AMI 7300 MICROPROCESSOR
MICROINSTRUCTION SUMMARY

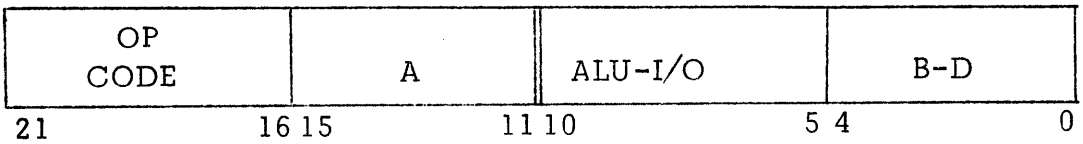
American Micro-systems, Inc.
Microprocessor Group
3800 Homestead Road
Santa Clara, CA 95051

February 1973

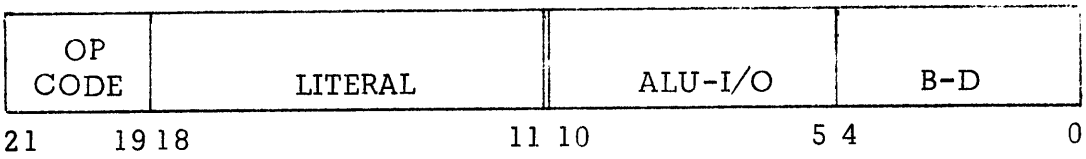
TABLE OF CONTENTS

1. MICROINSTRUCTION FORMATS

FORMAT 1 - REGISTER CONTROL FORMAT (R)

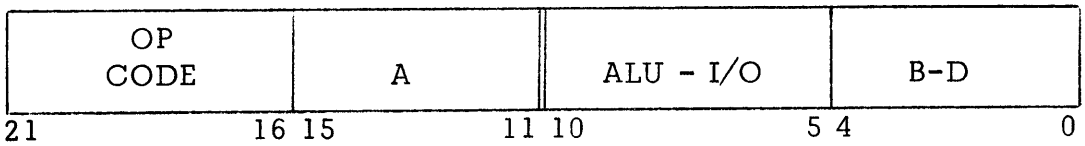


FORMAT 2 - LITERAL FORMAT (L)



FORMAT 3 - BRANCH FORMAT (B)

REGISTER CONTROL FORMAT (R) - FORMAT 1



OP CODE FIELD - Bits 16-21

MNEMONIC	HEX CODE	FUNCTION
RN	3F	Null - No Operation
RS1	34	Status Operation 1
RS2	35	Status Operation 2
RS3	36	Status Operation 3
RS4	37	Status Operation 4
RIF	38	Return to Instruction Fetch (RIF)
RFS1	30	RIF and Status Operation 1
RFS2	31	RIF and Status Operation 2
RFS3	32	RIF and Status Operation 3
RFS4	33	RIF and Status Operation 4
RCC	3B	CoCall
RMR	3A	Microreturn
RD1	3E	Decode I
RD2	3D	Decode II
RD3	3C	Decode III
RLR	39	Return from Microsubroutine Loop

2. GENERAL NOTES

1. Status operations are defined as follows:

Status Operation 1	TC → C TM → M TZ → Z
Status Operation 2	TM → M TZ → Z
Status Operation 3	TC → C TM → M TZ · Z → Z
Status Operation 4	TM → M TZ · Z → Z

2. The following definitions clarify the above mnemonics:

Null	No operation
CoCall	RAR + 1 → RAR, RRAS → RAR SC → SCS, SCS → SC
Decode I	DE → IR IMA → RAR, RAR + 1 → RRAS, RRAS↓ SC → SCS, SCS↓
Decode II, III	IMA → RAR, RAR + 1 → RRAS, RRAS↓ SC → SCS, SCS↓
Microreturn	RRAS → RAR, RRAS ↑ SCS → SC, SCS↑
Branch and Mark	BA → RAR, RAR + 1 → RRAS, RRAS↓ μI <19:16> → SC, SC → SCS, SCS↓
Return from Microsubroutine Loop	If SC=0, then RRAS → RAR, RRAS ↑ SCS → SC, SCS↑ If SC≠0, then RAR + 1 → RAR SC + 1 → SC

3. The Branch instruction may specify the Branch and Load Step Counter operation which is defined as follows:

If M = 0 (Don't Mark)

BA → RAR;

μI < 19:16 > → step counter

If M = 1 (Do Mark)

BA → RAR, RAR + 1 → RRAS, RRAS ↓

μI < 19:16 > → step counter, SC → SCS, SCS ↓

4. Return From Microsubroutine Loop is always executed conditionally upon the step counter equalling zero.

5. An unmapped macro OpCode field will cause the RAR to branch to μI ROM location 000_{16}. 000_{16} should always contain a Branch (Format 3) or the machine will halt and remain at 000_{16} until cleared. The RAR cannot increment from 000_{16} to 001_{16}.

