_{OVR}	SIO_n	20
SIO_n, QIO_n	QIO_o	16
SIO_o, QIO_o	QIO_n	16
CP	SIO_o, SIO_n QIO_o, QIO_n	21
I_6-I_{10}	SIO_o, SIO_n QIO_o, QIO_n	19

TABLE 8-3. ASSUMED SET-UP TIME.*

Input	TS
IOVR, IZ, IN, IC	20ns

*The actual set-up times where not available at the time this was written. See current data sheets for correct timing on these signals.

TABLE 8-4.

Am2901A — (MAY 18, 1978)

ROOM TEMPERATURE SWITCHING CHARACTERISTICS

Tables I, II, and III below define the timing characteristics of the Am2901A at 25°C. The tables are divided into three types of parameters; clock characteristics, combinational delays from inputs to outputs, and set-up and hold time requirements. The latter table defines the time prior to the end of the cycle (i.e., clock LOW-to-HIGH transition) that each input must be stable to guarantee that the correct data is written into one of the internal registers.

All values are at 25°C and 5.0V. Measurements are made at 1.5V with V_{IL} = 0V and V_{IH} = 3.0V. For three-state disable tests, C_L = 5.0pF and measurement is to 0.5V change on output voltage level. All outputs fully loaded.

TABLE I

CYCLE TIME AND CLOCK CHARACTERISTICS

TIME	TYPICAL	GUARANTEED
Read-Modify-Write Cycle (time from selection of A, B registers to end of cycle)	55ns	93ns
Maximum Clock Frequency to Shift Q Register (50% duty cycle)	40MHz	20MHz
Minimum Clock LOW Time	30ns	30ns
Minimum Clock HIGH Time	30ns	30ns
Minimum Clock Period	75ns	93ns

TABLE II

COMBINATIONAL PROPAGATION DELAYS (all in ns, C_L = 50pF (except output disable tests))

From Input \ To Output	TYPICAL 25°C, 5.0V								GUARANTEED 25°C, 5.0V							
	Y	F_3	C_{n+4}	$\overline{G}, \overline{P}$	F=0 R_L= 270	OVR	Shift Outputs		Y	F_3	C_{n+4}	$\overline{G}, \overline{P}$	F=0 R_L= 270	OVR	Shift Outputs	
							RAM_0 RAM_3	Q_0 Q_3							RAM_0 RAM_3	Q_0 Q_3
A, B	45	45	45	40	65	50	60	–	75	75	70	59	85	76	90	–
D (arithmetic mode)	30	30	30	25	45	30	40	–	39	37	41	31	55	45	59	–
D (I = X37) (Note 5)	30	30	–	–	45	–	40	–	36	34	–	–	51	–	53	–
C_n	20	20	10	–	35	20	30	–	27	24	20	–	46	26	45	–
I_{012}	35	35	35	25	50	40	45	–	50	50	46	41	65	57	70	–
I_{345}	35	35	35	25	45	35	45	–	50	50	50	42	65	59	70	–
I_{678}	15	–	–	–	–	–	20	20	26	–	–	–	–	–	26	26
$\overline{OE}$ Enable/Disable	20/20	–	–	–	–	–	–	–	30/33	–	–	–	–	–	–	–
A bypassing ALU (I = 2xx)	30	–	–	–	–	–	–	–	35	–	–	–	–	–	–	–
Clock ↓ (Note 6)	40	40	40	30	55	40	55	20	52	52	52	41	70	57	71	30

TABLE III

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

From Input	Notes	TYPICAL 25°C, 5.0V		GUARANTEED 25°C, 5.0V	
		Set-Up Time	Hold Time	Set-Up Time	Hold Time
A, B Source	2, 4 3, 5	40 $t_{pw}L$ + 15	0	93 $t_{pw}L$ + 25	0
B Dest.	2, 4	$t_{pw}L$ + 15	0	$t_{pw}L$ + 15	0
D (arithmetic mode)		25	0	70	0
D (I = X37) (Note 5)		25	0	60	0
C_n		40	0	55	0
I_{012}		30	0	64	0
I_{345}		30	0	70	0
I_{678}	4	$t_{pw}L$ + 15	0	$t_{pw}L$ + 25	0
$RAM_{0, 3}$, $Q_{0, 3}$		15	0	20	0

Notes: 1. See next page.
2. If the B address is used as a source operand, allow for the "A, B source" set-up time; if it is used only for the destination address, use the "B dest." set-up time.
3. Where two numbers are shown, both must be met.
4. "$t_{pw}L$" is the clock LOW time.
5. DV 0 is the fastest way to load the RAM from the D inputs. This function is obtained with I = 337.
6. Using Q register as source operand in arithmetic mode. Clock is not normally in critical speed path when Q is not a source.

TABLE 8-5.

A. Am2903 SWITCHING CHARACTERISTICS (TYPICAL ROOM TEMPERATURE PERFORMANCE) – (MAY 18, 1978)

Tables IA, IIA, and IIIA define the nominal timing characteristics of the Am2903 at 25°C and 5.0V. The Tables divide the parameters into three types: pulse characteristics for the clock and write enable, combinational delays from input to output, and set-up and hold times relative to the clock and write pulse.

Measurements are made at 1.5V with V_{IL} = 0V and V_{IH} = 3.0V. For three-state disable tests, C_L = 5.0pF and measurement is to 0.5V change on output voltage level.

TABLE IA – Write Pulse and Clock Characteristics

Time	
Minimum Time CP and $\overline{WE}$ both LOW to write	15ns
Minimum Clock LOW Time	15ns
Minimum Clock HIGH Time	35ns

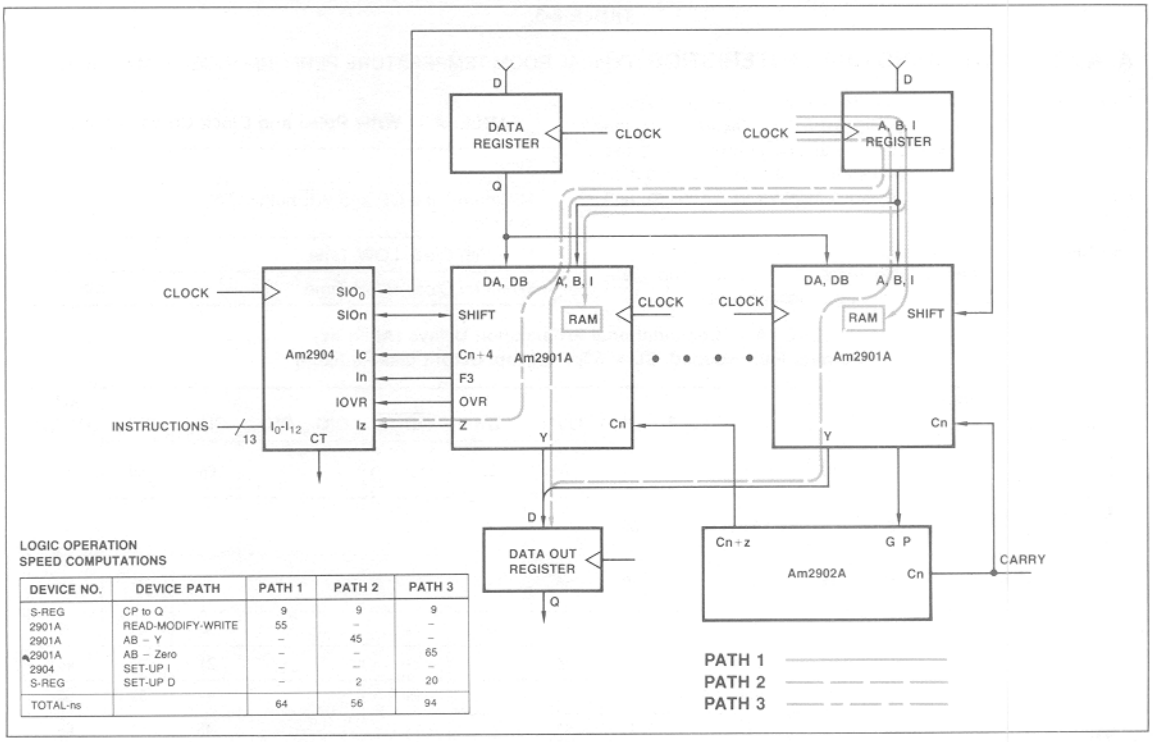
TABLE IIA – Combinational Propagation Delays (All in ns)
Outputs Fully Loaded. CL = 50pF (except output disable tests)

From Input \ To Output	Y	C_{n+4}	$\overline{G}$, $\overline{P}$	(S) Z	N	OVR	DB	$\overline{WRITE}$	QIO_0, QIO_3	SIO_0	SIO_3	SIO_0 (Parity)
A, B Addresses (Arith. Mode)	65	60	56	–	64	70	33	–	–	65	69	87
A, B Addresses (Logic Mode)	56	–	46	–	56	–	33	–	–	55	64	81
DA, DB Inputs	39	38	30	–	40	56	–	–	–	39	47	60
$\overline{EA}$	38	33	26	–	36	41	–	–	–	36	41	58
C_n	25	21	–	–	20	38	–	–	–	21	25	48
I_0	40	31	24	–	37	42	–	15(1)	–	41	39	63
I_{4321}	45	45	32	–	44	52	–	17(1)	–	45	51	68
I_{8765}	25	–	–	–	–	–	–	21	22/29(2)	24/17(2)	27/17(2)	24/17(2)
$\overline{IEN}$	–	–	–	–	–	–	–	10	–	–	–	–
$\overline{OEB}$ Enable/Disable	–	–	–	–	–	–	12/15(2)	–	–	–	–	–
$\overline{OEY}$ Enable/Disable	14/14(2)	–	–	–	–	–	–	–	–	–	–	–
SIO_0, SIO_3	13	–	–	–	–	–	–	–	–	–	19	20
Clock	58	57	40	–	56	72	24	–	28	56	63	76
Y	–	–	–	16	–	–	–	–	–	–	–	–
$\overline{MSS}$	25	–	25	–	25	25	–	–	–	24	27	24

Notes: 1. Applies only when leaving special functions.
2. Enable/Disable. Enable is defined as output active and correct. Disable is a three-state output turning off.
3. For delay from any input to Z, use input to Y plus Y to Z.

TABLE IIIA – Set-Up and Hold Times (All in ns)
CAUTION: READ NOTES TO TABLE III. NA = Note Applicable; no timing constraint.

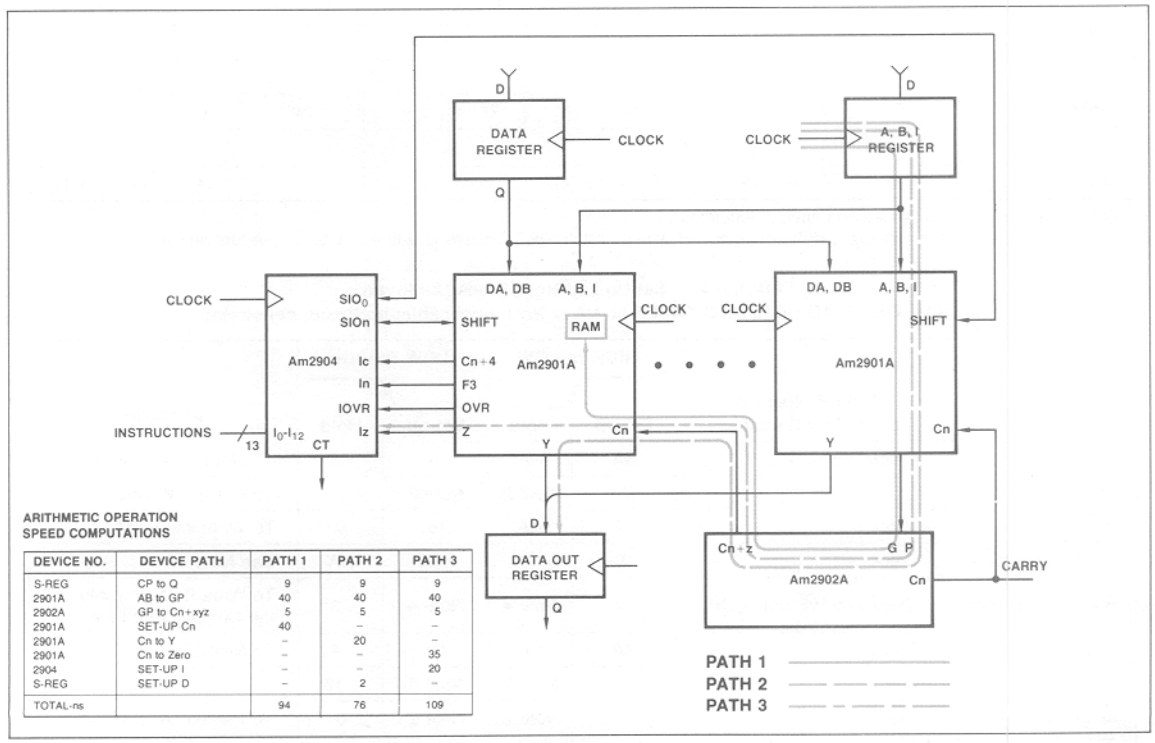
Input	With Respect to to this Signal	HIGH-to-LOW		LOW-to-HIGH		Comment
		Set-up	Hold	Set-up	Hold	
Y	Clock	NA	NA	9	−3	To store Y in RAM or Q
$\overline{WE}$ HIGH	Clock	5	Note 2	Note 2	0	To Prevent Writing
$\overline{WE}$ LOW	Clock	NA	NA	15	0	To Write into RAM
A,B as Sources	Clock	19	−3	NA	NA	See Note 3
B as a Destination	Clock and $\overline{WE}$ both LOW	−4	Note 4	Note 4	−3	To Write Data only into the Correct B Address
QIO_0, QIO_3	Clock	NA	NA	10	−4	To Shift Q
I_{8765}	Clock	2	Note 5	Note 5	−18	
$\overline{IEN}$ HIGH	Clock	10	Note 2	Note 2	0	To Prevent Writing into Q
$\overline{IEN}$ LOW	Clock	NA	NA	10	−5	To Write into Q



LOGIC OPERATION
SPEED COMPUTATIONS

DEVICE NO.	DEVICE PATH	PATH 1	PATH 2	PATH 3
S-REG	CP to Q	9	9	9
2901A	READ-MODIFY-WRITE	55	–	–
2901A	AB – Y	–	45	–
2901A	AB – Zero	–	–	65
2904	SET-UP I	–	–	–
S-REG	SET-UP D	–	2	20
TOTAL-ns		64	56	94

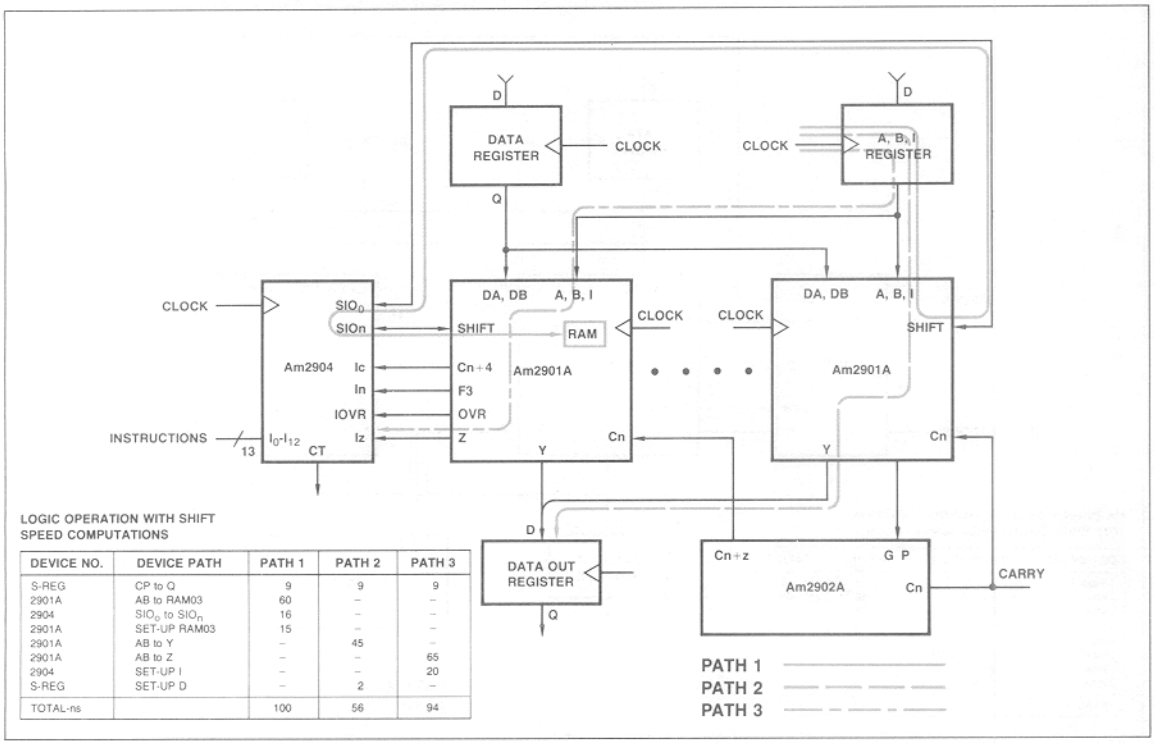
Figure 2-1.



ARITHMETIC OPERATION
SPEED COMPUTATIONS

DEVICE NO.	DEVICE PATH	PATH 1	PATH 2	PATH 3
S-REG	CP to Q	9	9	9
2901A	AB to GP	40	40	40
2902A	GP to Cn+xyz	5	5	5
2901A	SET-UP Cn	40	–	–
2901A	Cn to Y	–	20	–
2901A	Cn to Zero	–	–	35
2904	SET-UP I	–	–	20
S-REG	SET-UP D	–	2	–
TOTAL-ns		94	76	109

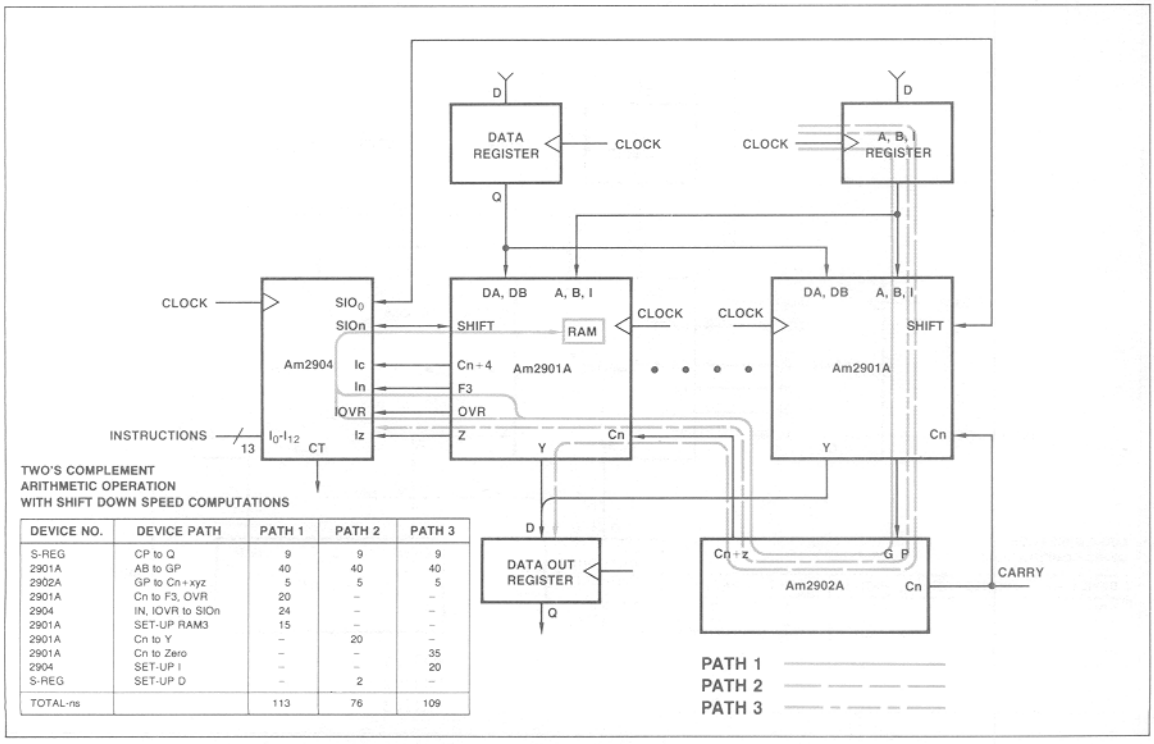
Figure 2-2.



LOGIC OPERATION WITH SHIFT
SPEED COMPUTATIONS

DEVICE NO.	DEVICE PATH	PATH 1	PATH 2	PATH 3
S-REG	CP to Q	9	9	9
2901A	AB to RAM03	60	–	–
2904	SIO_0 to SIO_n	16	–	–
2901A	SET-UP RAM03	15	–	–
2901A	AB to Y	–	45	–
2901A	AB to Z	–	–	65
2904	SET-UP I	–	–	20
S-REG	SET-UP D	–	2	–
TOTAL-ns		100	56	94

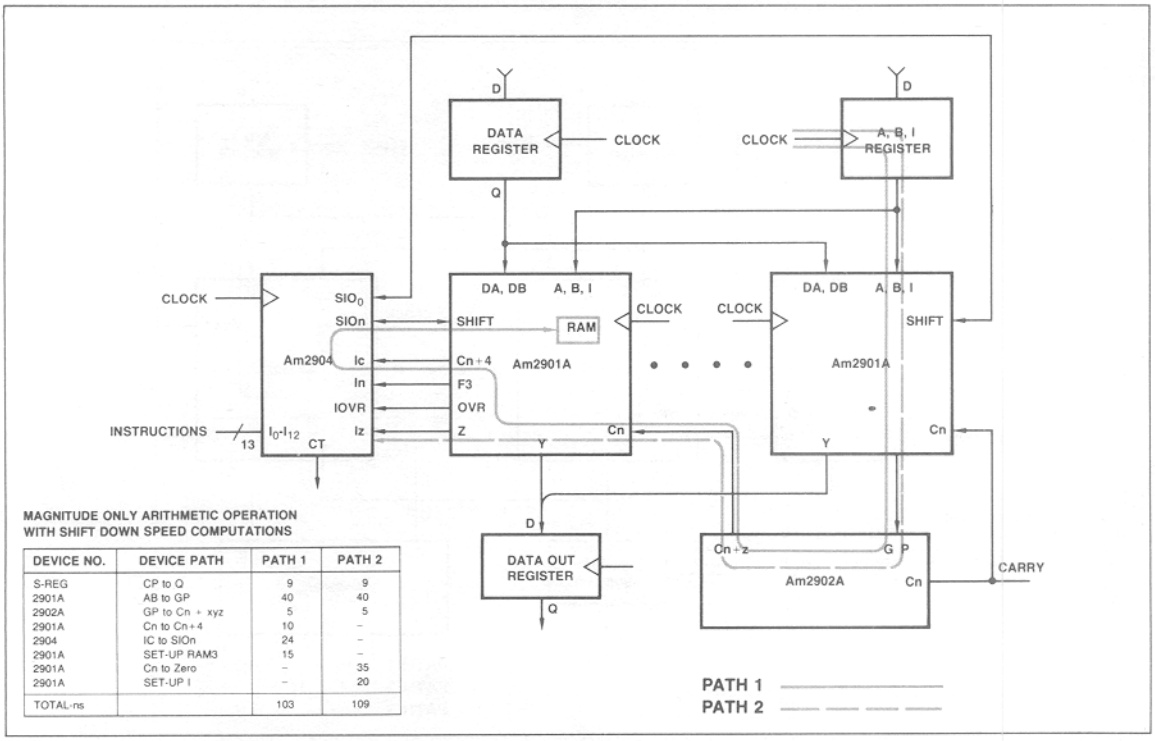
Figure 2-3.



TWO'S COMPLEMENT
ARITHMETIC OPERATION
WITH SHIFT DOWN SPEED COMPUTATIONS

DEVICE NO.	DEVICE PATH	PATH 1	PATH 2	PATH 3
S-REG	CP to Q	9	9	9
2901A	AB to GP	40	40	40
2902A	GP to Cn+xyz	5	5	5
2901A	Cn to F3, OVR	20	–	–
2904	IN, IOVR to SIOn	24	–	–
2901A	SET-UP RAM3	15	–	–
2901A	Cn to Y	–	20	–
2901A	Cn to Zero	–	–	35
2904	SET-UP I	–	–	20
S-REG	SET-UP D	–	2	–
TOTAL-ns		113	76	109

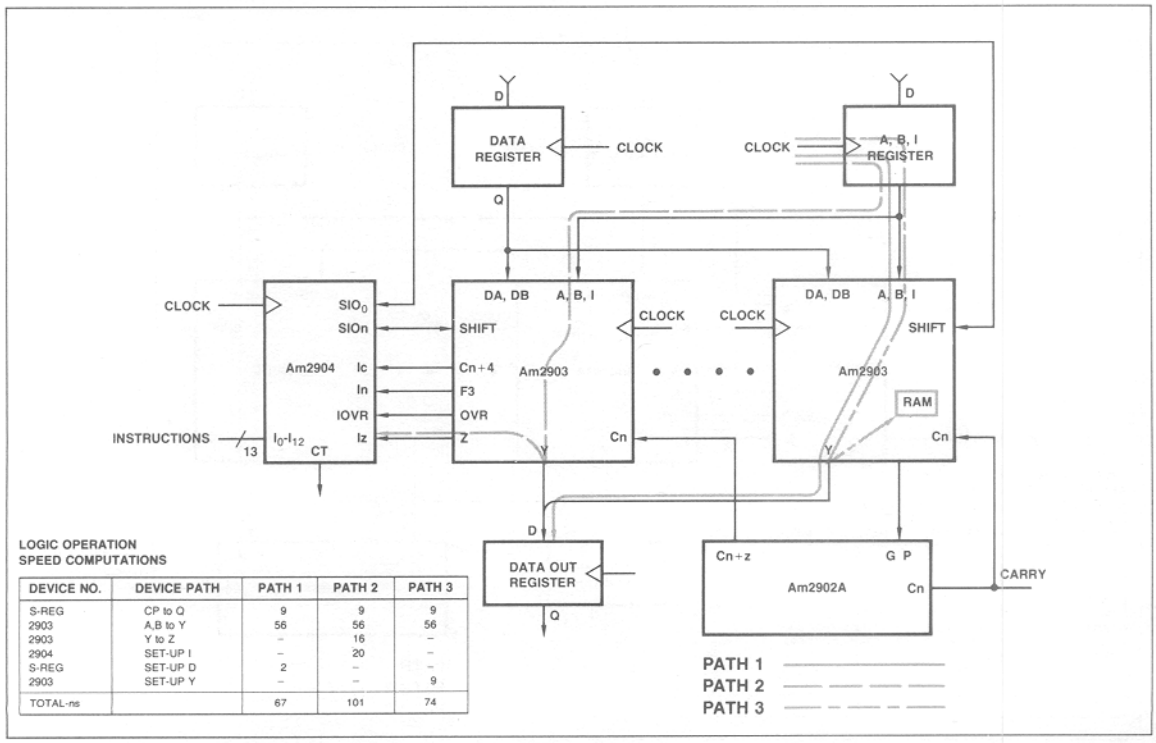
Figure 2-4.



MAGNITUDE ONLY ARITHMETIC OPERATION
WITH SHIFT DOWN SPEED COMPUTATIONS

DEVICE NO.	DEVICE PATH	PATH 1	PATH 2
S-REG	CP to Q	9	9
2901A	AB to GP	40	40
2902A	GP to Cn + xyz	5	5
2901A	Cn to Cn+4	10	–
2904	IC to SIOn	24	–
2901A	SET-UP RAM3	15	–
2901A	Cn to Zero	–	35
2901A	SET-UP I	–	20
TOTAL-ns		103	109

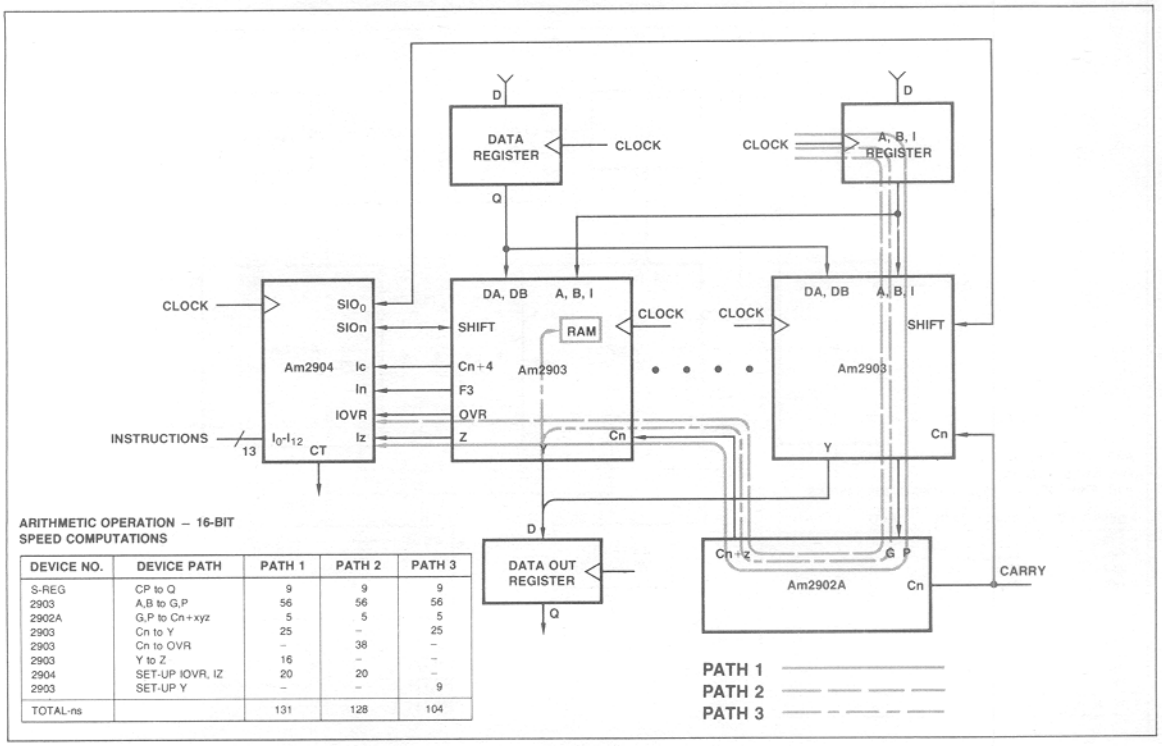
Figure 2-5.



LOGIC OPERATION
SPEED COMPUTATIONS

DEVICE NO.	DEVICE PATH	PATH 1	PATH 2	PATH 3
S-REG	CP to Q	9	9	9
2903	A,B to Y	56	56	56
2903	Y to Z	–	16	–
2904	SET-UP I	–	20	–
S-REG	SET-UP D	2	–	–
2903	SET-UP Y	–	–	9
TOTAL-ns		67	101	74

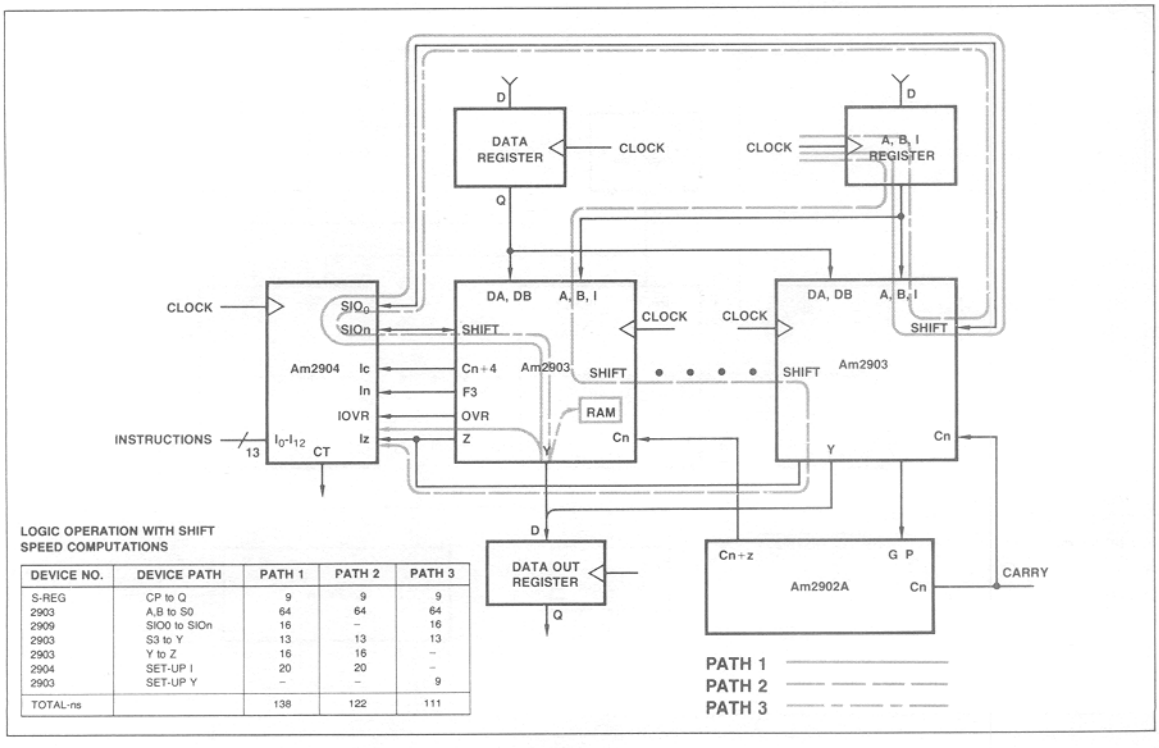
Figure 3-1.



ARITHMETIC OPERATION – 16-BIT SPEED COMPUTATIONS

DEVICE NO.	DEVICE PATH	PATH 1	PATH 2	PATH 3
S-REG	CP to Q	9	9	9
2903	A,B to G,P	56	56	56
2902A	G,P to Cn+xyz	5	5	5
2903	Cn to Y	25	–	25
2903	Cn to OVR	–	38	–
2903	Y to Z	16	–	–
2904	SET-UP IOVR, IZ	20	20	–
2903	SET-UP Y	–	–	9
TOTAL-ns		131	128	104

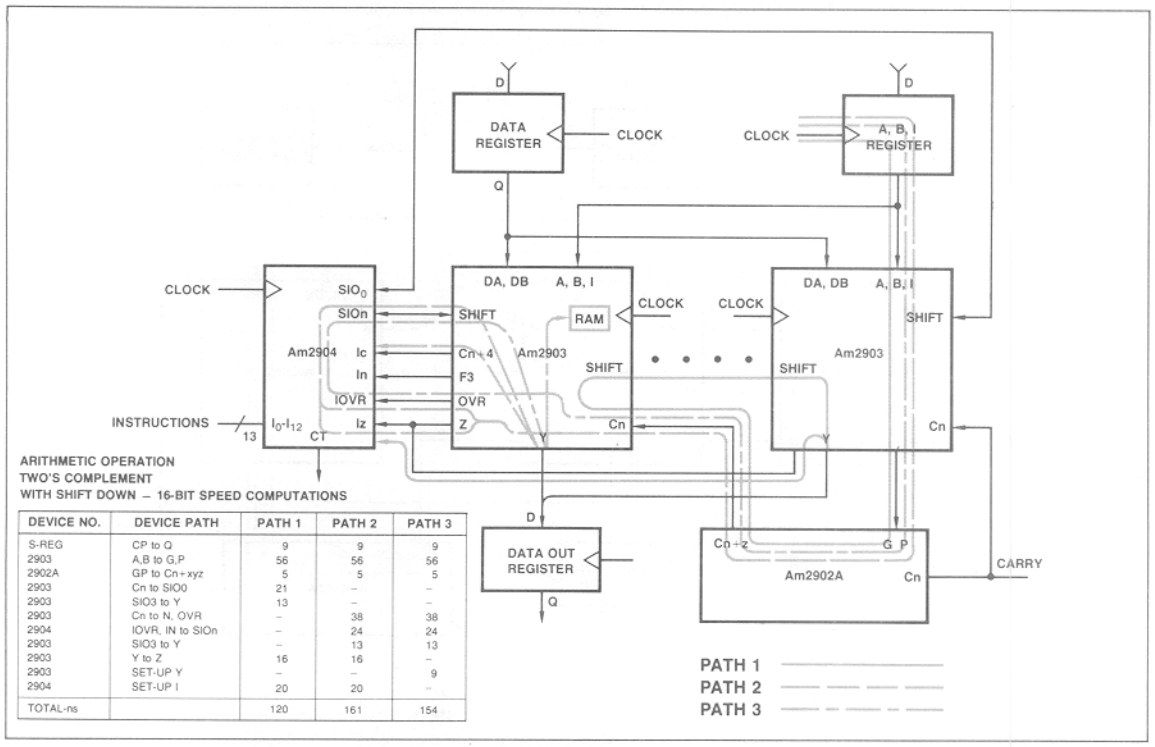
Figure 3-2.



LOGIC OPERATION WITH SHIFT SPEED COMPUTATIONS

DEVICE NO.	DEVICE PATH	PATH 1	PATH 2	PATH 3
S-REG	CP to Q	9	9	9
2903	A,B to S0	64	64	64
2909	SIO0 to SIOn	16	–	16
2903	S3 to Y	13	13	13
2903	Y to Z	16	16	–
2904	SET-UP I	20	20	–
2903	SET-UP Y	–	–	9
TOTAL-ns		138	122	111

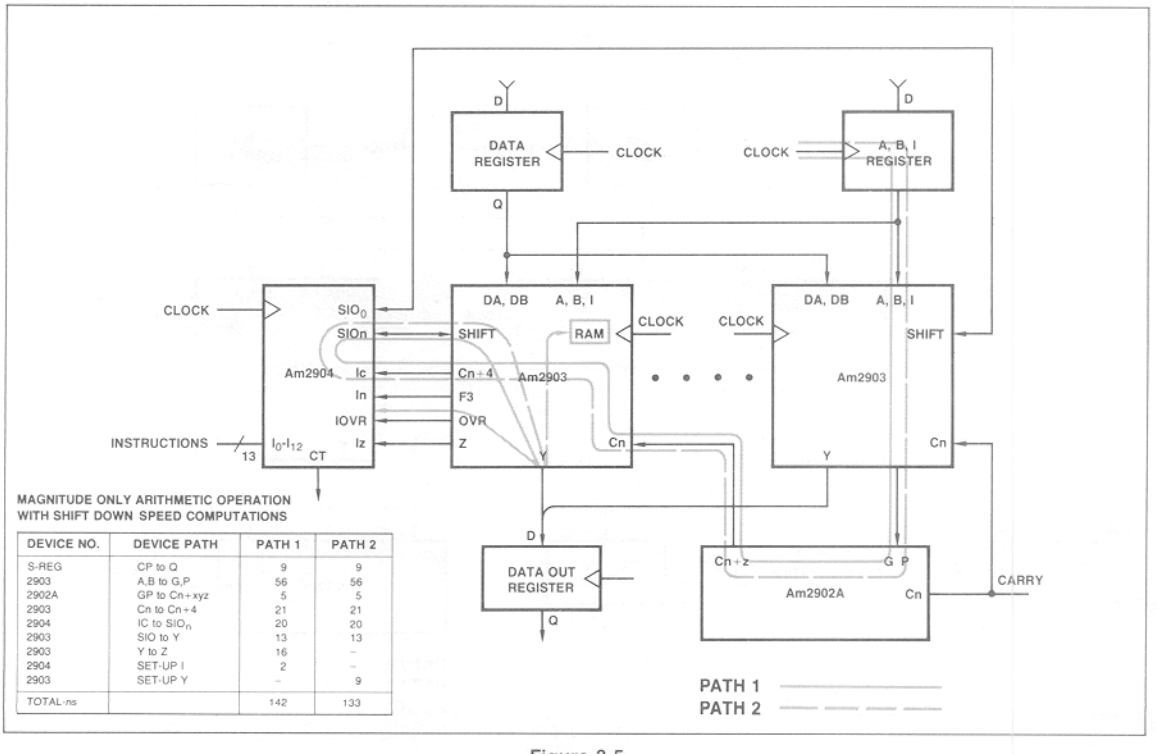
Figure 3-3.



ARITHMETIC OPERATION
TWO'S COMPLEMENT
WITH SHIFT DOWN – 16-BIT SPEED COMPUTATIONS

DEVICE NO.	DEVICE PATH	PATH 1	PATH 2	PATH 3
S-REG	CP to Q	9	9	9
2903	A,B to G,P	56	56	56
2902A	GP to Cn+xyz	5	5	5
2903	Cn to SIO0	21	–	–
2903	SIO3 to Y	13	–	–
2903	Cn to N, OVR	–	38	38
2904	IOVR, IN to SIOn	–	24	24
2903	SIO3 to Y	–	13	13
2903	Y to Z	16	16	–
2903	SET-UP Y	–	–	9
2904	SET-UP I	20	20	–
TOTAL-ns		120	161	154

Figure 3-4.



MAGNITUDE ONLY ARITHMETIC OPERATION
WITH SHIFT DOWN SPEED COMPUTATIONS

DEVICE NO.	DEVICE PATH	PATH 1	PATH 2
S-REG	CP to Q	9	9
2903	A,B to G,P	56	56
2902A	GP to Cn+xyz	5	5
2903	Cn to Cn+4	21	21
2904	IC to SIO_n	20	20
2903	SIO to Y	13	13
2903	Y to Z	16	–
2904	SET-UP I	2	–
2903	SET-UP Y	–	9
TOTAL-ns		142	133

Figure 3-5.

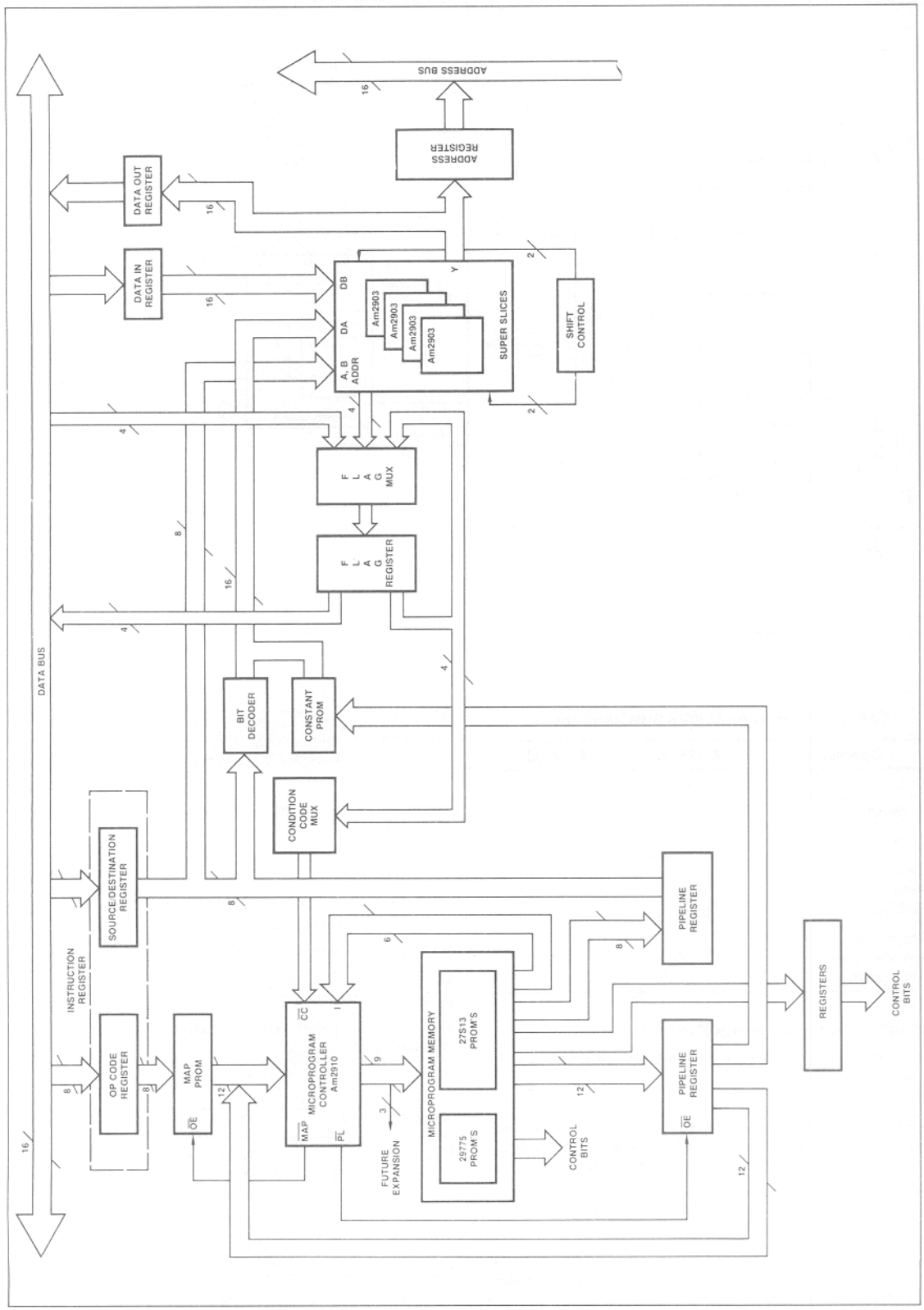

ADDRESS BUS
16
ADDRESS REGISTER
DATA OUT REGISTER
DATA IN REGISTER
16
DB
DA
A, B ADDR
Y
Am2903
Am2903
Am2903
SUPER SLICES
2
SHIFT CONTROL
2
4
FLAG MUX
FLAG REGISTER
8
16
4
4
DATA BUS
BIT DECODER
CONSTANT PROM
CONDITION CODE MUX
SOURCE/DESTINATION REGISTER
INSTRUCTION REGISTER
8
PIPELINE REGISTER
8
6
OP CODE REGISTER
8
MAP PROM
OE
12
CC
MAP
MICROPROGRAM CONTROLLER Am2910
PL
9
3
FUTURE EXPANSION
MICROPROGRAM MEMORY
27S13 PROM'S
29775 PROM'S
CONTROL BITS
12
PIPELINE REGISTER
OE
12
REGISTERS
CONTROL BITS
16

Figure 4a.

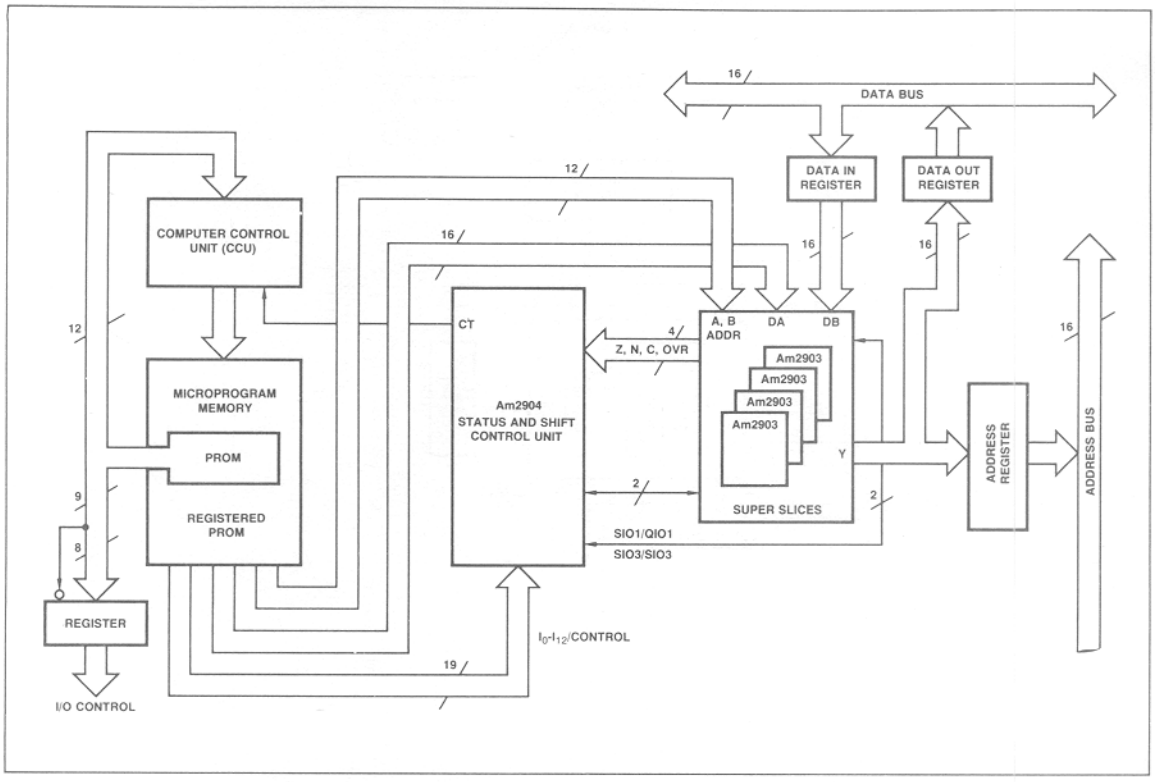


Figure 4b.

TABLE 9. TIMING ANALYSIS SUMMARY (ns).

Operation	Am2901A	Am2903
Logic	94	101
Arithmetic	109	131
Logic w/Shift	100	138
Two's Complement Arithmetic with Shift Down	113	161
Magnitude only Arithmetic with Shift Down	109	142

THE MICROPROGRAM STRUCTURE

The functions of the pipelined (PL) microprogram bits are illustrated in Figure 5 and as follows:

Bits PL0 through PL11 — This is a shared control field. The field is used for branching to a microprogram address or to load the CCU counter or control bits for I/O.

Bit PL12 — The shared control field is determined by PL12, LOW for branching and counting or HIGH for I/O control.

Bit PL13 — When LOW, enables the $\overline{\text{WRITE}}$ output and allows the Q Register and Sign Compare flip-flop to be written into.

Bits PL14 and PL15 — The $\overline{CE\mu}$ and $\overline{SE}$ control inputs of the Am2904, respectively. $\overline{CE\mu}$ enables the Micro Status Register. $\overline{SE}$ enables the Am2904 shift operations.

Bits PL16 through PL19 — CCU Next Address.

Bits PL20 through PL23 — CCU Multiplex test select.

Bit PL24 — This bit determines the polarity of the incoming test signal to the CCU.

Bit PL25 — Active LOW Instruction Register enable.

Bits PL26 through PL29 — CCU multi-way branching select.

Bits PL30 through PL32 — Selects the ALU operand sources.

PL30	PL31	PL32	ALU Operand R	ALU Operand S
L	L	L	RAM Output A	RAM Output B
L	L	H	RAM Output A	DB_{0-3}
L	H	X	RAM Output A	Q Register
H	L	L	DA_{0-3}	RAM Output B
H	L	H	DA_{0-3}	DB_{0-3}
H	H	X	DA_{0-3}	Q Register

L = LOW H = HIGH X = Don't Care

Bits PL33 through PL36 — Selects the ALU functions.

I_4	I_3	I_2	I_1	Hex Code	ALU Functions
L	L	L	L	0	I_0 = L: Special Functions I_0 = H: F_i = HIGH
L	L	L	H	1	F = S Minus R Minus 1 Plus C_n
L	L	H	L	2	F = R Minus S Minus 1 Plus C_n
L	L	H	H	3	F = R Plus S Plus C_n
L	H	L	L	4	F = S Plus C_n
L	H	L	H	5	F = $\overline{S}$ Plus C_n
L	H	H	L	6	F = R Plus C_n
L	H	H	H	7	F = $\overline{R}$ Plus C_n
H	L	L	L	8	F_i = LOW
H	L	L	H	9	F_i = $\overline{R_i}$ AND S_i
H	L	H	L	A	F_i = R_i EXCLUSIVE NOR S_i
H	L	H	H	B	F_i = R_i EXCLUSIVE OR S_i
H	H	L	L	C	F_i = R_i AND S_i
H	H	L	H	D	F_i = R_i NOR S_i
H	H	H	L	E	F_i = R_i NAND S_i
H	H	H	H	F	F_i = R_i OR S_i

L = LOW H = HIGH i = 0 to 3

Bits PL37 through 40 — Selects the ALU destination controls.

I_8	I_7	I_6	I_5	Hex Code	Special Function
L	L	L	L	0	Unsigned Multiply
L	L	H	L	2	Two's Complement Multiply
L	H	L	L	4	Increment by One or Two
L	H	L	H	5	Sign/Magnitude-Two's Complement
L	H	H	L	6	Two's Complement Multiply, Last Cycle
H	L	L	L	8	Single Length Normalize
H	L	H	L	A	Double Length Normalize and First Divide Op.
H	H	L	L	C	Two's Complement Divide
H	H	H	L	E	Two's Complement Divide, Correction and Remainder

Bits PL41 through PL44 — This 4-bit wide field is used for the A-address source.