6. Unconditional branchs are implemented by selecting 2s complement -1, Code lD, as the A Field source. Any other data with a logic 1 may be selected.

7. The step counter is a mod 16 counter based on the polynomial $X^4 + X + 1$.

3. A, B-D FIELDS

A - Bits 11-15
B/D - Bits 0-4

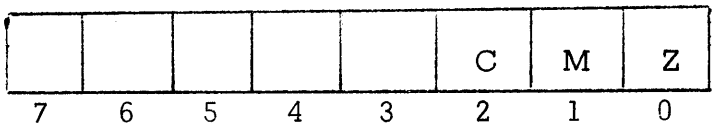
MNEMONIC	HEX CODE	FUNCTION
R0	0 0	General Register 0
R1	0 1	" " 1
R2	0 2	" " 2
R3	0 3	" " 3
R4	0 4	" " 4
R5	0 5	" " 5
R6	0 6	" " 6
R7	0 7	" " 7
R8	0 8	" " 8
R9	0 9	" " 9
R10	0 A	" " 10
R11	0 B	" " 11
R12	0 C	" " 12
R13	0 D	" " 13
R14	0 E	" " 14
R15	0 F	General Register 15
SK1	1 0	Stack 1
SK1I	1 I	Stack 1, Increment Pointer
SK1D	1 2	Stack 1, Decrement Pointer
SP1	1 3	Stack Pointer 1
SK2	1 4	Stack 2
SK2I	1 5	Stack 2, Increment Pointer
SK2D	1 6	Stack 2, Decrement Pointer
SP2	1 7	Stack Pointer 2

MNEMONIC	HEX CODE	FUNCTION
FS	1 9	Final Status
TS	1 A	Temporary Status (A Source Only)
PALU	1 B	Previous ALU Result (Null Destination)
DEL	1 C	Data Exchange Low
M1	1 D	2s Complement 1 (Null Destination)
LSP	1 E	Load Stack Pointers (Format 1 Only)
Z	1 F	Zero (Null Destination)

Notes:

1. The Stack Codes 10 and 14 cause selection of stack bytes without modifying the Pointers. Stack Codes 11 and 15 select a byte from the Stack and then increment the pointer after loading the A or B Source Registers. Stack Codes 12 and 16 select a byte from the stack and then decrement the pointer after loading the A or B Source Register.

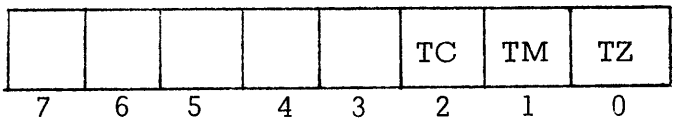
2. Final Status is assigned as follows:



where:

Bit	
0	Z - Zero Flag
1	M - Minus Flag
2	C - Carry Flag

3. Temporary Status is assigned as follows:



where:

Bit	
0	TZ - Temporary Zero
1	TM - Temporary Minus
2	TC - Temporary Carry

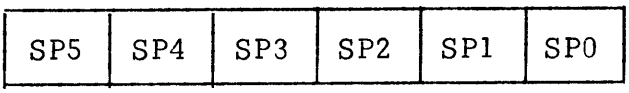
4. Previous ALU result is derived from the ALU Result Register (RR).

5. Data Exchange Low, DEL, causes the completion of a previously initiated memory or peripheral transfer. The machine will automatically wait until the data is available before completing the execution of the microinstruction.

6. The following tables depict in detail all of the possible Stack operations. N represents the Stack Pointer position at the beginning of microinstruction execution. The Final Positions of the Stack Pointers are shown to indicate where the pointers are addressing for use with the next microinstruction.

7. The Stack Pointer Counters bit assignments are as follows:

6 Bit Binary Counter



SP5	SP4	
0	0	Selects Stack Low Group
0	1	Selects Stack High Group
1	0	Selects General Registers
1	1	Null - Selects all 0s to A Source and B Source

SP3-SP0

Selects 1 of 16 registers of the selected register group as specified by SP5 and SP4.