Bits PL45 through PL48 — This 4-bit wide field is used for the B-address source.

Bits PL49 through PL52 — This 4-bit wide field is the B destination address into which new data is written.

Bit PL53 — Am2903 control input $\overline{OE}_Y$. When LOW enables the ALU shifter output data onto the Y bus.

Bits PL54 through PL59 — Am2904 instruction code field.

Bits PL60 through PL63 — Am2904 shift linkage multiplexer instruction code field.

Bits PL64 and PL65 — Am2904 "carry-in" control multiplexer field.

Bits PL66 through PL68 — The $\overline{CE}_M$, $\overline{OE}_{CT}$, $\overline{OE}_Y$ control inputs of the Am2904, respectively.

Bit PL69 — This bit when LOW, enables bits PL74 through PL89 onto the Am2903 DA Bus.

Bit PL70 — When LOW, zeros the carry in's to the Am2903 slices.

Bit PL71 — When HIGH, enables a status register used in BCD calculations.

Bit PL72 — When LOW, clears the status register.

Bit PL73 — When LOW, enables Am2909/11 registers.

Bits PL74 through PL89 — This field contains a 16-bit constant from microcode that is passed to the Am2903's via the DA bus. Constant is enabled by PL69.

I_0 OR I_1 OR I_2 OR I_4 = HIGH, $\overline{I_{EN}}$ = LOW

I_8	I_7	I_6	I_5	Hex Code	ALU Shifter Function	SIO_3 Most Sig. Slice	SIO_3 Other Slices	Y_3 Most Sig. Slice	Y_3 Other Slices	Y_2 Most Sig. Slice	Y_2 Other Slices	Y_1	Y_0	SIO_0	$\overline{Write}$	Q Reg & Shifter Function	QIO_3	QIO_0
L	L	L	L	0	Arith. F/2→Y	Input	Input	F_3	SIO_3	SIO_3	F_3	F_2	F_1	F_0	L	Hold	Hi-Z	Hi-Z
L	L	L	H	1	Log. F/2→Y	Input	Input	SIO_3	SIO_3	F_3	F_3	F_2	F_1	F_0	L	Hold	Hi-Z	Hi-Z
L	L	H	L	2	Arith. F/2→Y	Input	Input	F_3	SIO_3	SIO_3	F_3	F_2	F_1	F_0	L	Log. Q/2→Q	Input	Q_0
L	L	H	H	3	Log. F/2→Y	Input	Input	SIO_3	SIO_3	F_3	F_3	F_2	F_1	F_0	L	Log. Q/2→Q	Input	Q_0
L	H	L	L	4	F→Y	Input	Input	F_3	F_3	F_2	F_2	F_1	F_0	Parity	L	Hold	Hi-Z	Hi-Z
L	H	L	H	5	F→Y	Input	Input	F_3	F_3	F_2	F_2	F_1	F_0	Parity	H	Log. Q/2→Q	Input	Q_0
L	H	H	L	6	F→Y	Input	Input	F_3	F_3	F_2	F_2	F_1	F_0	Parity	H	F→Q	Hi-Z	Hi-Z
L	H	H	H	7	F→Y	Input	Input	F_3	F_3	F_2	F_2	F_1	F_0	Parity	L	F→Q	Hi-Z	Hi-Z
H	L	L	L	8	Arith. 2F→Y	F_2	F_3	F_3	F_2	F_1	F_1	F_0	SIO_0	Input	L	Hold	Hi-Z	Hi-Z
H	L	L	H	9	Log. 2F→Y	F_3	F_3	F_2	F_2	F_1	F_1	F_0	SIO_0	Input	L	Hold	Hi-Z	Hi-Z
H	L	H	L	A	Arith. 2F→Y	F_2	F_3	F_3	F_2	F_1	F_1	F_0	SIO_0	Input	L	Log. 2Q→Q	Q_3	Input
H	L	H	H	B	Log. 2F→Y	F_3	F_3	F_2	F_2	F_1	F_1	F_0	SIO_0	Input	L	Log. 2Q→Q	Q_3	Input
H	H	L	L	C	F→Y	F_3	F_3	F_3	F_3	F_2	F_2	F_1	F_0	Hi-Z	H	Hold	Hi-Z	Hi-Z
H	H	L	H	D	F→Y	F_3	F_3	F_3	F_3	F_2	F_2	F_1	F_0	Hi-Z	H	Log. 2Q→Q	Q_3	Input
H	H	H	L	E	SIO_0→Y_0, Y_1, Y_2, Y_3	SIO_0	SIO_0	SIO_0	SIO_0	SIO_0	SIO_0	SIO_0	SIO_0	Input	L	Hold	Hi-Z	Hi-Z
H	H	H	H	F	F→Y	F_3	F_3	F_3	F_3	F_2	F_2	F_1	F_0	Hi-Z	L	Hold	Hi-Z	Hi-Z

The Am2903 special functions can be selected by the following conditions: I_0 = I_1 = I_2 = I_3 = I_4 = LOW, $\overline{I_{EN}}$ = LOW

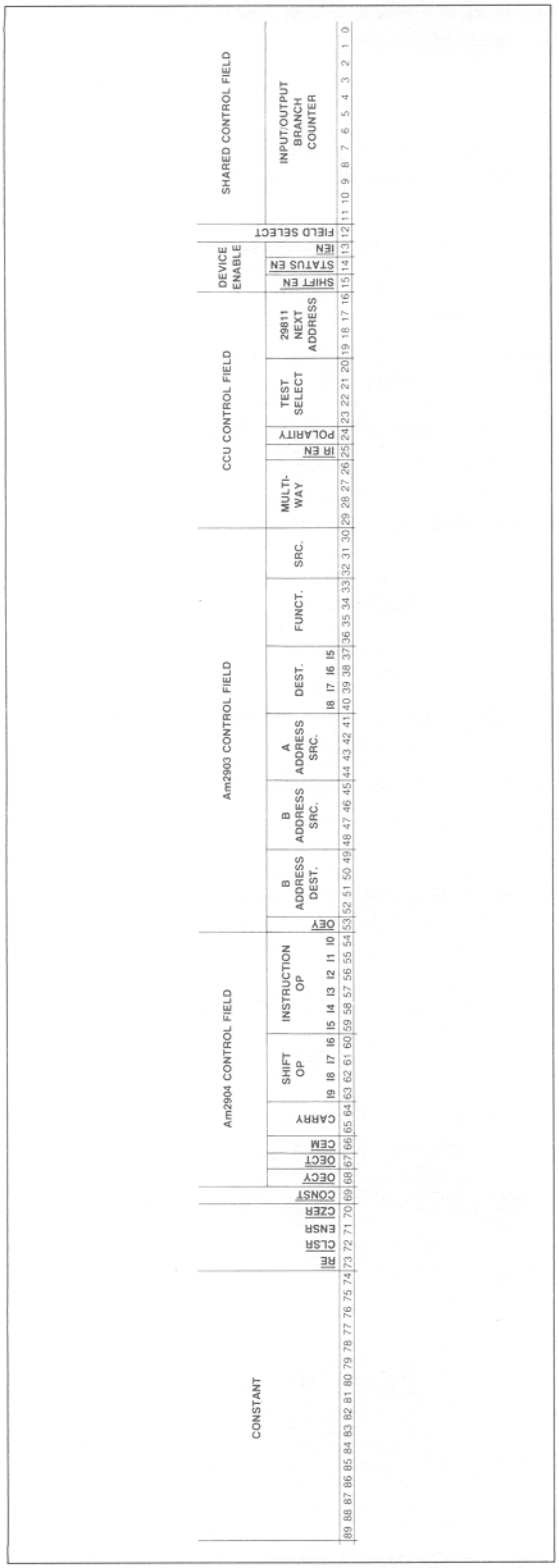


Figure 5.

SOME SAMPLE MICROROUTINES

The following algorithms are implemented using the Am2903 Superslices™ and Am2904 status and shift control unit. The algorithms were developed with the aid of AMDASM on System 29. All algorithms assume values and constants to be initialized prior to the entrance of the algorithms. Appendix A relates the actual microcode to the microword fields. Appendix B is the AMDASM Phase 1 and Phase 2 listings of the microprograms and the definitions of mnemonics. Figure 4b is a block diagram of the CPU hardware including the Am2904 Status and Shift Control Unit from which the microroutines were developed. A detailed diagram of the CPU hardware is in Appendix C.

Normalization, Single- and Double-Length

Normalization is used as a means of referencing a number to a fixed radix point. Normalization strips out all leading sign bits such that the two bits immediately adjacent to the radix point are of opposite polarity.

Normalization is commonly used in such operations as fixed-to-floating point conversion and division. The Am2903 provides for normalization by using the Single-Length and Double-Length Normalize commands. Figure 6a represents the Q Register of a 16-bit processor which contains a positive number. When the Single-Length Normalize command is applied, each positive edge of the clock will cause the bits to shift toward the most significant bit (bit 15) of the Q Register. Zeros are shifted in via the QIO0 port. When the bits on either side of the radix point (bits 14 and 15) are of opposite value, the number is considered to be normalized as shown in Figure 6b. The event of normalization is externally indicated by a HIGH level on the Cn+4 pin of the most significant slice (Cn+4 MSS = Q3 MSS ∀ Q2 MSS).

There are also provisions made for a normalization indication via the OVR pin one microcycle before the same indication is available on the Cn+4 pin (OVR = Q2 MSS ∀ Q1 MSS). This is for use in applications that require a stage of register buffering of the normalization indication.

Since a number comprised of all zeros is not considered for normalization, the Am2903 indicates when wuch a condition arises. If the Q Register is zero and the Single-Length Normalization command is given, a HIGH level will be present on the Z line.

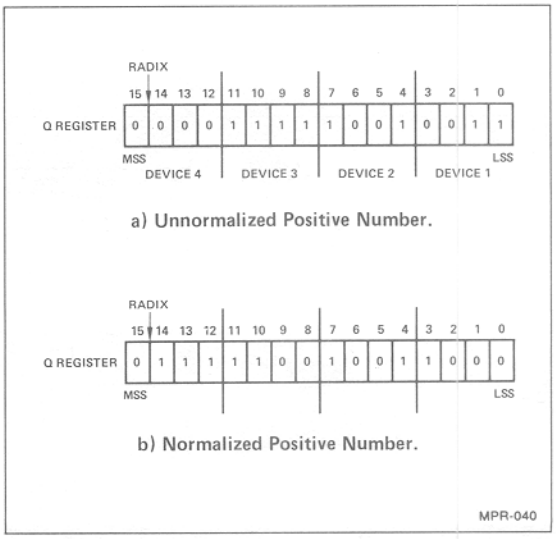


Figure 6.

The sign output, N, indicates the sign of the number stored in the Q register, Q3 MSS. An unnormalized negative number (Figure 7a) is normalized in the same manner as a positive number. The results of single-length normalization are shown in Figure 7b. The device interconnection for single-length normalization is outlined in Figure 8. During single-length normalization, the number of shifts performed to achieve normalization can be counted and stored in one of the working registers. This can be achieved by forcing a HIGH at the Cn input of the least significant slice, since during this special function the ALU performs the function [B] + Cn and the result is stored in B. Figure 9 illustrates the single-length normalize. However, the microcode is shown in Figure 10. Microcode for both single and double normalization can be reduced by one step by testing for zero during passing of number into Q.

Normalizing a double-length word can be done with the Double-Length Normalize command which assumes that a user-selected RAM Register contains the most significant portion of the word to be normalized while the Q Register holds the least significant half (Figure 11.) The device interconnection for double-length normalization is shown in Figure 12. The Cn+4, OVR, N, and Z outputs of the most significant slice perform the same functions in double-length normalization as they did in single-length normalization except that Cn+4, OVR, and N are derived from the output of the ALU of the most significant slice in the case of double-length normalization, instead of the Q Register of the most significant slice as in single-length normalization. A high-level Z line in double-length normalization reveals that the outputs of the ALU and Q Register are both zero, hence indicating that the double-length word is zero.

When double-length normalization is being performed, shift counting is done either with an extra microcycle or with an external counter. Figure 13 illustrates the double-length normalize flowchart and Figure 14 shows the microcode.

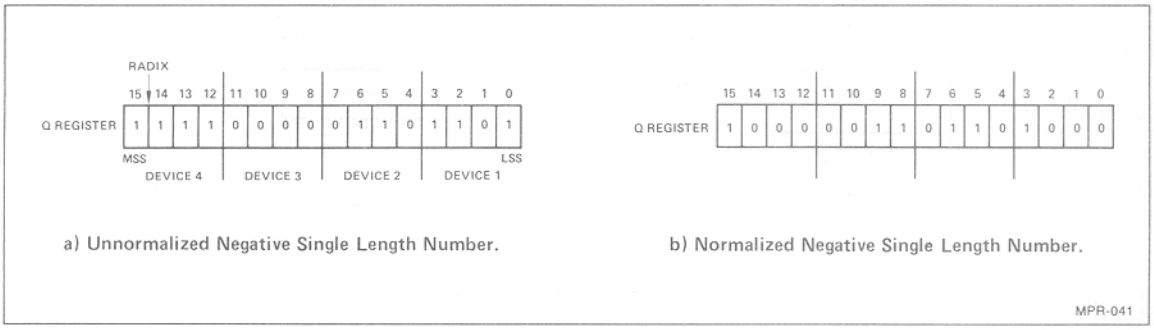


Figure 7.

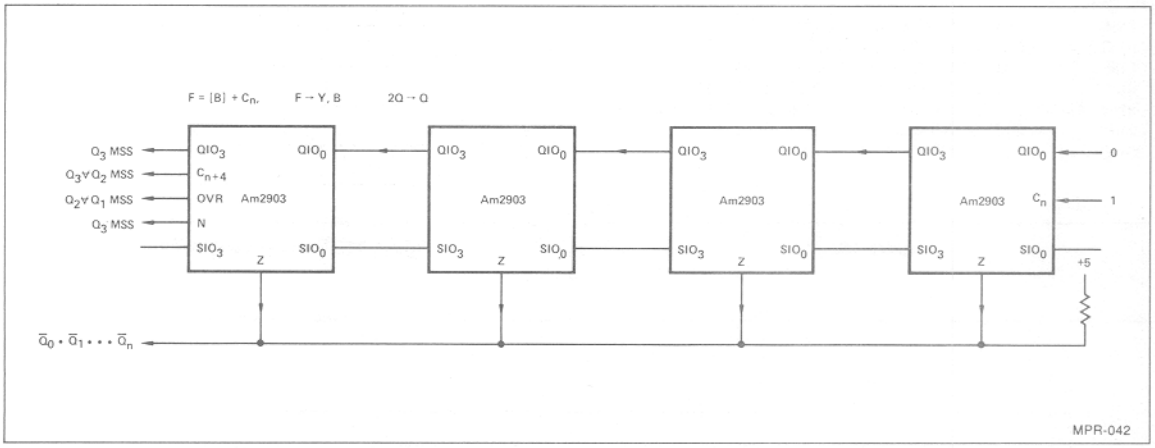


Figure 8. Single Length Normalize.

Unsigned Multiply

This Special Function allows for easy implementation of unsigned multiplication. Figure 15 is the unsigned multiply flow chart. The algorithm requires that initially the RAM word addressed by Address port B be zero, that the multiplier be in the Q Register, and that the multiplicand be in the register addressed by Address port A. The initial conditions for the execution of the algorithm are that: 1) register R_1 be reset to zero; 2) the multiplicand be in R_0 and 3) the multiplier be in R_{15}. The first operation transfers the multiplier, R_{15}, to the Q Register. The Unsigned Multiply instruction is then executed 16 times. During the Unsigned Multiply instruction, R1 is addressed by RAM address port B and the multiplicand is addressed by RAM address port A.

When the unsigned Multiply command is given, the Z pin of device 1 becomes an output while the Z pins of the remaining devices are specified as inputs as shown in Figure 18. The Z output of device 1 is the same state as the least significant bit of the multiplier in the Q Register. The Z output of device 1 informs

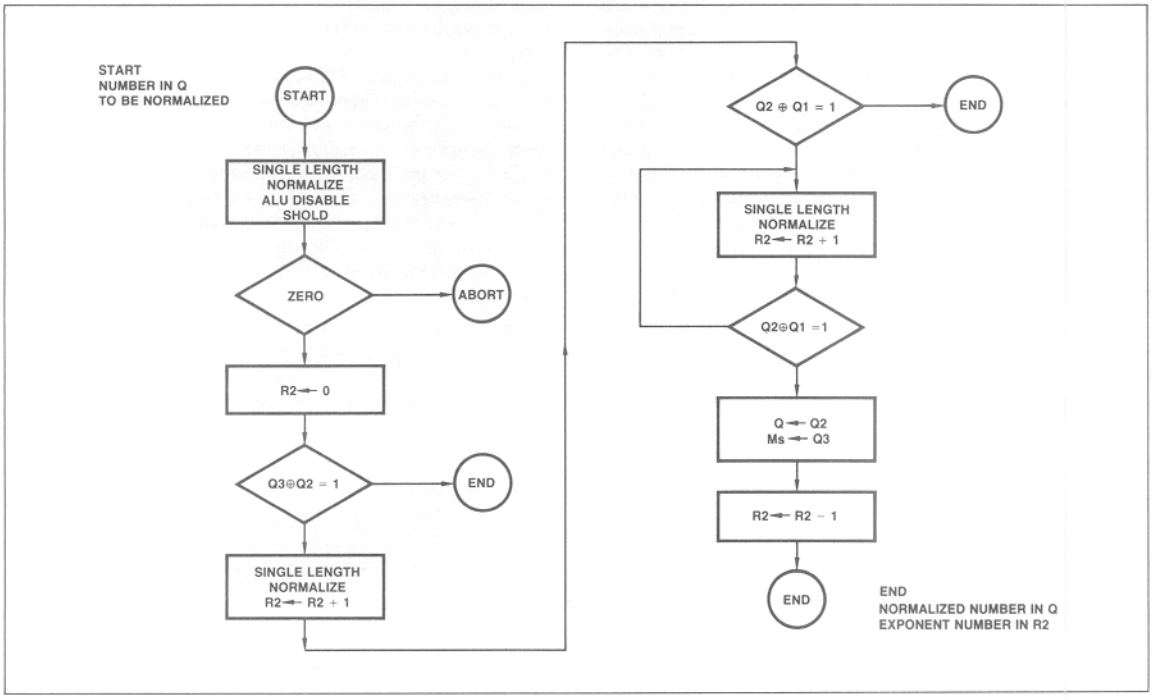


Figure 9. Single Length Normalize.

```
013C          SLN R2,R2,OFF & CONT & SHOLD
013D          MAZ & T & CJP & GOTO ABORT
013E          MAC & T & LOW R0 & CJP & GOTO END
013F          SLN R2,R2 & MAO & T & CJP ONE & GOTO END & SUL
0140  AGAIN:  SLN R2,R2 & MIO & T & CJP ONE & GOTO AGAIN & SUL
0141          SDQP & SMS & CONT
0142          SRS R2,R2,R0 & CONT
```

Figure 10.

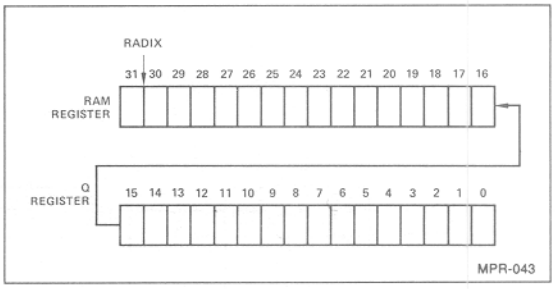


Figure 11. Double Length Word.

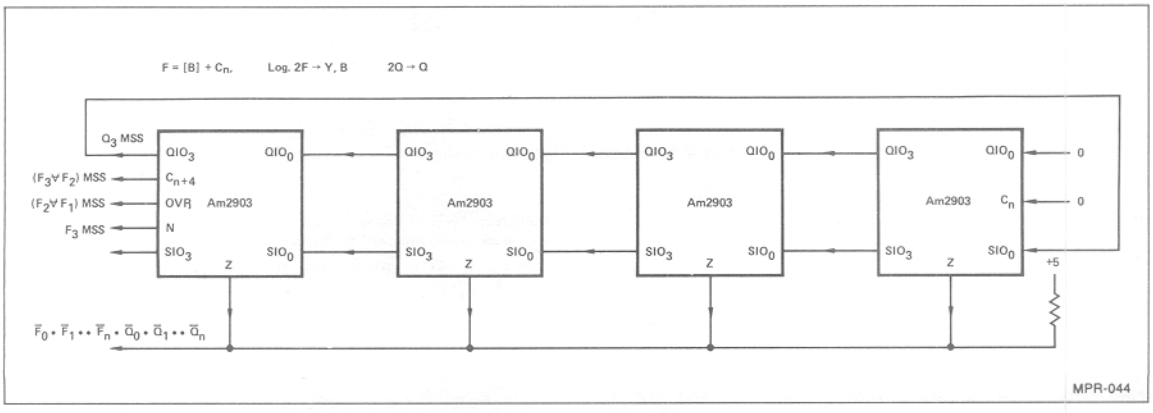


Figure 12. Double Length Normalize.

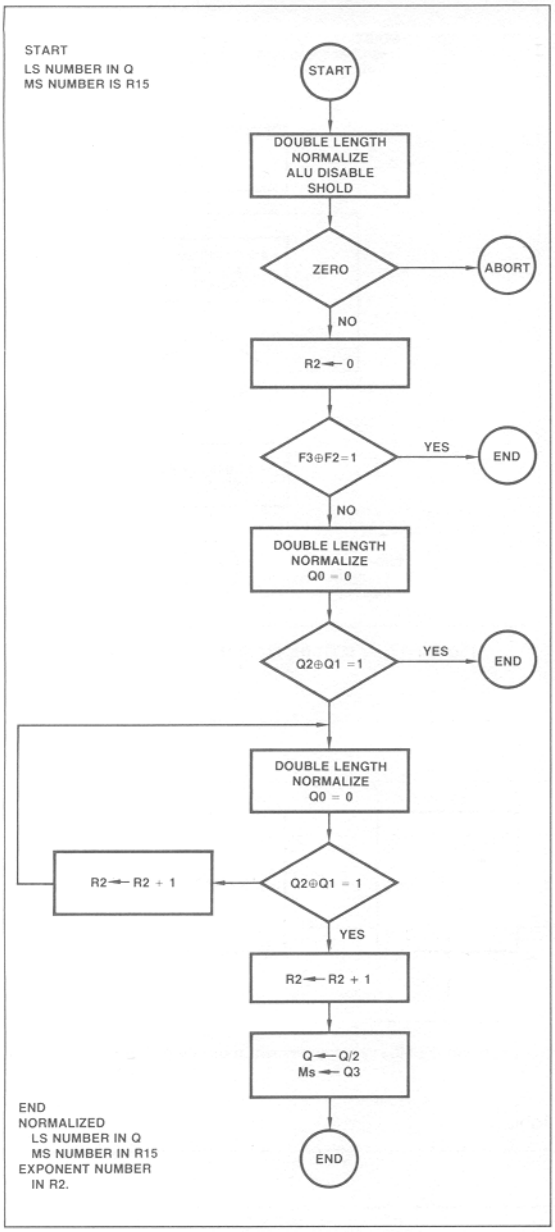


Figure 13. Double Length Normalize.

```
0148          DLN R15,R15,0FF & CONT & SHOLD
0149          MAZ & T & CJP & GOTO ABORT
014A          LOW R2 & MAC & T & CJP & GOTO END2
014B          DLN R15,R15 & SDUL & MAO & T & CJP & GOTO JUMP1
014C  LOOP4:  DLN R15,R15 & SDUL & MIO & T & CJP & GOTO JUMP1
014D          PAR R2,R2 & JP ONE & GOTO LOOP4
014E  JUMP1:  PAR R2,R2 & CONT ONE
014F          SDRQ R15, R15 & SDMS & END
```

Figure 14.

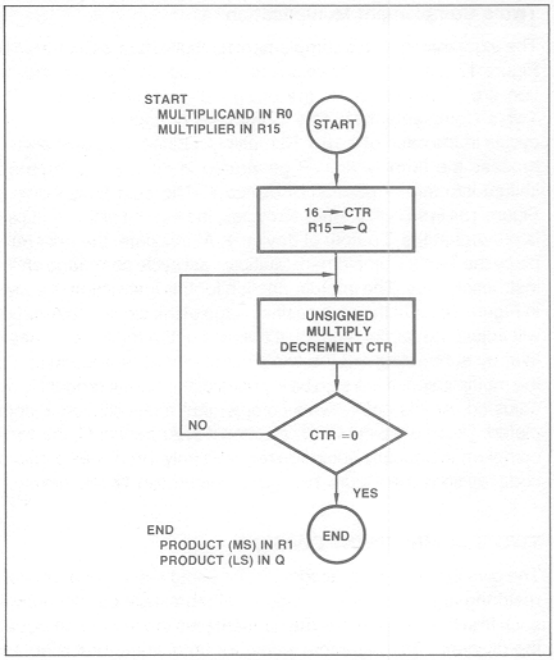


Figure 15. Unsigned 16 X 16 Multiply.

the ALUs of all the slices, via their Z pins, to add the partial product (referenced by the B address port) to the multiplicand (referenced by the A address port) if Z = 1. If Z = 0, the output of the ALU is simply the partial product (referenced by the B address port). Since Cn is held LOW, it is not a factor in the computation. Each positive-going edge of the clock will internally shift the ALU outputs toward the least significant bit and simultaneously store the shifted results in the register selected by the B address port, thus becoming the new partial sum. During the down shifting process, the Cn+4 generated in device 4 is internally shifted into the Y_3 position of device 4. At this time, one bit of the multiplier will down shift out of the QIO_0 ports of each device into the QIO_3 port of the next less significant slice. The partial product is shifted down between chips in a like manner, between the SIO_0 and SIO_3 ports, with SIO_0 of device 1 being connected to QIO_3 of device 4 for purposes of constructing a 32-bit long register to hold the 32-bit product. Shifting of the partial product between the B address and Q registers are accomplished via the Am2904. At the finish of the 16 x 16 multiply, the most significant 16 bits of the product will be found in the register referenced by the B address lines while the least significant 16 bits are stored in the Q Register. Using a typical Computer Control Unit (CCU), as shown in Appendix C, the unsigned multiply operation requires only two lines of microcode, as shown in Figure 16, and is executed in 17 microcycles.

```
010C          LQPT R15 & F & GRD & PUSH & COUNT 00E
010D          UMUL R1,R1,R0 & F & CNT & SDDL & RFCT
```

Figure 16.

Two's Complement Multiplication

The algorithm for two's complement multiplication is illustrated by Figure 17. The initial conditions for two's complement multiplication are the same as for the unsigned multiply operation. The Two's Complement Multiply Command is applied for 15 clock cycles in the case of a 16 x 16 multiply. During the down shifting process the term N ∀ OVR generated in device 4 is internally shifted into the Y_3 position of device 4. The data flow shown in Figure 18a is still valid. After 15 cycles, the sign bit of the multiplier is present at the Z output of device 1. At this time, the user must place the Two's Complement Multiply Last cycle command on the instruction lines. The interconnection for this instruction is shown in Figure 18b. On the next positive edge of the clock, the Am2903 will adjust the partial product, if the sign of the multiplier is negative, by subtracting out the two's complement representation of the multiplicand. If the sign bit is positive, the partial product is not adjusted. At this point, two's complement multiplication is completed. Using a typical CCU, as shown in Appendix C, the two's complement multiply operation requires only three lines of microcode, as shown in Figure 19, and is executed in 17 microcycles.

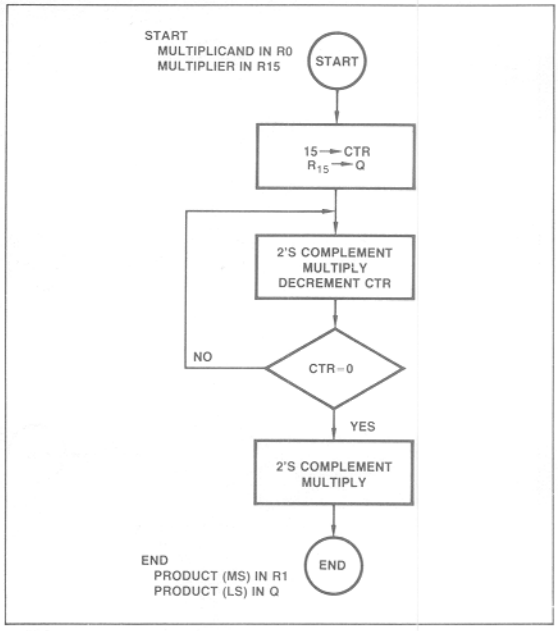


Figure 17. 2's Complement 16 X 16 Multiply.

TWO'S COMPLEMENT DIVISION

The division process is accomplished using a four quadrant nonrestoring algorithm which yields an algebraically correct answer such that the divisor times the quotient plus the remainder equals the dividend. The algorithm works for both single precision and

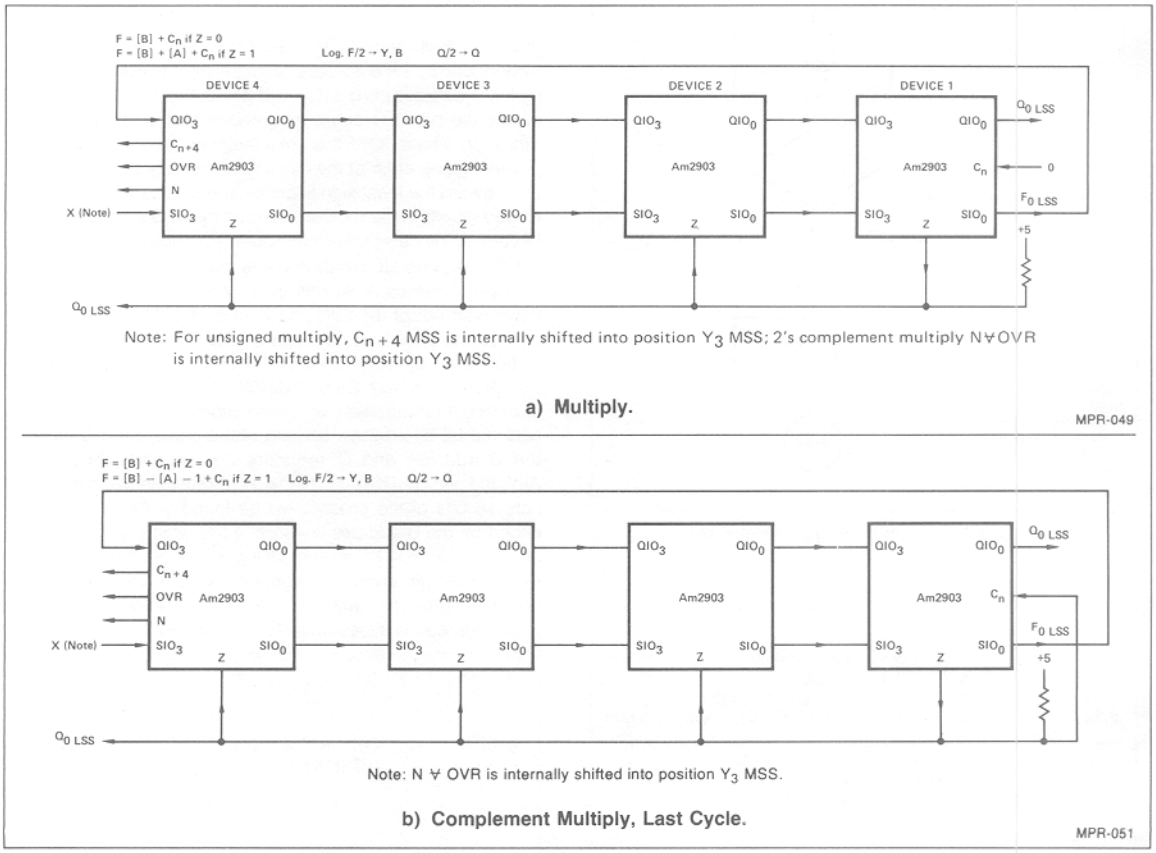


Figure 18.

```
0113        LQPT R15 & F & GRD & PUSH & COUNT 00D
0114        TCM R1,R1,R0 & F & CNT & SDDL & RFCT
0115        TCMC R1,R1,R0 & SDDL & CONT CZ
```

Figure 19.

multi-precision divide operations. The only condition that needs to be met is that the absolute magnitude of the divisor be greater than the absolute magnitude of the dividend. For multi-precision divide operations the least significant bit of the dividend is truncated. This is necessary if the answer is to be algebraically correct. Bias correction is automatically provided by forcing the least significant bit of the quotient to a one, yet an algebraically correct answer is still maintained. Once the algorithm is completed, the answer may be modified to meet the user's format requirements, such as rounding off or converting the remainder so that its sign is the same as the dividend. These format modifications are accomplished using the standard Am2903 instructions.

The true value of the remainder is equal to the value stored in the working register times 2^{n-1} when n is the number of quotient digits.

The following paragraphs describe a double precision divide operator.

Referring to the flow chart outlined in Figure 20, we begin the algorithm with the assumption that the divisor is contained in R_0, while the most significant and least significant halves of the dividend reside in R_1 and R_4 respectively. The first step is to duplicate the divisor by copying the contents of R_0 into R_3. Next the most significant half of the dividend is copied by transferring the contents of R_1 into R_2 while simultaneously checking to ascertain if the divisor (R_0) is zero. If the divisor is zero then division is aborted. If the divisor is not zero, the copy of the most significant half of the dividend in R_2 is converted from its two's complement to its sign magnitude representation. The divisor in R_3 is converted in like manner in

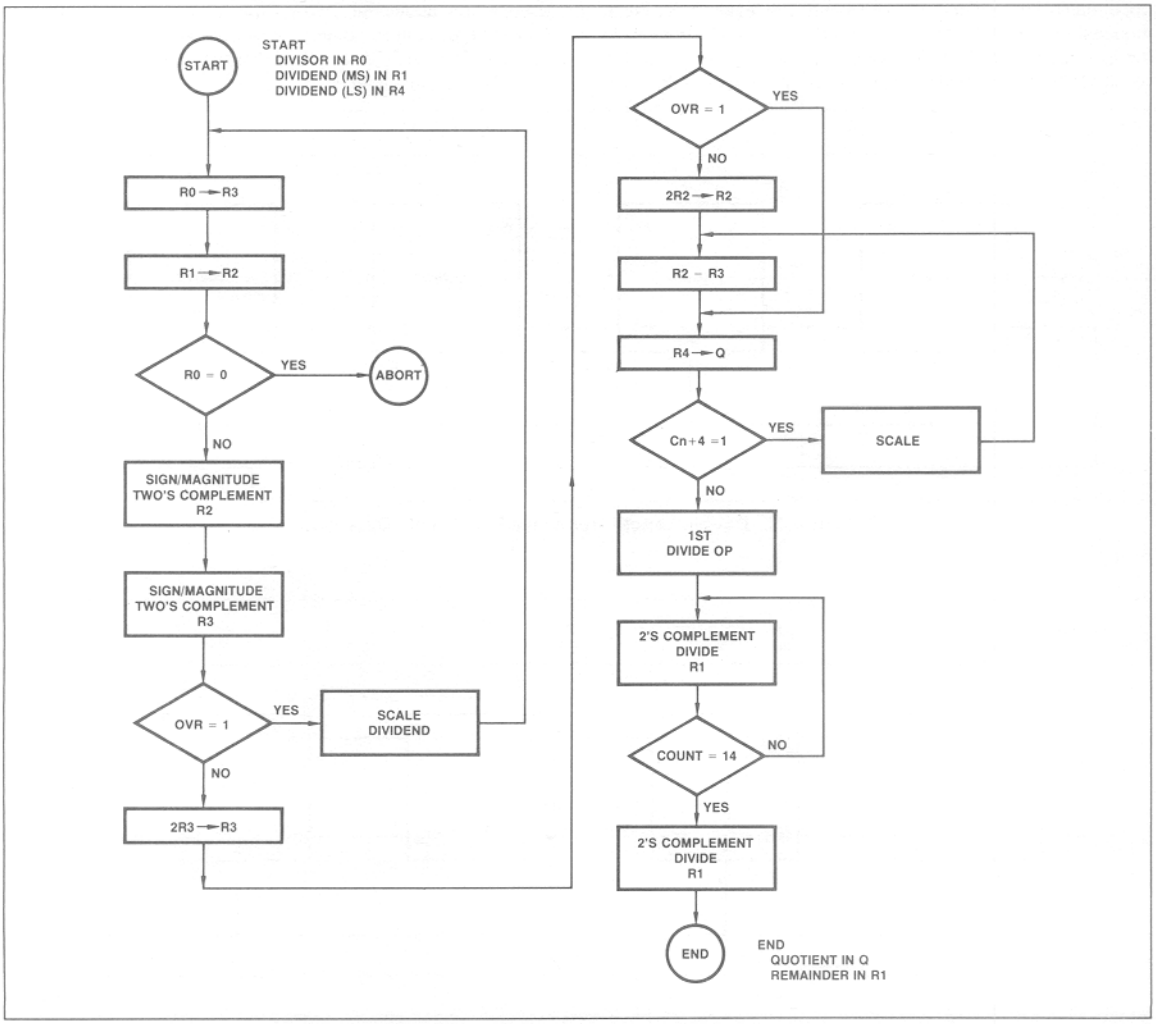


Figure 20. Two's Complement Division.

the next step, while testing to see if the results of the dividend conversion yielded an indication on the overflow pin of the Am2903. If the output of the overflow pin is 'one' then the dividend is -2^n and hence is the largest possible number, meaning that it cannot be less than the divisor. What must be done in this case is to scale the dividend by down shifting the upper and lower halves stored in R_1 and R_4 respectively. After scaling, the routine requires that the algorithm be reinitiated at the beginning.

Conversely, if the output of the overflow pin is not a one, the sign magnitude representation of the divisor (R_3) is shifted up in the Am2903, removing the sign while at the same time testing the results of two's complement to sign magnitude conversion of the divisor in the Am2910. If the results of the test indicate that the divisor is -2^n i.e., overflow equals one, then the lower half of the dividend is placed in the Q register and division may proceed. This is possible because the divisor is now guaranteed to be greater than the dividend. If overflow is not a one then we must proceed by shifting out the sign of the sign magnitude representation of the dividend stored in R_2. At this point we are able to check if the divisor is greater than the dividend by subtracting the absolute value of the divisor (R_3) from the absolute value of the upper half of the dividend (R_2) and storing the results in R_3. Next, the least significant half of the dividend is transferred from R_4 to the Q register while simultaneously testing the carry from the result of the divisor/dividend subtraction. If the carry (Cn+4) is one, indicating the divisor is not greater than the dividend then a scaling operation must occur. This involves either shifting up the divisor or shifting down the dividend. If the carry is not one then the divisor is greater than the dividend and division may now begin.

The first divide operation is used to ascertain the sign bit of the quotient. The two's complement divide instruction is then executed repetitively, fourteen times in the case of a sixteen bit divisor and a thirty-two bit dividend. The final step is the two's complement correction command which adjusts the quotient by allowing the least significant bit of the quotient to be set to one. At the end of the division algorithm the sixteen bit quotient is found in the Q register while the remainder now replaces the most significant half of the dividend in R_1. It should be noted that the remainder must be shifted down fifteen places to represent its true value. The interconnections for these instructions are shown in Figures 21, 22, 23. Using a typical CCU as shown in Appendix C, the double precision divide operation microcode, is shown in Figure 24.

For those applications that require truncation instead of bias correction, the same algorithm as above should be implemented except one additional Two's Complement Divide instruction should be used in lieu of the Two's Complement Divide Correction and Remainder instruction. However, this technique results in an invalid remainder.

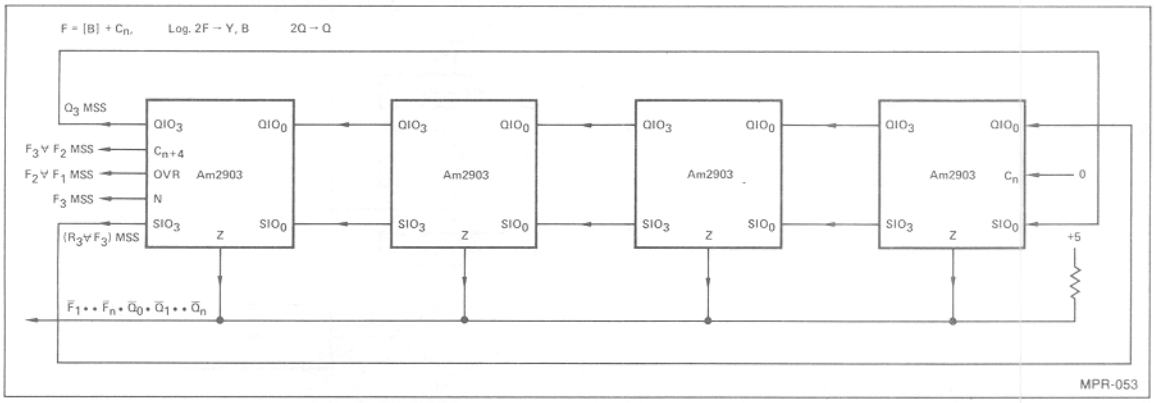


Figure 21. Double Length Normalize/First Divide Operation.

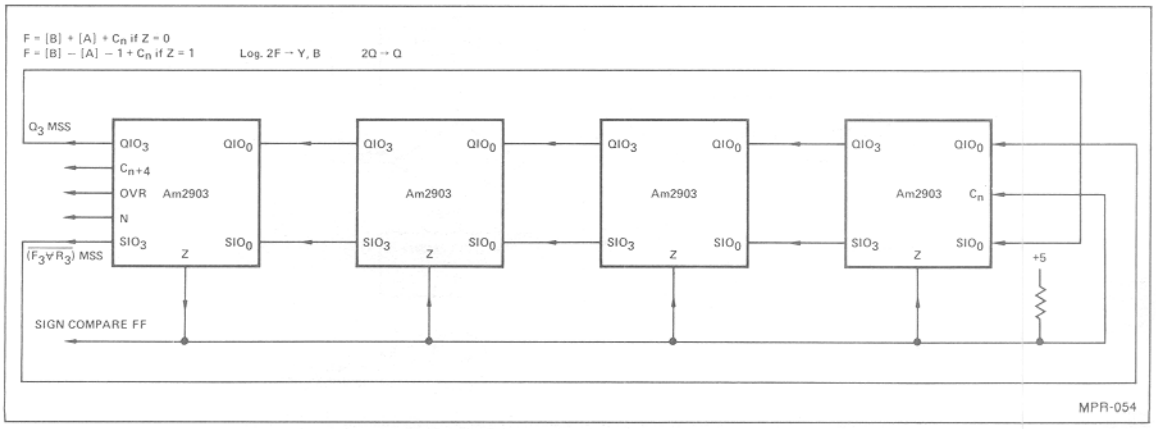


Figure 22. 2's Complement Divide.

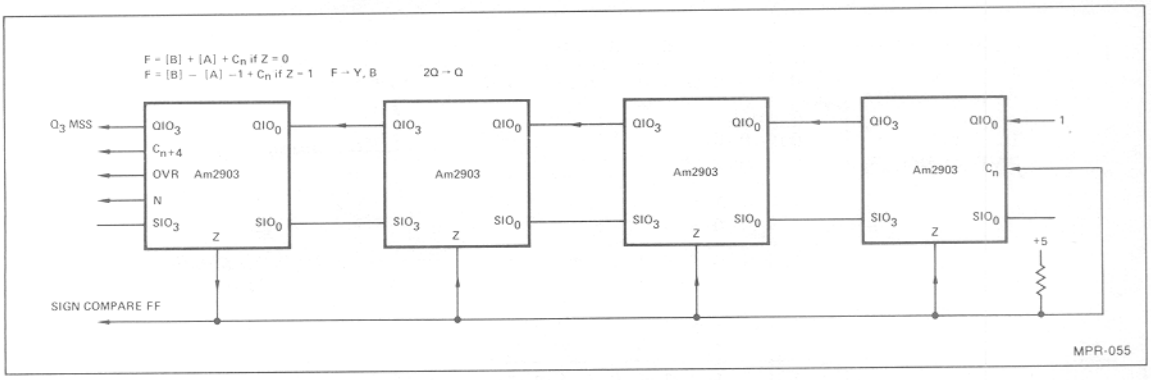


Figure 23. 2's Complement Divide Correction.

```
0119   DIV:      LOW R10 & JSR & GOTO INP
011A             PAR R7,R15 & JSR & GOTO INP
011B             PAR R1,R15, & JSR & GOTO INP
011C             PAR R4,R15 & CONT
011D   LOOP1:    PAR R3,R7 & CONT
011E             PAR R2,R1 & T & MIZ & CJP & GOTO ABORT
011F             SMTC R2,R2 & CONT Z
0120             SMTC R3,R3 & T & MIO & CJP CZ & GOTO SCALE1
0121             ALUOFF & T & MIO & CJP & GOTO SKIP6
0122             SURL R3,R3 & SUL & CONT
0123             SURL R2,R2 & SUL & CONT
0124             ALUOFF & JP & GOTO LOOP2
0125   SCALE1:   LQPT R4 & JSR & GOTO SDIVD
0126             ALUOFF & JP LOOP1
0127   LOOP2:    SSR R3,R2,YBUS & CONT ONE
0128   SKIP6:    LQPT R4 & F & MIC & CJP & GOTO SKIP3
0129             ALUOFF & JSR & GOTO SDIVD
012A             SURL R2,R2 & SDL & CONT
012B             ALUOFF & JP & GOTO LOOP2
012C   SKIP3:    ALUOFF & F & GRD & LDCT & COUNT 00C
012D             DLN R1,R1,R7 & T & GRD & SDUL & PUSH
012E             TDIV R1,R1,R7 & F & CNT & SDUL & RFCT CZ
012F             TDC R1,R1,R7 & SUH & CONT CZ
0130             QMOV R15 & JSR & GOTO OUTP
0131             PAR R15,R1 & JSR & GOTO OUTP
0132             ALUOFF & JP & GOTO DIV
0133   SDIVD:    PAR R1,R1 & CONT
0134             ALUOFF & T & MIS & CJP & GOTO NEG
0135             PAR R1,R1,ADRQ & SDDL & CONT
0136             ALUOFF & JP & GOTO RET
0137   NEG:      PAR R1,R1,ADRQ & SDDL & CONT
0138   RET:      QMOV R4 & CONT
0139             PAR R10,R10 & RTN ONE
```

Figure 24.

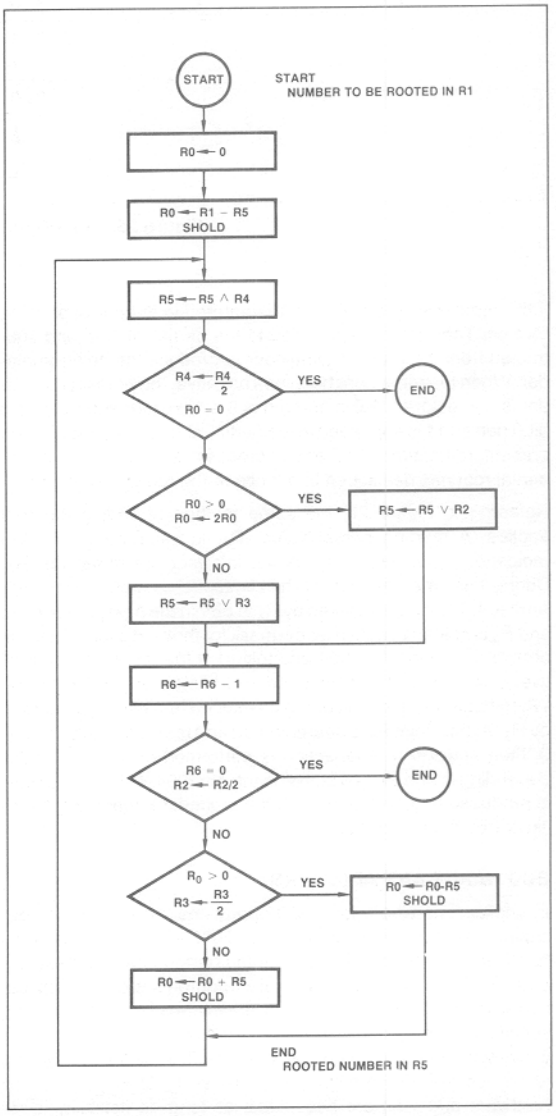


Figure 25. Non-Restoring Binary Root.

NON-RESTORING BINARY ROOTS

The algorithm for Non-Restoring Binary Roots is illustrated in Figure 25. The initial conditions required are: 1) the non-negative number to be rooted in the radicand register, R_1; 2) R_2 has the positive append bits 101_B; 3) R_3 has the negative append bits 011_B; 4) R_4 is the mask register with $BFFF_H$; 5) R_5 is the partial register with 4000_H; and 6) the counter register, R_6, with the value 08_H.

An example of the Non-Restoring Binary Root algorithm is shown in Figure 26. Starting at the binary point, the number to be rooted is partitioned into pairs. The partial value is subtracted from the first pair. An intermediate remainder and sign are then produced.

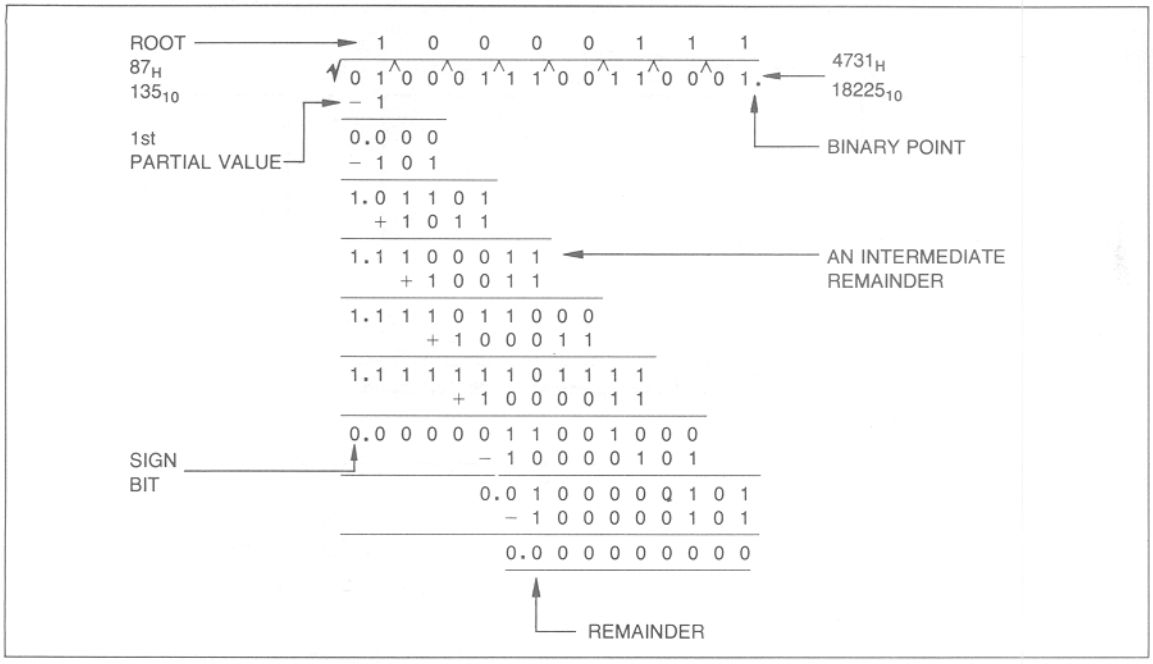


Figure 26. Non-Restoring Binary Root Example.

If the remainder is positive, a 1 is entered in the corresponding root bit. Then a 01 is appended to the partial, shifted and subtracted from the present remainder to produce the next remainder. When the remainder becomes negative, the present remainder is not restored. A 0 is entered in the next corresponding root bit. Then an 11 is appended to the partial, shifted and added to the present remainder. The entire process is repeated until the partial root has developed into 8 bits or the remainder is zero.

Referring to Figure 26, the same method of finding the root applies. A starting partial value, R_5, is subtracted from the radicand, R_1, which produces the intermediate remainder R_0. During this time, the sign of the remainder is stored within the Am2904. Then R_5 is masked by R_4 to obtain the next partial value and R_4 is shifted to obtain a new mask for the next cycle. Status is obtained from the Am2904 and tested. If the remainder is positive, a root bit of 1 is developed and bits 01 appended by R_2. When negative, a root bit of 0 is developed and bits 11 appended by R_3. At this point R_6 is decremented and tested for zero. If $R_6 \neq 0$, then addition or subtraction is performed on the remainder depending on the sign bit stored in the Am2904. A new remainder is produced and cycled through the procedure again. Figure 27 illustrates the microcode.

```
0152    SQRT:   LOW R10 & CONT
0153            LOW R0 & CONT
0154            PAR R1,R15 & CONT
0155            PAR R2,R0,,DARB & CONST 0005 & CONT
0156            PAR R3,R0,,DARB & CONST 0003 & CONT
0157            PAR R4,R0,,DARB & CONST H#BFFF & CONT
0158            PAR R4,R0,,DARB & CONST 4000 & CONT
0159            PAR R6,R0,,DARB & CONST 0008 & CONT
015A            SRS R0,R1,R5 & CONT & SHOLD
015B    CYCLE:  AND R5,R5,R4 & CONT
015C            SDRL R4,R4 & MAS & CJP & GOTO END3
015D            SDRL R0,R0, & T & MAS & CJP & GOTO POS
015E            OR R5,R3 & JP & GOTO CNT
015F    POS:    OR R5,R2 & CONT
0160    CNT:    SRS R6,R6,RIO & CONT
0161            SDRL R2,R2, & T & MIZ & CJP & GOTO END3
0162            SDRL R3,R3 & T & MAS & CJP & GOTO SUB
0163            ADD R0,R0,R5 & JP & GOTO CYCLE & SHOLD
0164    SUB:    SRS R0,R0,R5 & JP & GOTO CYCLE & SHOLD
0165    END3:   JP & GOTO SQRT
```

Figure 27.

BCD HARDWARE ADDITIONS

In applications where fast BCD operations are needed the designer has the option of using a slight amount of additional hardware to dramatically increase the performance of these operations. These firmware/hardware trade-off's are very application sensitive. The hardware-firmware examples given below are specifically for an intensive BCD system with a large fraction of conventional logic-arithmetic operations. The designer is willing to reduce cycle time slightly to increase BCD thru-put. Small hardware additions are acceptable as long as flexibility is retained.

The hardware additions finally decided on were chosen to increase the performance of BCD to binary conversion, binary to BCD conversion and BCD addition. The performance increases were approximately an order of magnitude in the first two cases, and a factor of 4 or 5 in the last case. A diagram of the additions (3¼ ICs) is given in Figure 28.

The 74S08 AND gates normally pass the carry from the Am2902A to the Am2903s. When microbit $\overline{CZER}$ is low the Carries-in are forced to zero. This is used to "disconnect" the carry so that a test may be done on each slice simultaneously. For example if a test for 5 or greater is desired a HEX B is added and

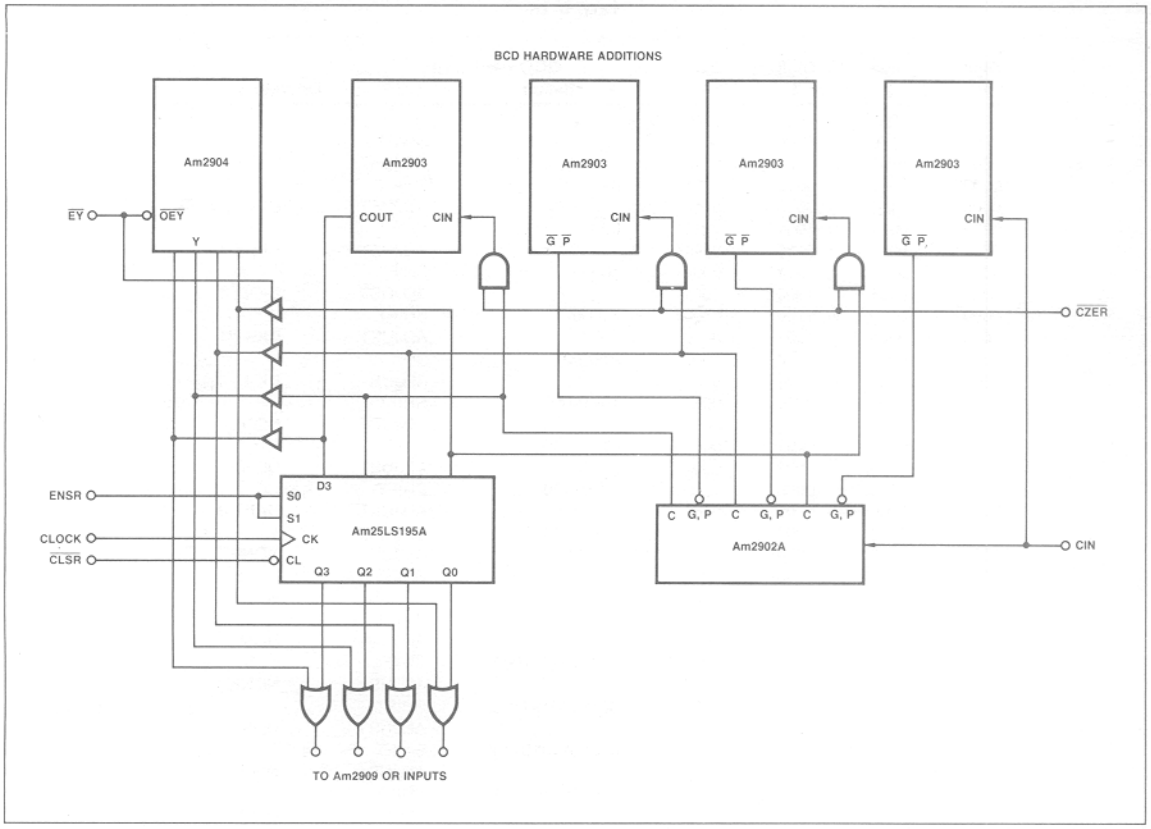


Figure 28.

the carry out of each slice will indicate the result of the test. This allows simultaneous tests on each individual slice and greatly increases thru-put. This addition increases the performance of BCD to binary conversion and binary to BCD conversion by at least an order of magnitude. The drawback to this addition is that the AND-gates introduce an extra gate delay in a critical path. The machine cycle time may be increased by about 8ns. The increase in BCD performance will more than offset this delay for BCD intensive systems.

Another hardware addition is the Am25LS241 three-state buffer. This buffer allows the Am2904 to be used to store the carry-out status bits via the bi-directional Y bus.

The 25LS195A is wired as a 4-bit register with clear and enable. This register is used to store the carry-out bits from a test cycle. The outputs of the 25LS195A are ORed with the output of the Am2904 Y-bus and connected to the Am2909 OR inputs in the CCU. This allows a multi-way branch on the OR of two test cycles, greatly increasing the performance of BCD addition.

BCD TO BINARY CONVERSION

The usual method of BCD to binary conversion is to divide the BCD number by 2. The 1-bit remainder will indicate if a 1 existed in the BCD number. The previous division result is divided by 2 again and the remainder will indicate if a 2 existed in the BCD number. In general the remainder from a division by 2^n will indicate if a 2^{n-1} existed in the BCD number. These remainders can be used to construct the binary representation, $b_n2^n + b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_12^1 + b_02^0$. The b_n bit is thus the remainder from division step n + 1. The binary representation may thus be created by shifting the rernainders down until the m-bit BCD number has been divided by 2 m times.

To divide a BCD number by 2 a down shift is executed. The 4, 2 and 1-bit positions will contain the correct result, but the 8-bit position is incorrect. Its value has changed from 10 to 8 instead of from 10 to 5. This means the resulting BCD number will have a value 3 greater than it should for the division by 2 to be correct. A 3 must be subtracted from any digit in which a 1 entered its 8-bit.

A sample conversion is given in Table 10. The BCD number is gradually shifted down and corrected when necessary. The binary number is finally correct after 16 cycles.

A flow diagram for the algorithm is given in Figure 29. The BCD input, A, is shifted down into the binary output B, to start the loop. The constant 0888 is added to A with the carries-in forced to zero. The resulting carries-out will indicate if A contained a 1 in any of the 8-bit positions. These carries are saved in status register SR1. A multi-way branch is then executed to enter the adjust table. The digits are adjusted depending on the previous test. At the same time a shift can be executed to prepare for the next test instruction. A test for end of loop is also done in this cycle to provide an exit if 16 iterations of the loop are complete. Finally a shift up of B is needed to cancel the extra right shift when the loop is exited. The microcode for this algorithm is given in Figure 30.

TABLE 10.

Digit 3	Digit 2	Digit 1	Digit 0	BCD → Binary Result	Operation	
0010	1001	0000	0100			
0001	0100	1000	0010	0	SHIFT	
0001	0100	0101	0010		ADJUST	DIGIT 1
0000	1010	0010	1001	00	SHIFT	
0000	0111	0010	0110		ADJUST	DIGITS 2, 0
0000	0011	1001	0011	000	SHIFT	
0000	0011	0110	0011		ADJUST	DIGIT 1
0000	0001	1011	0001	1000	SHIFT	
0000	0001	1000	0001		ADJUST	DIGIT 1
0000	0000	1100	0000	11000	SHIFT	
0000	0000	1001	0000		ADJUST	DIGIT 1
0000	0000	0100	1000	011000	SHIFT	
0000	0000	0100	0101		ADJUST	DIGIT 0
0000	0000	0010	0010	1011000	SHIFT	
0000	0000	0010	0010		ADJUST	NONE
0000	0000	0001	0001	01011000	SHIFT	
0000	0000	0001	0001		ADJUST	NONE
0000	0000	0000	1000	101011000	SHIFT	
0000	0000	0000	0101		ADJUST	DIGIT 0
0000	0000	0000	0010	1101011000	SHIFT	
0000	0000	0000	0010		ADJUST	NONE
0000	0000	0000	0001	01101011000	SHIFT	
			0001		ADJUST	NONE
			0000	101101011000	SHIFT	
			0000		ADJUST	NONE
			000	0101101011000	SHIFT	
			000		ADJUST	NONE
			00	00101101011000	SHIFT	
			00		ADJUST	NONE
			0	000101101011000	SHIFT	
			0		ADJUST	NONE
				0000101101011000	SHIFT	
					ADJUST	NONE

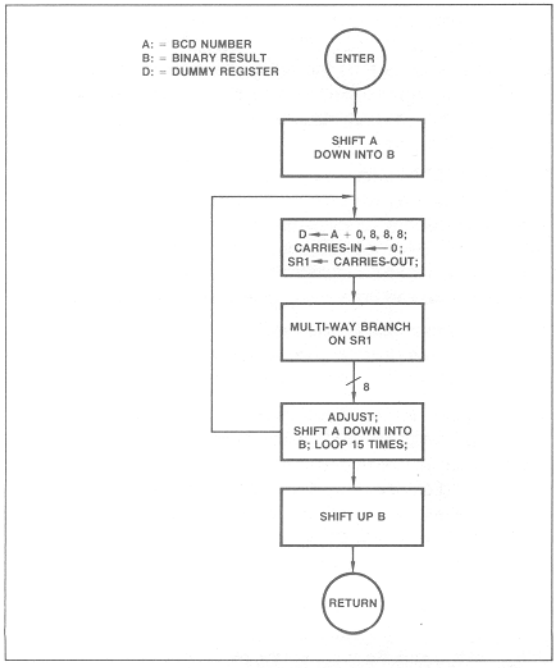


Figure 29. BCD to Binary Conversion (16 Bits to 14 Bits).

BINARY TO BCD CONVERSION

A method very similar to the one used for BCD to binary conversion may be used for binary to BCD conversion. The BCD number is created by shifting the binary number up, into a partial BCD result. The BCD number is adjusted to provide a multiplication by 2. The shift adjust process continues until the least significant binary bit is shifted into the BCD result.

The adjustment is needed when a 1 is shifted from the 8-bit position to the 1-bit position of the next digit. the value has increased from 8 to 10, instead of from 8 to 16. To correct this a 6 must be added to any digit that has a 1 shifted out of its 8-bit position. Alternately a 3 could be added before the shift to any digit that has a 1 in its 8-bit position.

Another correction is needed whenever an invalid BCD digit is encountered. If a number greater than 9 is detected in any digit a 10 must be subtracted from that digit and a 1 added to the next highest digit. The same correction can be accomplished if a 6 is added to the invalid digit after the shift. To correct before the shift a 3 is added to any digit which contains a 5, 6 or 7. These adjustments are summarized in Table 11. Both adjustments may be accomplished by adding a 3 to any digit which is greater than 4.

Table 12 shows an example conversion. The binary number is gradually shifted up and the BCD partial result adjusted. After 14 iterations the conversion is complete. A flow diagram for the algorithm is given in Figure 31.

```
A: = R0
B: = Q

                1   ENR & COUNT LOOP & CONT
                2   PAS R0, R0 LDRQ & SDDL & LDCT & CONST 15
LOOP:           3   ADD R1, R0, R0, DARB & ALUOFF & CONST 0888 & CZERO & ENSUR1 & CLSR2 & RPCT
                4   ALUOFF & MULTI 8WAY
                    ALIGN 8
                5   ALUOFF & CJRP & CNTR & GOTO EXIT
                6   SUB R0, R0, R0, LDRQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
                7   SUB R0, R0, R0, LDRQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
                8   SUB R0, R0, R0, LDRQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
                9   SUB R0, R0, R0, LDRQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               10   SUB R0, R0, R0, LDRQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               11   SUB R0, R0, R0, LDRQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               12   SUB R0, R0, R0, LDRQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
EXIT:          13   PAS R0, R0, R0, LURQ & SDUL & RTN
```

Figure 30.

TABLE 11.

Present #	Adjustment Before Shift	Reason
0000	NONE	–
0001	NONE	–
0010	NONE	–
0011	NONE	–
0100	NONE	–
0101	+3	Illegal BCD
0110	+3	
0111	+3	
1000	+3	Shift Thru Correction
1001	+3	
1010	+3	
1011	+3	
1100	+3	
1101	+3	
1110	+3	
1111	+3	

Initially the 14-bit binary number is left justified by two shift up operations. To start the loop the binary input, B, is shifted up, into the partial BCD result, A. The constant BBBB is added to A, with the carries-in forced to zero. The resulting carries-out are stored in status register SR1. A multi-way branch is used to enter the adjust table. The digits are adjusted depending on the result of the previous test. In the same instruction a shift is executed to prepare for the next test cycle. Additionally an end of loop test is used to provide an exit if 16 iterations of the loop are complete. Before the exit a fix-up cycle is used to cancel the extra shift executed in the loop. The microcode for this algorithm is given in Figure 32.

BCD ADD

One method of performing a 4-digit BCD add is to do a 16-bit binary add, with the carries-in forced to zero, and adjust the resulting sum. The adjustments are necessary to change invalid BCD digits to valid BCD digits. When an invalid digit is modified a carry to the next highest digit is generated. This could cause a

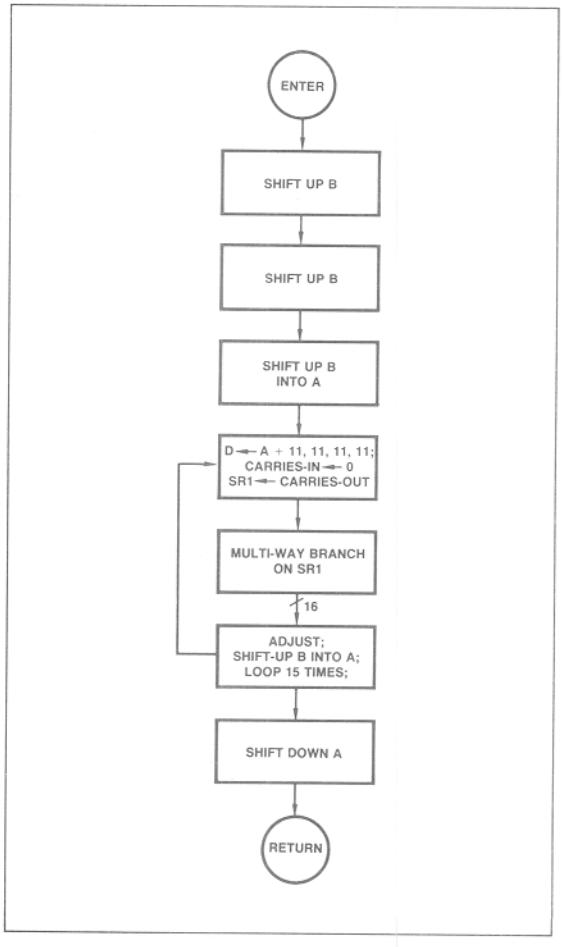


Figure 31. Binary to BCD Conversion (14 Bits to 16 Bits).

```
Q: =Binary Input
R0: = BCD Result

                1   SURL R0, R0 & SUL & CONT
                2   SURL R0, R0, & SUL & ENR & COUNT LOOP & CONT
                3   PAS R0, R0, ,LURQ & SDUL & LDCT & COUNT 15
LOOP:           4   ADD R1,R0, R0, DARB & ALUOFF & CONST BBBB & CZERO & ENSR1 & CLSR2 & RPCT
                5   ALUOFF & MULTI 16WAY
                    ALIGN 16
                6   ALUOFF & CJRP & GOTO EXIT
                7   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
                8   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
                9   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               10   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               11   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               12   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               13   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               14   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               15   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               16   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               17   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               18   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               19   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               20   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
               21   ADD R1, R0, R0, LURQ,DARB & CONST 0003 & CJRP & CNTR & GOTO EXIT
EXIT:          22   SDRL R0, R0, & SDL & RTN
```

Figure 32. Binary to BCD Conversion Microcode (14 Bits to 16 Bits).

TABLE 12.

Result						
Digit 3	Digit 2	Digit 1	Digit 0	Binary → BCD Conversion	Operation	
				00101101011000		
			0	0101101011000	SHIFT	
			0		ADJUST	NONE
			00	101101011000	SHIFT	
			00		ADJUST	NONE
			001	01101011000	SHIFT	
			001		ADJUST	NONE
			0010	1101011000	SHIFT	
			0010		ADJUST	NONE
		0	0101	101011000	SHIFT	
		0	1000		ADJUST	DIGIT 0
		01	0001	01011000	SHIFT	
		01	0001		ADJUST	NONE
		010	0010	1011000	SHIFT	
		010	0010		ADJUST	NONE
		0100	0101	011000	SHIFT	
		0100	1000		ADJUST	DIGIT 0
	0	1001	0000	11000	SHIFT	
	0	1100	0000		ADJUST	DIGIT 1
	01	1000	0001	1000	SHIFT	
	01	1011	0001		ADJUST	DIGIT 1
	011	0110	0011	000	SHIFT	
	011	1001	0011		ADJUST	DIGIT 1
	0111	0010	0110	00	SHIFT	
	1010	0010	1001		ADJUST	DIGIT 2
1	0100	0101	0010	0	SHIFT	
1	0100	1000	0010		ADJUST	DIGIT 1
10	1001	0000	0100		SHIFT	
10	1001	0000	0100		ADJUST	NONE
2	9	0	4			

previously valid digit to become invalid. The word must be checked and modified until all digits are valid (up to four modification cycles could be necessary).

Initially the two BCD numbers are added with the carries-in to each digit forced to zero. The carries out are saved. Next the hex number 6666 is added to the sum, with the carries-in forced to zero, and the resulting carries out are saved. This tests each digit for validity, a carry-out indicating an invalid BCD digit (greater than 9). If a carry was generated in either cycle a 6 is added to the invalid digit, with carries-in forced to zero, to create the valid BCD digit. Additionally a 1 must be added to the next highest digit to provide the BCD carry-out. Each time a digit is adjusted the carry-out may invalidate the next highest digit. Thus adjustment cycles must be followed by validity tests until all digits are valid. A flow diagram for this algorithm is given in Figure 33. The microcode for this algorithm is given in Figure 34.

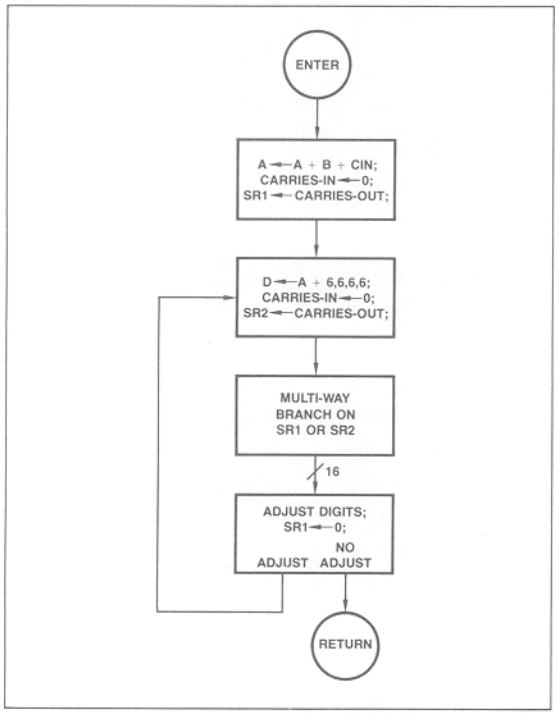


Figure 33. BCD Add.

```
A: = R1
B: = R0

 1 ADD R1,R1,R0 & CZERO & ENSR1 & CONT Z
 2 ADD R1,R1,R0,,DARB & ALUOFF & CZERO & ENSR2 & CONST 6666
 3 ALUOFF & MULTI 16WAY & RMAC
   ALIGN 16
 4 ALUOFF & JMP & GOTO EXIT & ENSR1
 5 ADD R1,R1,R0,,DARB & CONST 0016 & GOTO LOOP & CLRSR1
 6 ADD R1,R1,R0,,DARB & CONST 0160 & GOTO LOOP & CLRSR1
 7 ADD R1,R1,R0,,DARB & CONST 0176 & GOTO LOOP & CLRSR1
 8 ADD R1,R1,R0,,DARB & CONST 1600 & GOTO LOOP & CLRSR1
 9 ADD R1,R1,R0,,DARB & CONST 1616 & GOTO LOOP & CLRSR1
10 ADD R1,R1,R0,,DARB & CONST 1760 & GOTO LOOP & CLRSR1
11 ADD R1,R1,R0,,DARB & CONST 1776 & GOTO LOOP & CLRSR1
12 ALUOFF & JMP & GOTO LOOP & SMAC & CLRSR1
13 ADD R1,R1,R0,,DARB & CONST 0016 & GOTO LOOP & SMAC & CLRSR1
14 ADD R1,R1,R0,,DARB & CONST 0160 & GOTO LOOP & SMAC & CLRSR1
15 ADD R1,R1,R0,,DARB & CONST 0176 & GOTO LOOP & SMAC & CLRSR1
16 ADD R1,R1,R0,,DARB & CONST 1600 & GOTO LOOP & SMAC & CLRSR1
17 ADD R1,R1,R0,,DARB & CONST 1616 & GOTO LOOP & SMAC & CLRSR1
18 ADD R1,R1,R0,,DARB & CONST 1760 & GOTO LOOP & SMAC & CLRSR1
19 ADD R1,R1,R0,,DARB & CONST 1776 & GOTO LOOP & SMAC & CLRSR1

EXIT:
```

Figure 34. BCD Add Microcode.

SUMMARY

In this chapter, a detailed description of the Am2904 was presented, along with an example timing analysis. Several microcode algorithms were discussed to show how the Am2904 operates in a 2903 based CPU. As can be seen, the Am2904 provides a powerful, single-chip LSI solution to the shift multiplexer, status register, and carry multiplexer design portion of a CPU using either the Am2901B or the Am2903.

The Appendix includes a full microcode listing. The interested reader is encouraged to study these listings to gain a better understanding of the hardware organization (Appendix C). An additional microcode listing (Appendix B) gives the AMDASM™ definition file and source file for the microcode. The reader should study these listings while referring to the AMDASM Manual. (The Am2900 Family Data Book contains an AMDASM Reference Manual, document AM-PUB003, 4-78 FRODO.)

COMMENTS	ADDRESS	LABEL	CONSTANT																RE
			89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73
UNSIGN MULTIPLY																	
	0 1 0 C		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 0 D		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
TWO'S COMPLEMENT MULTIPLY	0 1 1 3		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 1 4		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 1 5		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
TWO'S COMPLEMENT DIVIDE	0 1 1 9		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 1 A		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 1 B		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 1 C		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 1 D	LOOP 1:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 1 E		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 1 F		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 0		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 1		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 2		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 3		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 4		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 5	SCALE 1:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 6		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 7	LOOP 2:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 8	SKIP 6:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 9		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 A		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 B		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 C	SKIP 3:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 D		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 E		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 2 F		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 3 0		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 3 1		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 3 2		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 3 3		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 3 4		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 3 5		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 3 6		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 3 7		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 3 8		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0 1 3 9		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

APPENDIX A

COMMENTS	ADDRESS	LABEL	CONSTANT 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70
SINGLE LENGTH NORMALIZE	0 1 3 C		X X X X X X X X X X X X X X X X X X X X
	0 1 3 D		X X X X X X X X X X X X X X X X X X X X
	0 1 3 E		X X X X X X X X X X X X X X X X X X X X
	0 1 3 F		X X X X X X X X X X X X X X X X X X X X
	0 1 4 0	AGAIN:	X X X X X X X X X X X X X X X X X X X X
	0 1 4 1		X X X X X X X X X X X X X X X X X X X X
	0 1 4 2		X X X X X X X X X X X X X X X X X X X X
DOUBLE LENGTH NORMALIZE	0 1 4 8		X X X X X X X X X X X X X X X X X X X X
	0 1 4 9		X X X X X X X X X X X X X X X X X X X X
	0 1 4 A		X X X X X X X X X X X X X X X X X X X X
	0 1 4 B		X X X X X X X X X X X X X X X X X X X X
	0 1 4 C	LOOP4:	X X X X X X X X X X X X X X X X X X X X
	0 1 4 D		X X X X X X X X X X X X X X X X X X X X
	0 1 4 E	JUMP1:	X X X X X X X X X X X X X X X X X X X X
	0 1 4 F		X X X X X X X X X X X X X X X X X X X X
BINARY ROOTS			
	0 1 5 2		X X X X X X X X X X X X X X X X X X X X
	0 1 5 3		X X X X X X X X X X X X X X X X X X X X
	0 1 5 4		X X X X X X X X X X X X X X X X X X X X
	0 1 5 5		0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 X X X X
	0 1 5 6		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 X X X X
	0 1 5 7		1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 X X X X
	0 1 5 8		0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X X X X
	0 1 5 9		0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 X X X X
	0 1 5 A		X X X X X X X X X X X X X X X X X X X X
	0 1 5 B	CYCLE:	X X X X X X X X X X X X X X X X X X X X
	0 1 5 C		X X X X X X X X X X X X X X X X X X X X
	0 1 5 D		X X X X X X X X X X X X X X X X X X X X
	0 1 5 E		X X X X X X X X X X X X X X X X X X X X
	0 1 5 F	POS:	X X X X X X X X X X X X X X X X X X X X
	0 1 6 0	CNT:	X X X X X X X X X X X X X X X X X X X X
	0 1 6 1		X X X X X X X X X X X X X X X X X X X X
	0 1 6 2		X X X X X X X X X X X X X X X X X X X X
	0 1 6 3		X X X X X X X X X X X X X X X X X X X X
	0 1 6 4	SUB:	X X X X X X X X X X X X X X X X X X X X

APPENDIX A

COMMENTS	ADDRESS	LABEL	CONSTANT 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70
SINGLE LENGTH NORMALIZE	0 1 3 C		X X X X X X X X X X X X X X X X X X X X
	0 1 3 D		X X X X X X X X X X X X X X X X X X X X
	0 1 3 E		X X X X X X X X X X X X X X X X X X X X
	0 1 3 F		X X X X X X X X X X X X X X X X X X X X
	0 1 4 0	AGAIN:	X X X X X X X X X X X X X X X X X X X X
	0 1 4 1		X X X X X X X X X X X X X X X X X X X X
	0 1 4 2		X X X X X X X X X X X X X X X X X X X X
DOUBLE LENGTH NORMALIZE	0 1 4 8		X X X X X X X X X X X X X X X X X X X X
	0 1 4 9		X X X X X X X X X X X X X X X X X X X X
	0 1 4 A		X X X X X X X X X X X X X X X X X X X X
	0 1 4 B		X X X X X X X X X X X X X X X X X X X X
	0 1 4 C	LOOP4:	X X X X X X X X X X X X X X X X X X X X
	0 1 4 D		X X X X X X X X X X X X X X X X X X X X
	0 1 4 E	JUMP1:	X X X X X X X X X X X X X X X X X X X X
	0 1 4 F		X X X X X X X X X X X X X X X X X X X X
BINARY ROOTS			
	0 1 5 2		X X X X X X X X X X X X X X X X X X X X
	0 1 5 3		X X X X X X X X X X X X X X X X X X X X
	0 1 5 4		X X X X X X X X X X X X X X X X X X X X
	0 1 5 5		0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 X X X X
	0 1 5 6		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 X X X X
	0 1 5 7		1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 X X X X
	0 1 5 8		0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X X X X
	0 1 5 9		0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 X X X X
	0 1 5 A		X X X X X X X X X X X X X X X X X X X X
	0 1 5 B	CYCLE:	X X X X X X X X X X X X X X X X X X X X
	0 1 5 C		X X X X X X X X X X X X X X X X X X X X
	0 1 5 D		X X X X X X X X X X X X X X X X X X X X
	0 1 5 E		X X X X X X X X X X X X X X X X X X X X
	0 1 5 F	POS:	X X X X X X X X X X X X X X X X X X X X
	0 1 6 0	CNT:	X X X X X X X X X X X X X X X X X X X X
	0 1 6 1		X X X X X X X X X X X X X X X X X X X X
	0 1 6 2		X X X X X X X X X X X X X X X X X X X X
	0 1 6 3		X X X X X X X X X X X X X X X X X X X X
	0 1 6 4	SUB:	X X X X X X X X X X X X X X X X X X X X

					CCU CONTROL FIELD														DEVICE ENABLE				SHARED CONTROL FIELD											
CT.		SRC.			MULTI-WAY				$\overline{\text{IR EN}}$	POLARITY	TEST SELECT				29811 NEXT ADDRESS				SHIFT EN	STATUS EN	$\overline{\text{IEN}}$	FIELD SELECT	INPUT/OUTPUT BRANCH COUNTER											
34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	X	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	X	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
0	0	0	0	0	0	0	0	0	1	X	1	1	1	1	1	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
0	0	X	X	X	0	0	0	0	1	0	0	0	0	0	0	0	0	1	X	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	X	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	X	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	1	1	X	0	0	1	0	0	0	1	0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	1	1	X	0	0	1	0	0	0	1	0	0	1	0	0	1	0	1
X	X	X	X	X	0	0	0	0	1	1	1	0	1	1	0	0	1	1	X	0	1	1	0	0	0	1	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
0	1	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	0	0	0	0	1	X	X	X	X	X	1	1	1	1	X	0	1	1	0	0	0	1	0	0	1	0	0	1	1	1
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	X	0	0	1	0	0	0	1	0	0	1	1	0	0	1	1
X	X	X	X	X	0	0	0	0	1	X	X	X	X	X	1	1	1	1	X	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	0	0	1	1	X	0	0	1	0	0	0	1	0	0	1	0	1	1	0	0
X	X	X	X	X	0	0	0	0	1	0	0	0	0	0	0	0	0	1	X	0	1	1	0	0	0	1	0	0	1	1	0	0	1	1
0	1	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	0	0	0	0	1	X	X	X	X	X	1	1	1	1	X	0	1	1	0	0	0	1	0	0	1	0	0	1	1	1
X	X	X	X	X	0	0	0	0	1	0	0	0	0	0	1	1	0	0	X	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
0	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	X	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	X	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0
X	X	X	X	X	0	0	0	0	1	X	X	X	X	X	1	1	1	1	X	0	1	1	0	0	0	1	0	0	0	1	1	0	0	1
1	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	0	0	0	0	1	1	1	0	1	1	0	0	1	1	X	0	1	1	0	0	0	1	0	0	1	1	0	1	1	1
0	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	0	0	0	0	1	X	X	X	X	X	1	1	1	1	X	0	1	1	0	0	0	1	0	0	1	1	1	0	0	0
1	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
0	0	0	1	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X

ENSR	$\overline{CZER}$	$\overline{CONST}$	$\overline{OECY}$	$\overline{OECT}$	$\overline{CEM}$	CARRY		SHIFT OP				INSTRUCTION OP						$\overline{OEY}$	B ADDRESS DEST.				B ADDRESS SRC.				A ADDRESS SRC.				DEST.				FUN	
								I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0														I_8	I_7	I_6	I_5		
71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X	X	X	X	1	1	1	1	0	1	1	0	0	1
X	X	X	X	X	X	0	0	0	1	1	0	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X	X	X	X	1	1	1	1	0	1	1	0	0	1
X	X	X	X	X	X	0	0	0	1	1	0	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
X	X	X	X	X	X	1	1	0	1	1	0	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	1	0	1	0	X	X	X	X	X	X	X	X	1	1	1	1	1	0
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	0	1	1	1	X	X	X	X	1	1	1	1	1	1	1	1	0	1
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	X	X	X	X	1	1	1	1	1	1	1	1	0	1
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	0	1	0	0	X	X	X	X	1	1	1	1	1	1	1	1	0	1
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	1	1	X	X	X	X	0	1	1	1	1	1	1	1	0	1
X	X	X	X	0	X	0	0	X	X	X	X	0	1	0	1	0	0	0	0	0	1	0	X	X	X	X	0	0	0	1	1	1	1	1	0	1
X	X	X	X	X	X	1	0	X	X	X	X	X	X	X	X	X	X	0	0	0	1	0	0	0	1	0	X	X	X	X	0	1	0	1	0	0
X	X	X	X	0	X	1	0	X	X	X	X	0	1	0	1	1	0	0	0	0	1	1	0	0	1	1	X	X	X	X	0	1	0	1	0	0
X	X	X	X	0	X	0	0	X	X	X	X	0	1	0	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	0	0	0	0	1	0	X	X	X	X	X	X	0	0	0	1	1	0	0	1	1	X	X	X	X	1	0	0	1	0	1
X	X	X	X	X	X	0	0	0	0	1	0	X	X	X	X	X	X	0	0	0	1	0	0	0	1	0	X	X	X	X	1	0	0	1	0	1
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X	X	X	X	0	1	0	0	0	1	1	0	0	1
X	X	X	X	X	X	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	0	1	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	0	0	1	1	0	0	1	0	1	1	0	0	0	0
X	X	X	X	0	X	0	0	X	X	X	X	0	1	1	0	1	0	0	X	X	X	X	X	X	X	X	0	1	0	0	0	1	1	0	0	1
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	0	0	0	0	0	0	X	X	X	X	X	X	0	0	0	1	0	0	0	1	0	X	X	X	X	0	0	0	1	0	1
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	0	0	1	1	1	1	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	1	1	1	1	0	1	0	0	0
X	X	X	X	X	X	1	0	1	1	1	1	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	1	1	1	1	1	0	0	0	0
X	X	X	X	X	X	1	0	0	0	1	1	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0	0
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	X	X	X	X	X	X	X	X	1	1	1	1	0	1
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	X	X	X	X	0	0	0	1	1	1	1	1	0	1
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	X	X	X	X	0	0	0	1	1	1	1	1	0	1
X	X	X	X	0	X	0	0	X	X	X	X	0	1	1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	0	0	0	1	1	0	X	X	X	X	X	X	0	0	0	0	1	X	X	X	X	0	0	0	1	0	0	1	1	0	0
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	0	0	0	1	1	0	X	X	X	X	X	X	0	0	0	0	1	X	X	X	X	0	0	0	1	0	0	1	0	0	1
X	X	X	X	X	X	0	0	X	X	X	X	X	X	X	X	X	X	0	0	1	0	0	X	X	X	X	X	X	X	X	1	1	1	1	0	1
X	X	X	X	X	X	0	1	X	X	X	X	X	X	X	X	X	X	0	1	0	1	0	X	X	X	X	1	0	1	0	1	1	1	1	0	1

Am2904 CONTROL FIELD | Am2903 CONTROL FIELD

COMMENTS	ADDRESS	LABEL	CONSTANT																	
			89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	RE
UNSIGN MULTIPLY																				
	0 1 0 C		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 0 D		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
TWO'S COMPLEMENT MULTIPLY	0 1 1 3		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 1 4		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 1 5		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
TWO'S COMPLEMENT DIVIDE	0 1 1 9		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 1 A		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 1 B		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 1 C		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 1 D	LOOP 1:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 1 E		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 1 F		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 0		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 1		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 2		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 3		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 4		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 5	SCALE 1:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 6		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 7	LOOP 2:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 8	SKIP 6:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 9		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 A		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 B		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 C	SKIP 3:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 D		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 E		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 2 F		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 3 0		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 3 1		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 3 2		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 3 3		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 3 4		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 3 5		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 3 6		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 3 7		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 3 8		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
	0 1 3 9		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

COMMENTS	ADDRESS	LABEL	CONSTANT																ENR
			89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73
	0	ENTER	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0
BCD TO BINARY	1		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1
CONVERSION	2		0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
ROUTINE	3	LOOP	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1
BRANCH	8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
TABLE	9		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	10		0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
	11		0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1
	12		0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1
	13		0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1
	14		0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	1
	15		0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	1
	16	EXIT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1
BINARY TO BCD	0	ENTER	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1
CONVERSION	1		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0
ROUTINE	2		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1
	3	LOOP	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1
	4		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1
BRANCH	16		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
TABLE	17		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	18		0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
	19		0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1
	20		0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1
	21		0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1
	22		0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	1
	23		0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	1
	24		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
	25		0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1
	26		0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	1
	27		0	0	1	1	0	0	0	0	0	0	1	1	0	0	1	1	1
	28		0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1
	29		0	0	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1
	30		0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	1
	31		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1
	32	EXIT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1

			Am2904 CONTROL FIELD															Am2903 CONTROL FIELD																		
ENSR	$\overline{CLRSR}$	$\overline{CONST}$	$\overline{OECY}$	$\overline{OECT}$	$\overline{CEM}$	CARRY		SHIFT OP				INSTRUCTION OP						$\overline{OEY}$	B ADDRESS DEST.				B ADDRESS SRC.				A ADDRESS SRC.				DEST.				FUN	
								I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0														I_8	I_7	I_6	I_5		
71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35
0	1	1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0	1	1	0	1	1	X	X	0	1	1	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	0	0	1	1	0	0	0	X	X	X	X	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0	1	0	0	1	1	0	1	0	1	1	1	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	1	1	0	1	0	1	1	1	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	1	1	0	1	0	1	1	1	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	1	1	0	1	0	1	1	1	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	1	1	0	1	0	1	1	1	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	1	1	0	1	0	1	1	1	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	1	1	0	1	0	1	1	1	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	1	1	0	1	0	1	1	1	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	1	0	1	1	X	X	1	1	1	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1
0	1	1	0	1	1	0	0	0	0	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1
0	1	1	0	1	1	0	0	0	0	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1
0	1	1	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1
0	0	0	1	1	0	0	0	X	X	X	X	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	0	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
0	1	1	0	1	1	X	X	0	1	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

					CCU CONTROL FIELD														DEVICE ENABLE				SHARED CONTROL FIELD											
…CT.		SRC.			MULTI-WAY				IR EN (overlined)	POLARITY	TEST SELECT				29811 NEXT ADDRESS				SHIFT EN (overlined)	STATUS EN (overlined)	IEN (overlined)	FIELD SELECT	INPUT/OUTPUT BRANCH COUNTER											
…4	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	0	0	0	0	1	X	X	X	X	X	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	1	1
1	1	1	0	0	0	0	0	0	1	X	X	X	X	X	1	0	0	1	1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	0	1	1	1	1	X	X	X	X	X	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0
0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	0	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	1	X	X	X	X	X	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	1	1
1	1	1	0	0	0	0	0	0	1	X	X	X	X	X	1	0	0	1	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	1	1	1	1	1	X	X	X	X	X	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X

APPENDIX A

COMMENTS	ADDRESS	LABEL	CONSTANT																				
																			$\overline{ENR}$	$\overline{CZERO}$	ENSR	$\overline{CLRSR}$	$\overline{CONST}$
			89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69
	0	ENTER	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	0	0	1	1
BCD ADD	1		0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0	1	1	0
ROUTINE	2		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	0	1	1
ADJUST	16	TAB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
TABLE	17		0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	0	0
	18		0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1	1	0	0	0
	19		0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0	1	1	0	0	0
	20		0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0
	21		0	0	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	0	0	0
	22		0	0	0	1	0	1	1	1	0	1	1	0	0	0	0	0	1	1	0	0	0
	23		0	0	0	1	0	1	1	1	0	1	1	1	0	1	1	0	1	1	0	0	0
	24		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
	25		0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	0	0
	26		0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1	1	0	0	0
	27		0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0	1	1	0	0	0
	28		0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0
	29		0	0	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	0	0	0
	30		0	0	0	1	0	1	1	1	0	1	1	0	0	0	0	0	1	1	0	0	0
	31		0	0	0	1	0	1	1	1	0	1	1	1	0	1	1	0	1	1	0	0	0
		EXIT																					

Am2904 CONTROL FIELD																Am2903 CONTROL FIELD																				
$\overline{OECY}$	$\overline{OECT}$	$\overline{CEM}$	CARRY		SHIFT OP				INSTRUCTION OP						$\overline{OEY}$	B ADDRESS DEST.				B ADDRESS SRC.				A ADDRESS SRC.				DEST.				FUNCT.				SR
					I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0														I_8	I_7	I_6	I_5					
68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
1	1	0	1	1	X	X	X	X	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	0
1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
0	1	0	X	X	X	X	X	X	0	0	0	0	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	1	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	0	0	0	X	X	X	X	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	0	0	0	X	X	X	X	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	0	0	0	X	X	X	X	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	0	0	0	X	X	X	X	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	0	0	0	X	X	X	X	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	0	0	0	X	X	X	X	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	0	0	0	X	X	X	X	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1
1	1	0	0	0	X	X	X	X	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1

C.	MULTI-WAY				$\overline{\text{IR EN}}$	POLARITY	TEST SELECT				29811 NEXT ADDRESS				$\overline{\text{SHIFT EN}}$	$\overline{\text{STATUS EN}}$	$\overline{\text{IEN}}$	FIELD SELECT	INPUT/OUTPUT BRANCH COUNTER											
	CCU CONTROL FIELD														DEVICE ENABLE				SHARED CONTROL FIELD											
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X
0	0	0	0	0	1	X	X	X	X	X	1	1	1	0	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X
X	1	1	1	1	1	X	X	X	X	X	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	X	X	X	X	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.1
CPUII DEFINITIONS

 ;ADVANCE MICRO DEVICES
 ; AM2903 AND AM2904 DEFINITION FILE FOR CPUII
 ;
 ;REV. OCTOBER 17, 1978


 WORD 90

 ;EQUATES

 MEM:     EQU H#F
 SPE:     EQU H#0
 OFF:     EQU B#1


 ;2903 DESTINATION MODIFIERS

 ADR:     EQU H#0
 LDR:     EQU H#1
 ADRQ:    EQU H#2
 LDRQ:    EQU H#3
 RPT:     EQU H#4
 LDQP:    EQU H#5
 QPT:     EQU H#6
 RQPT:    EQU H#7
 AUR:     EQU H#8
 LUR:     EQU H#9
 AURQ:    EQU H#A
 LURQ:    EQU H#B
 YBUS:    EQU H#C
 LUQ:     EQU H#D
 SINX:    EQU H#E


 ;CONSTANTS

 R0:      EQU H#0
 R1:      EQU H#1
 R2:      EQU H#2
 R3:      EQU H#3
 R4:      EQU H#4
 R5:      EQU H#5
 R6:      EQU H#6
 R7:      EQU H#7
 R8:      EQU H#8
 R9:      EQU H#9
 R10:     EQU H#A
 R11:     EQU H#B
 R12:     EQU H#C
 R13:     EQU H#D
 R14:     EQU H#E
 R15:     EQU H#F
```

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.1
CPUII DEFINITIONS


 ;2903 SOURCE MODIFIERS

 RADB:    EQU 3B#001
 RAQ:     EQU 3B#010
 DARB:    EQU 3B#100
 DADB:    EQU 3B#101
 DAQ:     EQU 3B#110

 ;I/O

 IOIN:    EQU 12H#01
 BIN:     EQU 12H#10
 BOUT:    EQU 12H#08
 LMAR:    EQU 12H#10
 YREG:    EQU 12H#02
 AOUT:    EQU 12H#40
 IOUT:    EQU 12H#04


 ;CARRY SELECT

 ONE:     EQU 2B#01
 CZ:      EQU 2B#10

 ;SUB DEFINITIONS


 SUB0:    SUB 36X,1B#0,4VX,4VX,4VX
 SUB1:    SUB 36X,1B#0,4VX,4VX,4VX,4VH#F
 SUB2:    SUB 36X,1B#0,4VX,4VX,4X,4VH#F
 SUB3:    SUB 3VB#000,16X,1B#0,13X
 SUB4:    SUB 36X,1B#0,12X
 SUB5:    SUB 44X,1B#0,15X
 SUB6:    SUB 44X,1B#0,15X
 SUB7:    SUB 26X
 SUB8:    SUB 36X,1B#0,4VX,8X,4VH#F
 SUB9:    SUB 36X,1B#0,4VX,4X,4VX,4VH#F
 SUB10:   SUB 36X,1B#0,4VX,4VX,4X
 SUB11:   SUB 24X,2VB#00,34X,4B#0000,1B#1,5X
 SUB12:   SUB 77X,1B#1,12VXH#0%
 SUB13:   SUB SPF,3VB#000,16X,1B#0,13X
 SUB14:   SUB 24X,2VB#00,34X,4B#0000,2B#10
 SUB15:   SUB 23X,1B#0,6X
 SUB16:   SUB SPF,3B#000,16X,1VB#0,13X
 SUB17:   SUB 54X
 SUB18:   SUB 22X,1B#0,7X
 SUB19:   SUB 16X,1B#0,13X
 SUB20:   SUB 1X,1VB#0,14X
 SUB21:   SUB 30X,H#B,20X




 ;CCU CONTROL
```

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.1
CPUII DEFINITIONS

 ACK:     DEF 66X,H#9,20X
 OBF:     DEF 66X,H#A,20X
 CNT:     DEF 66X,H#F,20X
 GRD:     DEF 66X,H#0,20X
 JZ:      DEF SUB11,H#0,SUB20
 CJS:     DEF SUB11,H#1,SUB20
 JMAP:    DEF SUB11,H#2,SUB20
 CJP:     DEF SUB11,H#3,SUB20
 PUSH:    DEF SUB11,H#4,SUB20
 JSRP:    DEF SUB11,H#5,SUB20
 CJV:     DEF SUB11,H#6,SUB20
 JRP:     DEF SUB11,H#7,SUB20
 RFCT:    DEF SUB11,H#8,SUB20
 RPCT:    DEF SUB11,H#9,SUB20
 CRTN:    DEF SUB11,H#A,SUB20
 CJPP:    DEF SUB11,H#B,SUB20
 LDCT:    DEF SUB11,H#C,SUB20
 LOOP:    DEF SUB11,H#D,SUB20
 CONT:    DEF SUB11,H#E,SUB20
 JP:      DEF SUB11,H#F,SUB20
 JSR:     DEF SUB14,H#01,SUB20
 RTN:     DEF SUB14,H#0A,SUB20

 ;SHARED CONTROL FIELD

 GOTC:    DEF SUB12
 COUNT:   DEF SUB12
 PUT:     DEF 77X,1B#0,12VXH#0%

 ;POLARITY CONTROL

 T:       DEF 65X,1B#1,24X
 F:       DEF 65X,1B#0,24X

 ;2903 CONTROL/FUNCTIONS

 IN:      DEF 36X,1B#1,H#F,8X,H#F,H#0,19X,1B#0,13X
 OUT:     DEF 36X,1B#0,8X,H#F,H#C,H#6,SUB3
 YOFF:    DEF 36X,1B#1,53X
 HIGH:    DEF SUB8,H#0,33#010,SUB19
 SRS:     DEF SUB1,H#1,SUB3
 SSR:     DEF SUB1,H#2,SUB3
 ADD:     DEF SUB1,H#3,SUB3
 PAS:     DEF SUB2,H#4,SUB3
 COMS:    DEF SUB2,H#5,SUB3
 PAR:     DEF SUB9,H#6,SUB3
 COMR:    DEF SUB9,H#7,SUB3
 LOW:     DEF SUB8,H#8,3X,SUB19
 CRAS:    DEF SUB1,H#9,SUB3
 XNRS:    DEF SUB1,H#A,SUB3
 XOR:     DEF SUB1,H#B,SUB3
 AND:     DEF SUB1,H#C,SUB3
 NOR:     DEF SUB1,H#D,SUB3
 NAND:    DEF SUB1,H#E,SUB3
 OR:      DEF SUB1,H#F,SUB3

 ;2903 SPECIAL FUNCTIONS
```

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.1
CPUII DEFINITIONS


UMUL:    DEF SUBØ,H#Ø,SUB16
TCM:     DEF SUBØ,H#2,SUB16
SMTC:    DEF SUB1Ø,H#5,SUB16
TCMC:    DEF SUBØ,H#6,SUB16
SLN:     DEF SUB1Ø,H#8,SUB16
DLN:     DEF SUBØ,H#A,SUB16
TDIV:    DEF SUBØ,H#C,SUB16
TDC:     DEF SUBØ,H#E,SUB16
INC:     DEF SUB1Ø,H#4,SUB16
SDQP:    DEF SUB4,H#5,4X,SUB3
SUQP:    DEF SUB4,H#D,4X,SUB3
LQPT:    DEF 36X,1B#Ø,8X,4VX,H#6,H#6,SUB3
RMOV:    DEF SUB2,H#4,SUB3
QMOV:    DEF 36X,1B#Ø,4VX,8X,MEM,H#4,3B#Ø1Ø,SUB19
SDRL:    DEF SUB1Ø,H#1,H#4,SUB3
SURL:    DEF SUB1Ø,H#9,H#4,SUB3


;2904 SHIFT CONTROL

SDDH:    DEF SUB7,H#3,SUB6
SDUH:    DEF SUB7,H#7,SUB5
SDDL:    DEF SUB7,H#6,SUB6
SDUL:    DEF SUB7,H#6,SUB5
RDD:     DEF SUB7,H#F,SUB6
RDU:     DEF SUB7,H#F,SUB5
SSXO:    DEF SUB7,H#E,SUB6
RSD:     DEF SUB7,H#A,SUB6
RSU:     DEF SUB7,H#A,SUB5
SUL:     DEF SUB7,H#2,SUB5
SUH:     DEF SUB7,H#3,SUB5
SDL:     DEF SUB7,H#Ø,SUB6
SDH:     DEF SUB7,H#1,SUB6
SDMS:    DEF SUB7,H#5,SUB6
SMS:     DEF SUB7,H#2,SUB6
SDDC:    DEF SUB7,H#7,SUB6
SDUC:    DEF SUB7,H#4,SUB5

;2904 MICRO INSTRUCTION CODES

RSTI:    DEF 3ØX,6B#ØØØØ11,SUB17
SWAP:    DEF 3 X,6B#ØØØØ1Ø,SUB17
SHLD:    EQU 1B#1


;2904 MACHINE INSTRUCTION CODES

LMA:     DEF SUB15,6B#ØØØØØØ,SUB17
RSTA:    DEF SUB15,6B#ØØØØ11,SUB17
SHOLD:   DEF 23X,1B#Ø,66X


;2904 MICRO STATUS SELECT
```

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.1
CPUII DEFINITIONS

 MIZ:     DEF SUB18,6B#010100,SUB21
 MIO:     DEF SUB18,6B#010110,SUB21
 MIC:     DEF SUB18,6B#011010,SUB21
 MIS:     DEF SUB18,6B#011110,SUB21

 ;2904 MACHINE STATUS SELECT

 MAZ:     DEF SUB18,6B#100100,SUB21
 MAO:     DEF SUB18,6B#100110,SUB21
 MAC:     DEF SUB18,6B#101010,SUB21
 MAS:     DEF SUB18,6B#101110,SUB21

 ;DEVICE DISABLE


 ALUCFF: DEF 7 X,1B#1,13X
 ALLOFF: DEF 7 X,3B#111,13X


 ;LOAD CONSTANT

 CONST:   DEF 16 VXH#0%,4X,1B#0,69X

 ;BCD STATUS REGISTER CONTROL

 ENR:     DEF 16X,1B#0,73X
 CLSR2:   DEF 17X,1B#0,72X
 ENSR1:   DEF 18X,1B#1,71X
 CZERO:   DEF 19X,1B#0,70X




 END

 TOTAL PHASE 1 ERRORS =    0
```

```
      ;ADVANCE MICRO DEVICES
      ;  AM2903 AND AM2904 CPUII SOURCE FILE

0100            ORG H#100
0100 INP:       ALUOFF & T & OBF & CJP & GOTO INP
0101            ALUOFF & PUSH
0102            IN & T & OBF & LOOP & PUT IOIN
0103            ALUOFF & RTN

0104 OUTP:      OUT & CONT & PUT YREG
0105            ALUOFF & PUSH
0106            ALUOFF & F & ACK & LOOP & PUT IOUT
0107            ALUOFF & PUSH
0108            ALUOFF & T & ACK & LOOP
0109            ALUOFF & RTN

010A USM:       LOW R1 & JSR & GOTO INP
010B            PAR R0,R15 & JSR & GOTO INP
010C            LQPT R15 & F & GRD & PUSH & COUNT 00E
010D            UMUL R1,R1,R0 & F & CNT & SDDL & RFCT
010E            PAR R15,R1 & JSR & GOTO OUTP
010F            QMOV R15 & JSR & GOTO OUTP
0110            JP & GOTO USM

0111 SM:        LOW R1 & JSR & GOTO INP
0112            PAR R0,R15 & JSR & GOTO INP
0113            LQPT R15 & F & GRD & PUSH & COUNT 00D
0114            TCM R1,R1,R0 & F & CNT & SDDL & RFCT
0115            TCMC R1,R1,R0 & SDDL & CONT CZ
0116            PAR R15,R1 & JSR & GOTO OUTP
0117            QMOV R15 & JSR & GOTO OUTP
0118            ALUOFF & JP & GOTO SM

0119 DIV:       LOW R10 & JSR & GOTO INP
011A            PAR R7,R15 & JSR & GOTO INP
011B            PAR R1,R15 & JSR & GOTO INP
011C            PAR R4,R15 & CONT
011D LOOP1:     PAR R3,R7 & CONT
011E            PAR R2,R1 & T & MIZ & CJP & GOTO ABORT
011F            SMTC R2,R2 & CONT CZ
0120            SMTC R3,R3 & T & MIO & CJP CZ & GOTO SCALE1
0121            ALUOFF & T & MIO & CJP & GOTO SKIP6
0122            SURL R3,R3 & SUL & CONT
0123            SURL R2,R2 & SUL & CONT
0124            ALUOFF & JP & GOTO LOOP2
0125 SCALE1:    LQPT R4 & JSR & GOTO SDIVD
0126            ALUOFF & JP LOOP1
0127 LOOP2:     SSR R15,R3,R2,YBUS & CONT ONE
0128 SKIP6:     LQPT R4 & F & MIC & CJP & GOTO SKIP3
0129            ALUOFF & JSR & GOTO SDIVD
012A            SDRL R2,R2 & SDL & CONT
012B            ALUOFF & JP & GOTO LOOP2
012C SKIP3:     ALUOFF & F & GRD & LDCT & COUNT 00C
012D            DLN R1,R1,R7 & T & GRD & RDU & PUSH
012E            TDIV R1,R1,R7 & F & CNT & RDU & RFCT CZ
012F            TDC R1,R1,R7 & SUH & CONT CZ
0130            QMOV R15 & JSR & GOTO OUTP
```

```
0131            PAR R15,R1 & JSR & GOTO OUTP
0132            ALUOFF & JP & GOTO DIV
0133 SDIVD:     PAR R1,R1 & CONT
0134            ALUOFF & T & MIS & CJP & GOTO NEG
0135            PAR R1,R1,ADRQ & SDDL & CONT
0136            ALUOFF & JP & GOTO RET
0137 NEG:       PAR R1,R1,ADRQ & SDDL & CONT
0138 RET:       QMOV R4 & CONT
0139            PAR R10,R10 & RTN ONE

013A SLNORM:    JSR & GOTO INP
013B            LQPT R15 & CONT
013C            SLN R2,R2,CFF & CONT & SHOLD
013D            MAZ & T & CJP & GOTO ABORT
013E            MAC & T & LOW RØ & CJP & GOTO END
013F            SLN R2,R2 & MAC & T & CJP ONE & GOTO END & SUB
0140 AGAIN:     SIN R2,R2 & MIO & F & CJP ONE & GOTO AGAIN & SUB
0141            SDQP & SMS & CONT
0142            SRS R2,R2,RØ & CONT
0143            QMOV R15 & JSR & GOTO OUTP
0144            PAR R15,R2 & JSR & GOTO OUTP
0145 END:       JP & GOTO SLNORM

0146 DLNORM:    JSR & GOTO INP
0147            LQPT R15 & JSR & GOTO INP
0148            DLN R15,R15,R15,CFF & CONT & SHOLD
0149            MAZ & T & CJP & GOTO ABORT
014A            LOW R2 & MAC & T & CJP & GOTO END2
014B            DLN R15,R15,R15 & SDUL & MAO & T & CJP & GOTO JUMP1
014C LOOP4:     DLN R15,R15,R15 & SDUL & MIO & T & CJP & GOTO JUMP1
014D            PAR R2,R2 & JP ONE & GOTO LOOP4
014E JUMP1:     PAR R2,R2 & CONT ONE
014F            SDRQ R15,R15 & SDMS & JSR & GOTO OUTP
0150            QMOV R15 & JSR & GOTO OUTP
0151 END2:      JP & GOTO DLNORM

0152 SQRT:      LOW R10 & CONT
0153            LOW RØ & JSR & GOTO INP
0154            PAR R1,R15 & CONT
0155            PAR R2,RØ,,DARB & CONST ØØØ5 & CONT
0156            PAR R3,RØ,,DARB & CONST ØØØ3 & CONT
0157            PAR R4,RØ,,DARB & CONST H#BFFF & CONT
0158            PAR R5,RØ,,DARB & CONST 4ØØØ & CONT
0159            PAR R6,RØ,,DARB & CONST ØØØ8 & CONT
015A            SRS RØ,R1,R5 & CONT & SHOLD
015B CYCLE:     AND R5,R5,R4 & CONT
015C            SDRL R4,R4 & MAS & CJP & GOTO END3
015D            SURI RØ,RØ & T & MAS & CJP & GOTO POS
015E            OR R5,R3 & JP & GOTO CNT
015F POS:       OR R5,R2 & CONT
0160 CNT:       SRS R6,R6,R10 & CONT
0161            SDRL R2,R2 & T & MIZ & CJP ,SHLD & GOTO END3
0162            SDRL R3,R3 & T & MAS & CJP & GOTO SUB
0163            ADD RØ,RØ,R5 & JP & GOTO CYCLE & SHOLD
0164 SUB:       SRS RØ,RØ,R5 & JP & GOTO CYCLE & SHOLD
0165 END3:      JP & GOTO SQRT
```

```
AMDOS/29 AMDASM MICRO ASSEMBLER, V1.1

 0166 ABORT:  ALUOFF & JP & GOTO ABORT
 0167         JP & GOTO DIV


      END
```

```
0100 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1110100011X01100 0100000000
0101 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX0100X01XXX XXXXXXXXXX
0102 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX11111XXXXXXX X1111000XXX0000
     1110101101X00000 0000000001
0103 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXX0000
     1000001010X01XXX XXXXXXXXXX
0104 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX0XXXXXXXX111 1110001100000000
     1XXXXX1110X00000 0000000010
0105 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXX0000
     1XXXXX0100X01XXX XXXXXXXXXX
0106 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXX0000
     1010011101X01000 0000000100
0107 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXX0000
     1XXXXX0100X01XXX XXXXXXXXXX
0108 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXX0000
     1110011101X01XXX XXXXXXXXXX
0109 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXX0000
     1000001010X01XXX XXXXXXXXXX
010A XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00001XXXXXXX X1111000XXX0000
     1000000001X00100 0100000000
010B XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00000XXXX111 1111101100000000
     1000000001X00100 0100000000
010C XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX0XXXXXXXX111 1011001100000000
     1000000100X00100 0000001110
010D XXXXXXXXXXXXXXXX XXXXXXXX00011011 XXXX0000010001000 0000000000000000
     1011111000000XXX XXXXXXXXXX
010E XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01111XXXX000 1111101100000000
     1000000001X00100 0100000100
010F XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01111XXXXXXX X1111010001000000
     1000000001X00100 0100000100
0110 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXX0000
     1XXXXX1111X0X100 0100001010
0111 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00001XXXXXXX X1111100XXX0000
     1000000001X00100 0100000000
0112 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00000XXXX111 1111101100000000
     1000000001X00100 0100000000
0113 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX0XXXXXXXX111 1011001100000000
     1000000100X00100 0000001101
0114 XXXXXXXXXXXXXXXX XXXXXXXX00011011 XXXX0000010001000 0001000000000000
     1011111000000XXX XXXXXXXXXX
0115 XXXXXXXXXXXXXXXX XXXXXXXX10011011 XXXX0000010001000 0011000000000000
     1XXXXX1110000XXX XXXXXXXXXX
0116 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01111XXXX000 1111101100000000
     1000000001X00100 0100000100
0117 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01111XXXXXXX X1111010001000000
     1000000001X00100 0100000100
0118 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXX0000
     1XXXXX1111X01100 0100010001
0119 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01010XXXXXXX X1111100XXX0000
     1000000001X00100 0100000000
011A XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00111XXXX111 1111101100000000
     1000000001X00100 0100000000
011B XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00001XXXX111 1111101100000000
     1000000001X00100 0100000000
011C XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX02100XXXX111 1111101100000000
     1XXXXX1110X00XXX XXXXXXXXXX
```

```
011D XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00011XXXX011 1111101100000000
     1XXXXX1110X00XXX XXXXXXXXXX
011E XXXXXXXXXXXXXXXX XXXXXXXX00XXXX01 0100000010XXX000 1111101100000000
     1110110011X00100 0101100110
011F XXXXXXXXXXXXXXXX XXXXXXXX10XXXXXX XXXX0001000010XXX X0101000000000000
     1XXXXX1110X00XXX XXXXXXXXXX
0120 XXXXXXXXXXXXXXXX XXXXXX0X10XXXX01 0110000110011XXX X0101000000000000
     1110110011X00100 0100100101
0121 XXXXXXXXXXXXXXXX XXXXXX0X00XXXX01 0110XXXXXXXXXXXX XXXXXXXXXXXX0000
     1110110011X01100 0100101000
0122 XXXXXXXXXXXXXXXX XXXXXXXX00001OXX XXXX0001100011XXX X1001010000000000
     1XXXXX1110200XXX XXXXXXXXXX
0123 XXXXXXXXXXXXXXXX XXXXXXXX00001OXX XXXX0001000010XXX X1001010000000000
     1XXXXX1110000XXX XXXXXXXXXX
0124 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX1111X01100 0100100111
0125 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX0XXXXXXXX010 0011001100000000
     1000000001X00100 0100110011
0126 XXXXXXXXXXXXXXXX XXXXXXXX01XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX1111X01XXX XXXXXXXXXX
0127 XXXXXXXXXXXXXXXX XXXXXXXX01XXXXXX XXXX01111100011001 0110000100000000
     1XXXXX1110X00XXX XXXXXXXXXX
0128 XXXXXXXXXXXXXXXX XXXXXX0X00XXXX01 10100XXXXXXXXX010 0011001100000000
     1010110011X00100 0100101100
0129 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1000000001X01100 0100110011
012A XXXXXXXXXXXXXXXX XXXXXXXX00000OXX XXXX0001000010XXX X0001010000000000
     1XXXXX1110000XXX XXXXXXXXXX
012B XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX1111X01100 0100100111
012C XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1000001100X01100 0000001100
012D XXXXXXXXXXXXXXXX XXXXXXXX001111XX XXXX0000100011011 1101000000000000
     1100000100J00XXX XXXXXXXXXX
012E XXXXXXXXXXXXXXXX XXXXXXXX101111XX XXXX0000100011011 1110000000000000
     1011111000J00XXX XXXXXXXXXX
012F XXXXXXXXXXXXXXXX XXXXXXXX100011XX XXXX0000100011011 1111000000000000
     1XXXXX1110000XXX XXXXXXXXXX
0130 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01111XXXXXXX X1111010001000000
     1000000001X00100 0100000100
0131 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01111XXXX000 1111101100000000
     1000000001X00100 0100000100
0132 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX1111X01100 0100011001
0133 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00001XXXX000 1111101100000000
     1XXXXX1110X00XXX XXXXXXXXXX
0134 XXXXXXXXXXXXXXXX XXXXXX0X00XXXX01 1110XXXXXXXXXXXX XXXXXXXXXXXX0000
     1110110011X01100 0100110111
0135 XXXXXXXXXXXXXXXX XXXXXXXX000110XX XXXX00001XXXX000 1001001100000000
     1XXXXX1110000XXX XXXXXXXXXX
0136 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX1111X01100 0100111000
0137 XXXXXXXXXXXXXXXX XXXXXXXX000110XX XXXX00001XXXX000 1001001100000000
     1XXXXX1110000XXX XXXXXXXXXX
0138 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00100XXXXXXX X1111010001000000
     1XXXXX1110X00XXX XXXXXXXXXX
0139 XXXXXXXXXXXXXXXX XXXXXXXX01XXXXXX XXXX01010XXXX101 0111101100000000
     1000001010X00XXX XXXXXXXXXX
```

```
013A XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1000000001X0X100 0100000000
013B XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX0XXXXXXXX111 1011001100000000
     1XXXXX1110X00XXX XXXXXXXXXX
013C XXXXXXXXXXXXXXXX XXXXXXX000XXXXXX XXXX0001000010XXX X1000000000000000
     1XXXXX1110X01XXX XXXXXXXXXX
013D XXXXXXXXXXXXXXXX XXXXXX0X00XXXX10 0100XXXXXXXXXXXX XXXXXXXXXXXX0000
     1110110011X0X100 0101100110
013E XXXXXXXXXXXXXXXX XXXXXX0X00XXXX10 1010000000XXXXXX X1111100XXX00000
     1110110011X00100 0101000101
013F XXXXXXXXXXXXXXXX XXXXXX0X01001010 0110000100010XXX X1000000000000000
     1110110011000100 0101000101
0140 XXXXXXXXXXXXXXXX XXXXXX0X01001001 0110000100010XXX X1000000000000000
     1010110011000100 0101000000
0141 XXXXXXXXXXXXXXXX XXXXXXXX00001XXX XXXX0XXXXXXXXXXX X0101XXXX0000002
     1XXXXX1110X00XXX XXXXXXXXXX
0142 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX0001000010000 0111100010000000
     1XXXXX1110X00XXX XXXXXXXXXX
0143 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01111XXXXXXX X1111010001000000
     1000000001X00100 0100000100
0144 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01111XXXX001 0111101100000000
     1000000001X00100 0100000100
0145 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX1111X0X100 0100111010
0146 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1000000001X0X100 0100000000
0147 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX0XXXXXXXX111 1011001100000000
     1000000001X00100 0100000000
0148 XXXXXXXXXXXXXXXX XXXXXXX000XXXXXX XXXX011111111111 1101000000000000
     1XXXXX1110X01XXX XXXXXXXXXX
0149 XXXXXXXXXXXXXXXX XXXXXX0X00XXXX10 0100XXXXXXXXXXXX XXXXXXXXXXXX0000
     1110110011X0X100 0101100110
014A XXXXXXXXXXXXXXXX XXXXXX0X00XXXX10 1010000010XXXXXX X1111100XXX00000
     1110110011X00100 0101010001
014B XXXXXXXXXXXXXXXX XXXXXX0X00011010 0110011111111111 1101000000000000
     1110110011000100 0101001110
014C XXXXXXXXXXXXXXXX XXXXXX0X00011001 0110011111111111 1101000000000000
     1110110011000100 0101001110
014D XXXXXXXXXXXXXXXX XXXXXXXX01XXXXXX XXXX00010XXXX001 0111101100000000
     1XXXXX1111X00100 0101001100
014E XXXXXXXXXXXXXXXX XXXXXXXX01XXXXXX XXXX00010XXXX001 0111101100000000
     1XXXXX1110X00XXX XXXXXXXXXX
014F XXXXXXXXXXXXXXXX XXXXXXXX000101XX XXXX011111111XXX X0011010000000000
     1000000001000100 0100000100
0150 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01111XXXXXXX X1111010001000000
     1000000001X00100 0100000100
0151 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX1111X0X100 0101000110
0152 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX01010XXXXXXX X1111100XXX00000
     1XXXXX1110X00XXX XXXXXXXXXX
0153 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00000XXXXXXX X1111100XXX00000
     1000000001X00100 0100000000
0154 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXXX XXXX00001XXXX111 1111101100000000
     1XXXXX1110X00XXX XXXXXXXXXX
0155 0000000000000101 XXXX0XXX00XXXXXX XXXX00010XXXX000 0111101101000000
     1XXXXX1110X00XXX XXXXXXXXXX
0156 0000000000000011 XXXX0XXX00XXXXXX XXXX00011XXXX000 0111101101000000
     1XXXXX1110X00XXX XXXXXXXXXX
```

```
0157 1011111111111111 XXXX0XXX00XXXXX XXXX00100XXXX000 0111101101000000
     1XXXXX1110X03XXX XXXXXXXXX
0158 0100000000000000 XXXX0XXX00XXXXX XXXX00101XXXX000 0111101101000000
     1XXXXX1110X00XXX XXXXXXXXX
0159 0000000000001000 XXXX0XXX00XXXXX XXXX00110XXXX000 0111101101000000
     1XXXXX1110X00XXX XXXXXXXXX
015A XXXXXXXXXXXXXXXX XXXXXX7000XXXXX XXXX0000000001010 1111100010000000
     1XXXXX1110X00XXX XXXXXXXXX
015B XXXXXXXXXXXXXXXX XXXXXXXX00XXXXX XXXX00101010101010 0111111000000000
     1XXXXX1110X00XXX XXXXXXXXX
015C XXXXXXXXXXXXXXXX XXXXXX0X00XXX10 1110001000100XXX X0001010000000000
     1X1011001100100 0101100101
015D XXXXXXXXXXXXXXXX XXXXXX0X00XXX10 1110000000000XXX X1001010000000000
     1110110011X00100 0101011111
015E XXXXXXXXXXXXXXXX XXXXXXXX00XXXXX XXXX00101001011XXX X1111111100000000
     1XXXXX1111X00100 0101100000
015F XXXXXXXXXXXXXXXX XXXXXXXX00XXXXX XXXX00101001010XXX X1111111100000000
     1XXXXX1110X00XXX XXXXXXXXX
0160 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXX XXXX00110011010101 0111100010000000
     1XXXXX1110X00XXX XXXXXXXXX
0161 XXXXXXXXXXXXXXXX XXXXXX0X00XXX01 0100001001001XXX X0001010000000000
     1110110011X10100 0101100101
0162 XXXXXXXXXXXXXXXX XXXXXX0X00XXX10 1110000110011XXX X0001010000000000
     1110110011X00100 0101100100
0163 XXXXXXXXXXXXXXXX XXXXXXX000XXXXX XXXX0000000000010 1111100110000000
     1XXXXX1111X00100 0101011011
0164 XXXXXXXXXXXXXXXX XXXXXXX000XXXXX XXXX0000000000010 1111100010000000
     1XXXXX1111X00100 0101011011
0165 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX1111X0X100 0101010010
0166 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX1111X01100 0101100110
0167 XXXXXXXXXXXXXXXX XXXXXXXX00XXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX0000
     1XXXXX1111X0X100 0100011001
```

Am2903 MNEMONICS

I_0 FUNCTION

RAMB	RAM B – OUTPUT
Q	Q REGISTER
SPF	SPECIAL FUNCTIONS

ALU Functions

SPF	Special Functions	
HIGH	Fi = HIGH	HIGHS
SRS	Subtract R from S	$S - R - 1 + C_n$
SSR	Subtract S from R	$R - S - 1 + C_n$
ADD	Add R and S	$R + S + C_n$
PAS	Pass S	$S + C_n$
COMS	2's Complement S	$\overline{S} + C_n$
PAR	Pass R	$R + C_n$
COMR	2's Complement R	$\overline{R} + C_n$
LOW	Fi = LOW	LOW'S
CRAS	Complement R AND with S	$\overline{RAS}$
XNRS	Exclusive NOR R with S	$\overline{RVS}$
XOR	Exclusive OR R with S	RVS
AND	AND R with S	RAS
NOR	NOR R with S	$\overline{RVS}$
NAND	NAND R with S	$\overline{RAS}$
OR	OR R with S	RVS

ALU Destination Control

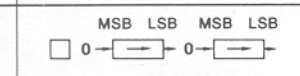
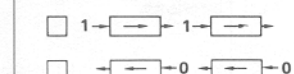
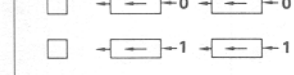
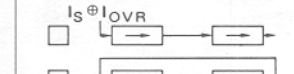
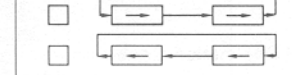
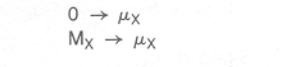
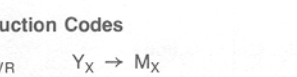
ADR	Arithmetic Shift Down, Results Into RAM
LDR	Logical Shift Down, Results Into RAM
ADRQ	Arithmetic Shift Down, Results Into RAM and Q Register
LDRQ	Logical Shift Down, Results Into RAM and Q Register
RPT	Results Into RAM, Generate Parity
* LDQP	Logical Shift Down Contents of Q Register, Generate Parity
* QPT	Results Into Q Register, Generate Parity
RQPT	Results Into RAM and Q Register, Generate Parity
AUR	Arithmetic Shift Up, Results Into RAM
LUR	Logical Shift Up, Results Into RAM
AURQ	Arithmetic Shift Up, Results Into RAM and Q Register
LURQ	Arithmetic Shift Up, Results Into RAM and Q Register
* YBUS	Results to Y BUS Only
* LUQ	Logical Shift Up the Contents of the Q Register
SINX	Sign Extend
REG	Results to RAM, Sign Extend

* = $\overline{WRITE}$ = H

Special Functions

UMUL	Unsigned Multiply
TCM	Two's Complement Multiply
INC	Increment by One or Two
SMTC	Sign Magnitude ⟷ Two's Complement
TCMC	Two's Complement Multiply Last Step
SLN	Single Length Normalize
DLN	Double Length Normalize
TDN	Two's Complement Multiply Division
TDC	Two Complement Division Correction

SHIFT INSTRUCTIONS

	I_{10}	I_9	I_8	I_7	I_6	M_C RAM Q	SIO_0	SIO_n	QIO_0	QIO_n	Loaded into M_C
SDL	0	0	0	0	0		Z	0	Z	0	
SUH	0	0	0	0	1		Z	1	Z	1	
SUL	1	0	0	1	0		0	Z	0	Z	
SUH	1	0	0	1	1		1	Z	1	Z	
SDDH	0	0	0	1	1		Z	1	Z	SIO_0	
SDDL	0	0	1	1	0		Z	0	Z	SIO_0	
SDUL	1	0	1	1	0		QIO_n	Z	0	Z	
SDUH	1	0	1	1	1		QIO_n	Z	1	Z	
RSD	0	1	0	1	0		Z	SIO_0	Z	QIO_0	
RSU	1	1	0	1	0		SIO_n	Z	QIO_n	Z	
SSXO	0	1	1	1	0		Z	$I_N \oplus I_{OVR}$	Z	SIO_0	
RDD	0	1	1	1	1		Z	QIO_0	Z	SIO_0	
RDU	1	1	1	1	1		QIO_n	Z	SIO_n	Z	
SDMS	0	0	1	0	1		Z	M_N	Z	SIO_0	
SMS	0	0	0	1	0		Z	0	Z	M_N	SIO_0
SDDC	0	0	1	1	1		Z	0	Z	SIO_0	QIO_0
SDUC	1	0	1	0	0		QIO_n	Z	0	Z	SIO_n

Microstatus Register Instruction Codes

RSTI	Reset μSR	$0 \rightarrow \mu_X$
SWAP	Register Swap	$M_X \rightarrow \mu_X$
SHLD	Hold Status	

Microregister Condition Code Output (CT)

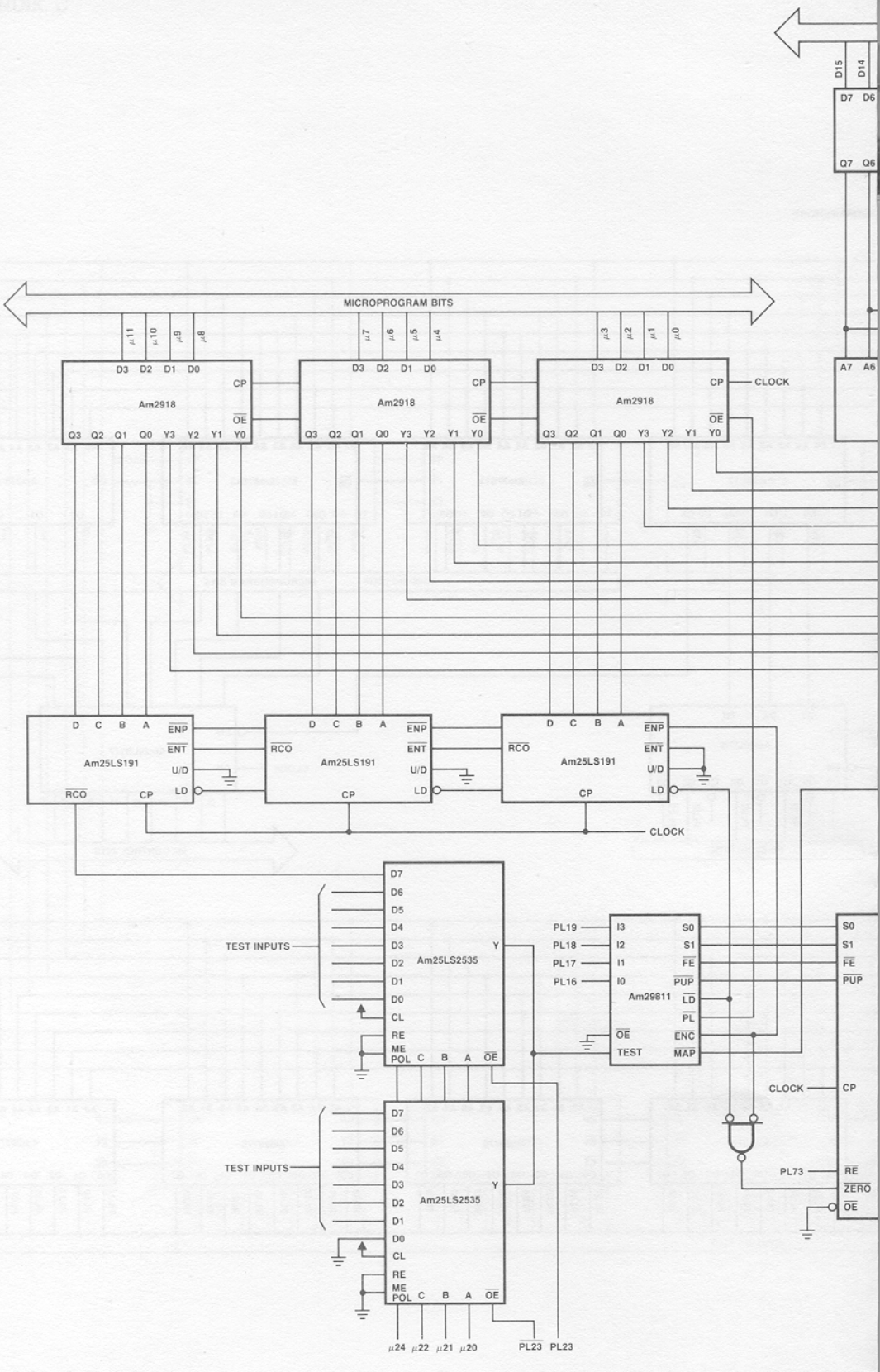
MIZ	Zero	$\mu_Z \rightarrow C_T$
MIO	Overflow	$\mu_{OVR} \rightarrow C_T$
MIC	Carry	$\mu_C \rightarrow C_T$
MIS	Sign	$\mu_N \rightarrow C_T$

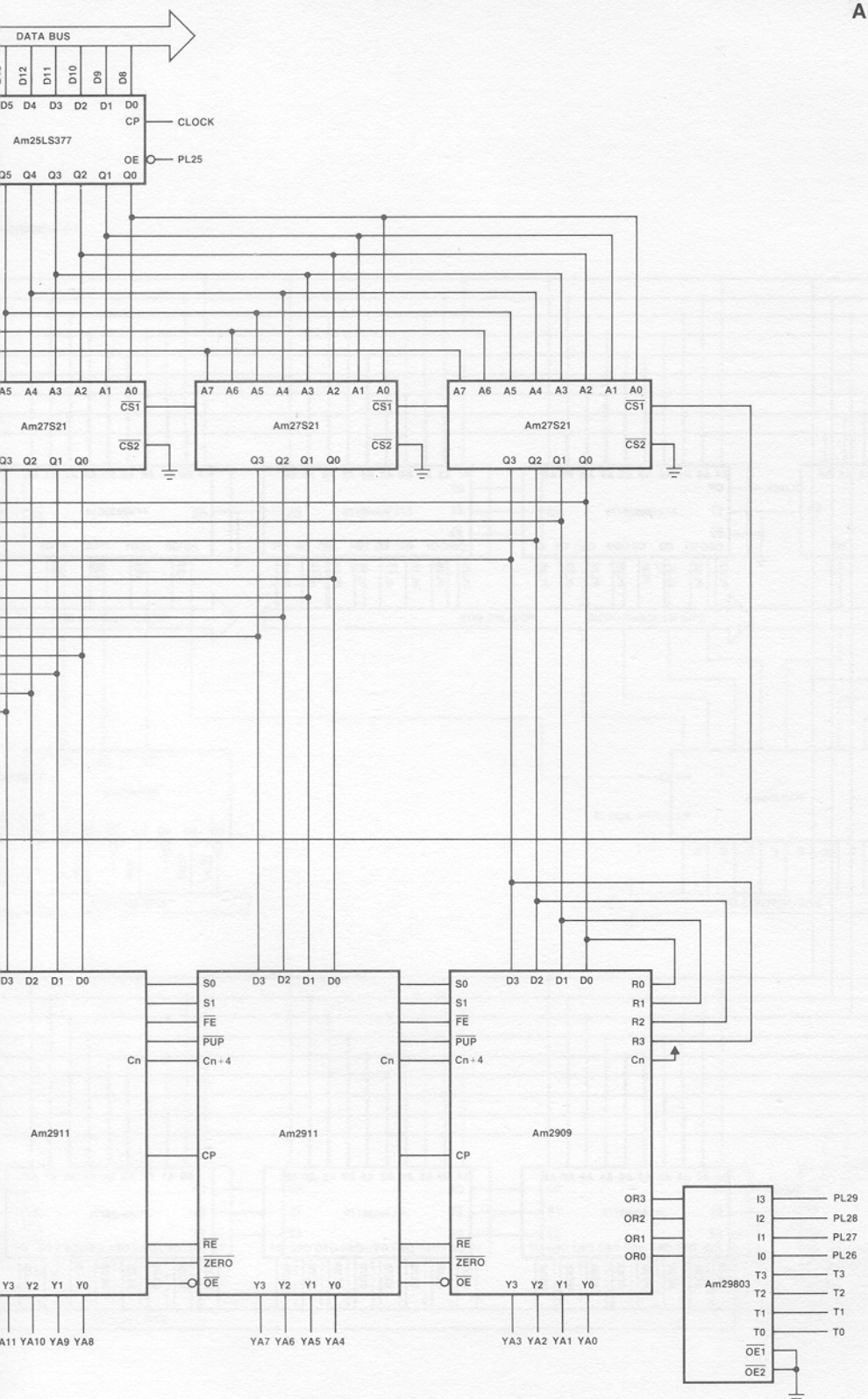
Machine Status Register Instruction Codes

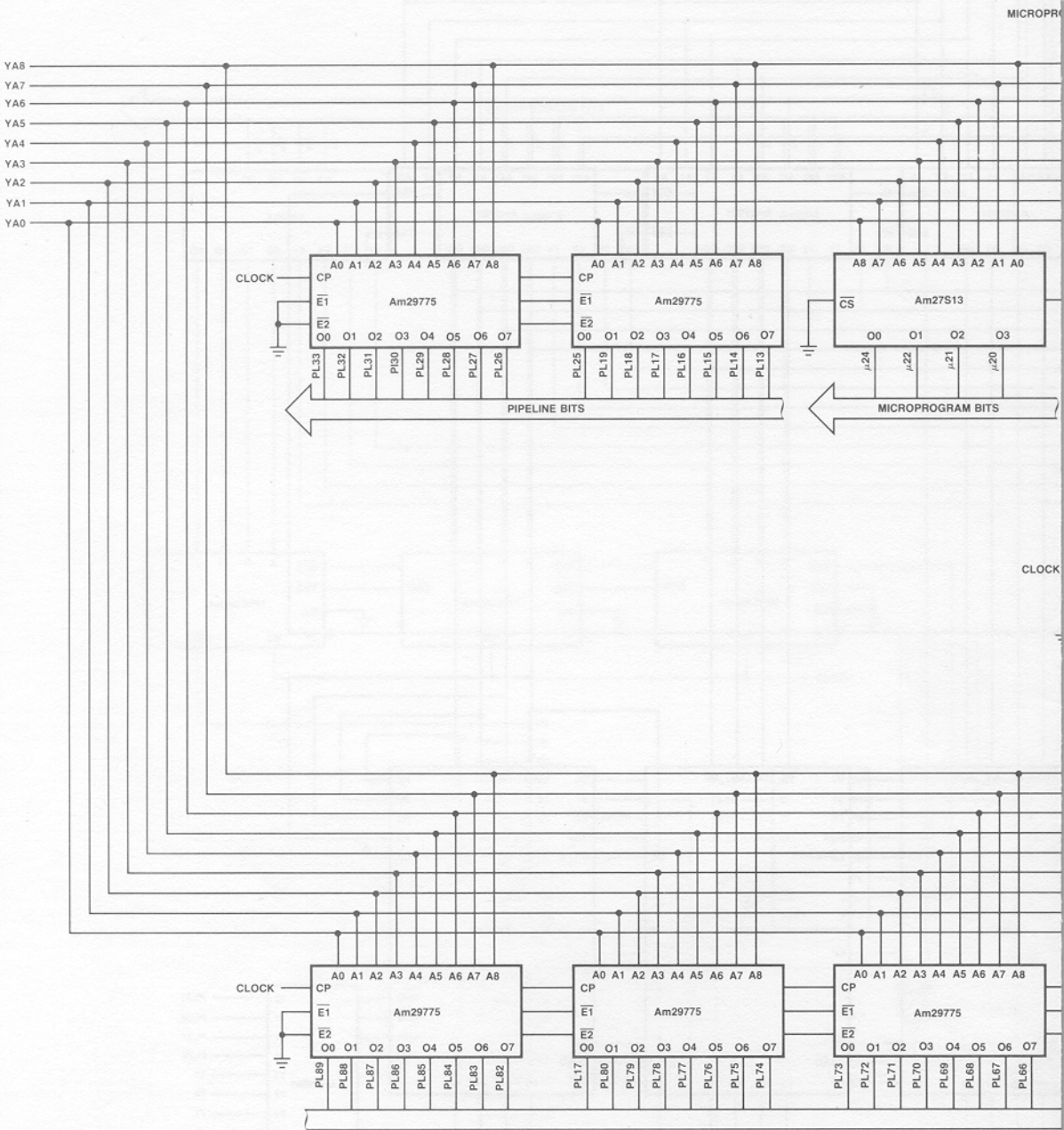
LMA	Load Y_Z, Y_C, Y_N, Y_{OVR} To MSR	$Y_X \rightarrow M_X$
RSTA	Reset MSR	$0 \rightarrow M_X$
SHOLD	Hold Status	

Machine Register Condition Code Output (CT)

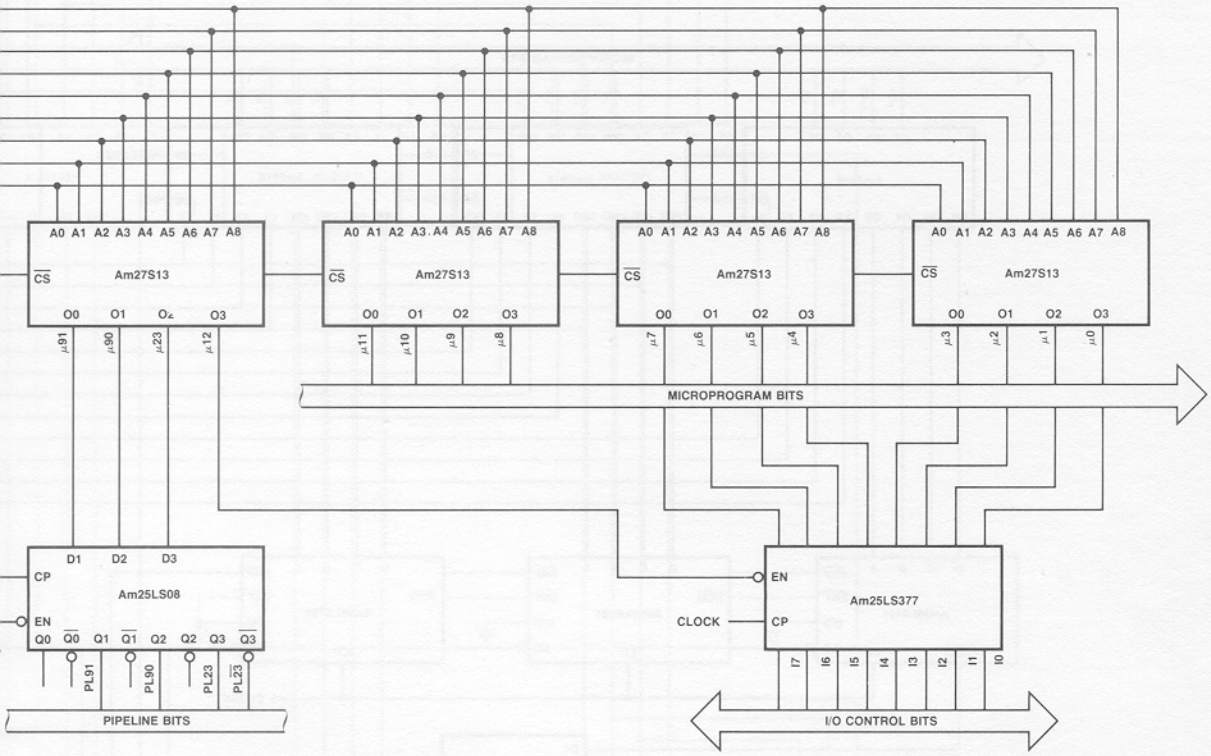
MAZ	Zero	$M_Z \rightarrow C_T$
MAO	Overflow	$M_{OVR} \rightarrow C_T$
MAC	Carry	$M_C \rightarrow C_T$
MAS	Sign	$M_N \rightarrow C_T$

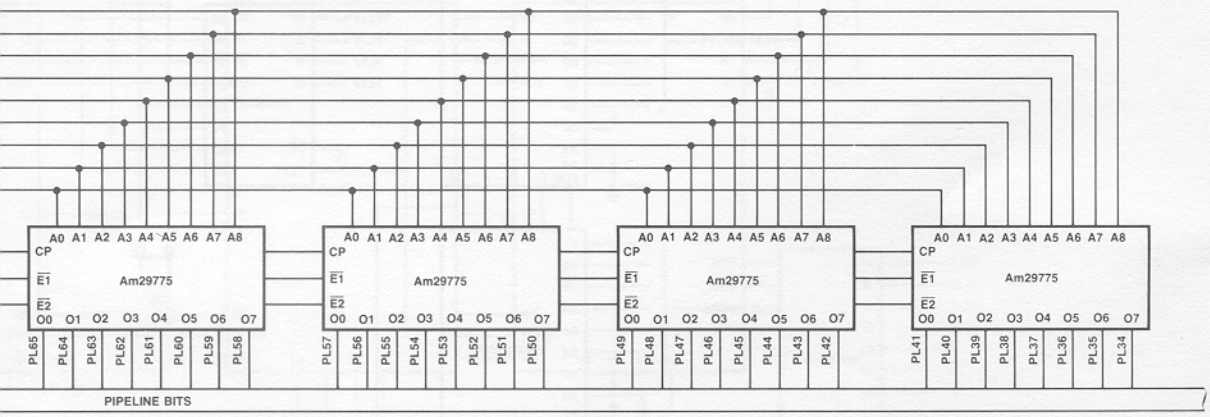

MICROPROGRAM BITS
μ11 μ10 μ9 μ8
μ7 μ6 μ5 μ4
μ3 μ2 μ1 μ0
Am2918
D3 D2 D1 D0
CP
OE
Q3 Q2 Q1 Q0 Y3 Y2 Y1 Y0
CLOCK
Am25LS191
D C B A
ENP
ENT
U/D
LD
RCO
CP
CLOCK
Am25LS2535
D7 D6 D5 D4 D3 D2 D1 D0
CL
RE
ME
POL C B A OE
Y
TEST INPUTS
Am29811
PL19 PL18 PL17 PL16
I3 I2 I1 I0
S0 S1 FE PUP LD PL ENC MAP
OE TEST
PL73
ZERO
μ24 μ22 μ21 μ20
PL23
D15 D14
D7 D6
Q7 Q6
A7 A6

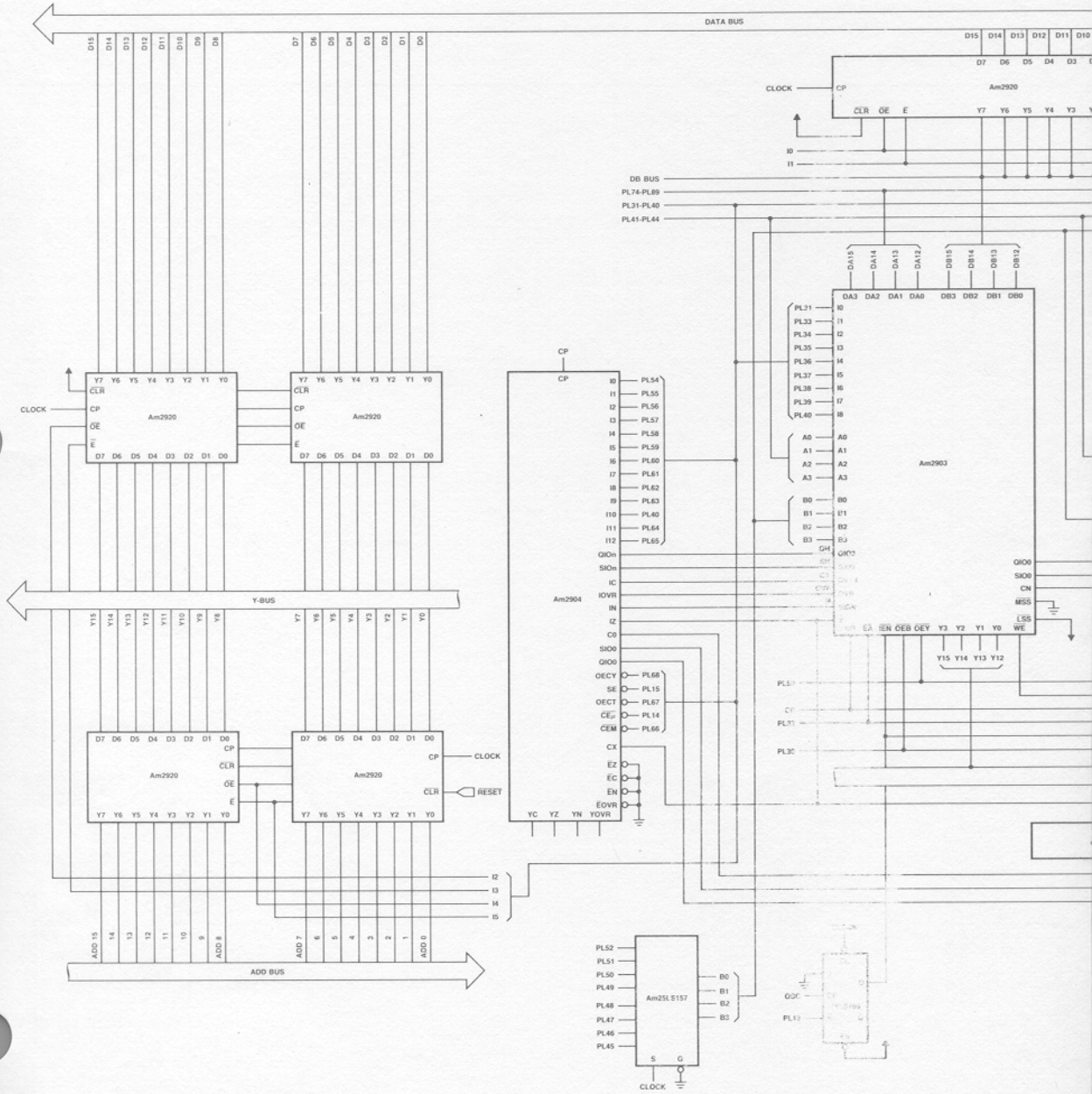

DATA BUS
D12
D11
D10
D9
D8
D5 D4 D3 D2 D1 D0
CP
CLOCK
Am25LS377
OE
PL25
Q5 Q4 Q3 Q2 Q1 Q0
A5 A4 A3 A2 A1 A0
CS1
CS2
Am27S21
Q3 Q2 Q1 Q0
A7 A6 A5 A4 A3 A2 A1 A0
D3 D2 D1 D0
S0
S1
FE
PUP
Cn+4
Cn
Am2911
RE
ZERO
OE
Y3 Y2 Y1 Y0
A11 YA10 YA9 YA8
YA7 YA6 YA5 YA4
Am2909
R0
R1
R2
R3
OR3
OR2
OR1
OR0
YA3 YA2 YA1 YA0
Am29803
I3
I2
I1
I0
T3
T2
T1
T0
OE1
OE2
PL29
PL28
PL27
PL26

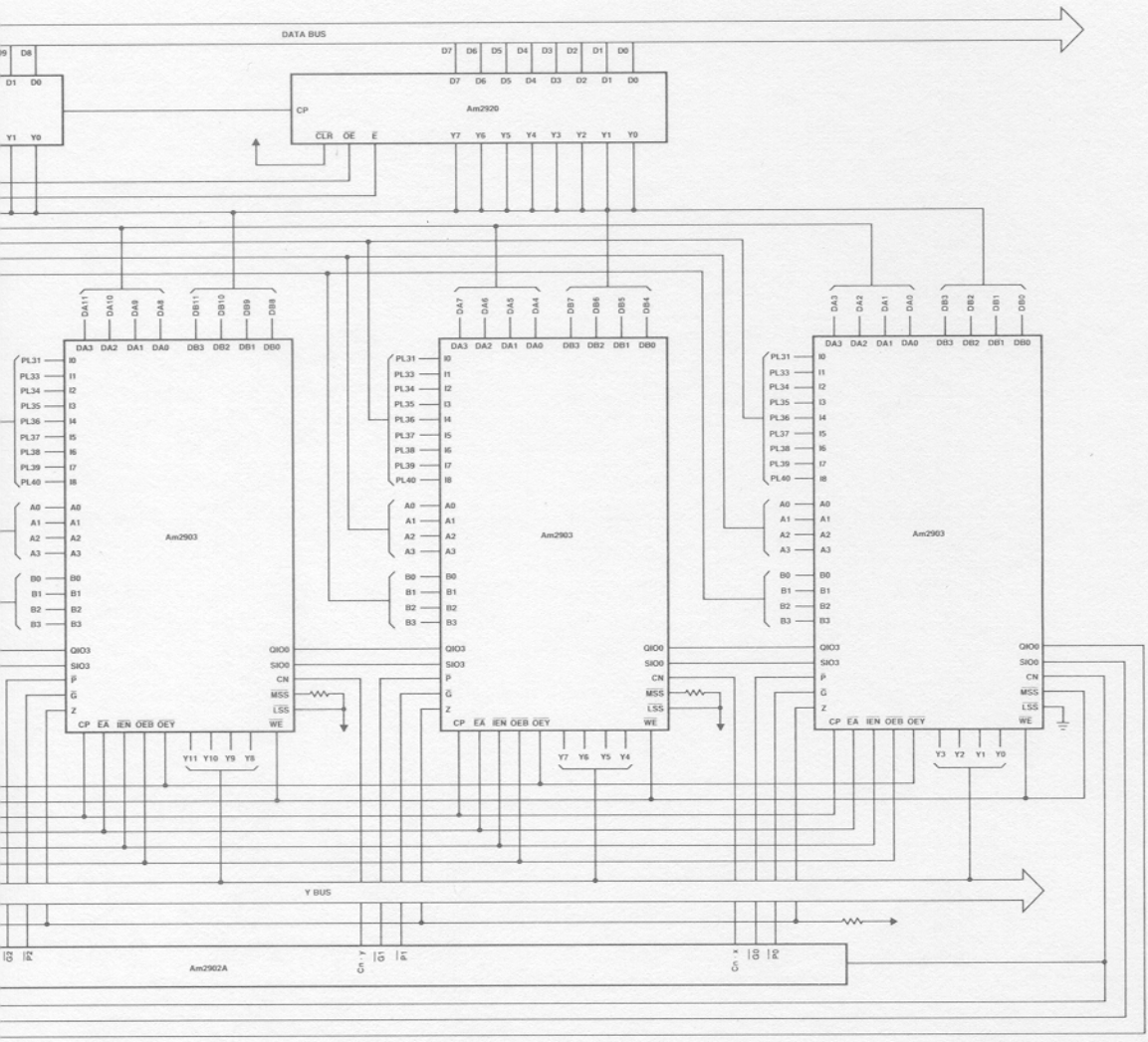

MICROPR
YA8
YA7
YA6
YA5
YA4
YA3
YA2
YA1
YA0
CLOCK
A0 A1 A2 A3 A4 A5 A6 A7 A8
CP
E1
E2
Am29775
O0 O1 O2 O3 O4 O5 O6 O7
PL33 PL32 PL31 PI30 PL29 PL28 PL27 PL26
PL25 PL19 PL18 PL17 PL16 PL15 PL14 PL13
PIPELINE BITS
A8 A7 A6 A5 A4 A3 A2 A1 A0
CS
Am27S13
O0 O1 O2 O3
μ24 μ22 μ21 μ20
MICROPROGRAM BITS
CLOCK
PL89 PL88 PL87 PL86 PL85 PL84 PL83 PL82
PL17 PL80 PL79 PL78 PL77 PL76 PL75 PL74
PL73 PL72 PL71 PL70 PL69 PL68 PL67 PL66

GRAM MEMORY


A0 A1 A2 A3 A4 A5 A6 A7 A8
CS
Am27S13
O0 O1 O2 O3
μ91 μ90 μ23 μ12
μ11 μ10 μ9 μ8
μ7 μ6 μ5 μ4
μ3 μ2 μ1 μ0
MICROPROGRAM BITS
D1 D2 D3
CP
Am25LS08
EN
Q0 Q0 Q1 Q1 Q2 Q2 Q3 Q3
PL91 PL90 PL23 PL23
PIPELINE BITS
EN
Am25LS377
CLOCK
CP
I7 I6 I5 I4 I3 I2 I1 I0
I/O CONTROL BITS


A0 A1 A2 A3 A4 A5 A6 A7 A8
CP
E1
E2
Am29775
O0 O1 O2 O3 O4 O5 O6 O7
PL65 PL64 PL63 PL62 PL61 PL60 PL59 PL58
PL57 PL56 PL55 PL54 PL53 PL52 PL51 PL50
PL49 PL48 PL47 PL46 PL45 PL44 PL43 PL42
PL41 PL40 PL39 PL38 PL37 PL36 PL35 PL34
PIPELINE BITS


DATA BUS
Am2920
CLOCK
DB BUS
PL74-PL89
PL31-PL40
PL41-PL44
Am2903
Am2904
Y-BUS
ADD BUS
RESET
Am25LS157


DATA BUS
Am2920
CP
CLR OE E
D7 D6 D5 D4 D3 D2 D1 D0
Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
Am2903
DA3 DA2 DA1 DA0 DB3 DB2 DB1 DB0
PL31 PL33 PL34 PL35 PL36 PL37 PL38 PL39 PL40
I0 I1 I2 I3 I4 I5 I6 I7 I8
A0 A1 A2 A3
B0 B1 B2 B3
QIO3 SIO3 P G Z
QIO0 SIO0 CN MSS LSS WE
CP EA IEN OEB OEY
Y11 Y10 Y9 Y8
Y7 Y6 Y5 Y4
Y3 Y2 Y1 Y0
Y BUS
Am2902A

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