8. Temporary Status is a Null Destination. In the A, B-D Fields all unused codes generate 2s complement -1 as a source:

4. A + 0 → D STACK OPERATION

SOURCES A	SOURCES B-D	STACK FUNCTION	FINAL POSITIONS SP1	FINAL POSITIONS SP2
SK1	SK1	$SK1_N \rightarrow SK1_N$	N	N
SK1	SK1I	$SK1_N \rightarrow SK1_N$	N + 1	N
SK1	SK1D	$SK1_N \rightarrow SK1_N$	N - 1	N
SK1	SK2	$SK1_N \rightarrow SK2_N$	N	N
SK1	SK2I	$SK1_N \rightarrow SK2_N$	N	N + 1
SK1	SK2D	$SK1_N \rightarrow SK2_N$	N	N - 1
SK1I	SK1	$SK1_N \rightarrow SK1_{N+1}$	N + 1	N
SK1I	SK1I	$SK1_N \rightarrow SK1_{N+1}$	N + 2	N
SK1I	SK1D	$SK1_N \rightarrow SK1_{N+1}$	N	N
SK1I	SK2	$SK1_N \rightarrow SK2_N$	N + 1	N
SK1I	SK2I	$SK1_N \rightarrow SK2_N$	N + 1	N + 1
SK1I	SK2D	$SK1_N \rightarrow SK2_N$	N + 1	N - 1
SK1D	SK1	$SK1_N \rightarrow SK1_{N-1}$	N - 1	N
SK1D	SK1I	$SK1_N \rightarrow SK1_{N-1}$	N	N
SK1D	SK1D	$SK1_N \rightarrow SK1_{N-1}$	N - 2	N
SK1D	SK2	$SK1_N \rightarrow SK2_N$	N - 1	N
SK1D	SK2I	$SK1_N \rightarrow SK2_N$	N - 1	N + 1
SK1D	SK2D	$SK1_N \rightarrow SK2_N$	N - 1	N - 1
SK2	SK1	$SK2_N \rightarrow SK1_N$	N	N
SK2	SK1I	$SK2_N \rightarrow SK1_N$	N + 1	N
SK2	SK1D	$SK2_N \rightarrow SK1_N$	N - 1	N
SK2	SK2	$SK2_N \rightarrow SK2_N$	N	N
SK2	SK2I	$SK2_N \rightarrow SK2_N$	N	N + 1
SK2	SK2D	$SK2_N \rightarrow SK2_N$	N	N - 1
SK2I	SK1	$SK2_N \rightarrow SK1_N$	N	N + 1
SK2I	SK1I	$SK2_N \rightarrow SK1_N$	N + 1	N + 1
SK2I	SK1D	$SK2_N \rightarrow SK1_N$	N - 1	N + 1

SOURCES A	SOURCES B-D	STACK FUNCTION	FINAL POSITIONS SP1	FINAL POSITIONS SP2
SK2I	SK2	$SK2_N \rightarrow SK2_{N+1}$	N	N + 1
SK2I	SK2I	$SK2_N \rightarrow SK2_{N+1}$	N	N + 2
SK2I	SK2D	$SK2_N \rightarrow SK2_{N+1}$	N	N
SK2D	SK1	$SK2_N \rightarrow SK1_N$	N	N - 1
SK2D	SK1I	$SK2_N \rightarrow SK1_N$	N + 1	N - 1
SK2D	SK1D	$SK2_N \rightarrow SK1_N$	N - 1	N - 1
SK2D	SK2	$SK2_N \rightarrow SK2_{N-1}$	N	N - 1
SK2D	SK2I	$SK2_N \rightarrow SK2_{N-1}$	N	N
SK2D	SK2D	$SK2_N \rightarrow SK2_{N-1}$	N	N - 2

5. B + 0 → STACK OPERATIONS

SOURCES A	SOURCES B-D	STACK FUNCTION	FINAL POSITIONS SP1	FINAL POSITIONS SP2
SK1	SK1	$SK1_N \rightarrow SK1_N$	N	N
SK1	SK1I	$SK1_N \rightarrow SK1_N$	N + 1	N
SK1	SK1D	$SK1_N \rightarrow SK1_N$	N - 1	N
SK1	SK2	$SK2_N \rightarrow SK2_N$	N	N
SK1	SK2I	$SK2_N \rightarrow SK2_N$	N	N + 1
SK1	SK2D	$SK2_N \rightarrow SK2_N$	N	N - 1
SK1I	SK1	$SK1_{N+1} \rightarrow SK1_{N+1}$	N + 1	N
SK1I	SK1I	$SK1_{N+1} \rightarrow SK1_{N+1}$	N + 2	N
SK1I	SK1D	$SK1_{N+1} \rightarrow SK1_{N+1}$	N	N
SK1I	SK2	$SK2_N \rightarrow SK2_N$	N + 1	N
SK1I	SK2I	$SK2_N \rightarrow SK2_N$	N + 1	N + 1
SK1I	SK2D	$SK2_N \rightarrow SK2_N$	N + 1	N - 1
SK1D	SK1	$SK1_{N-1} \rightarrow SK1_{N-1}$	N - 1	N
SK1D	SK1I	$SK1_{N-1} \rightarrow SK1_{N-1}$	N	N
SK1D	SK1D	$SK1_{N-1} \rightarrow SK1_{N-1}$	N - 2	N
SK1D	SK2	$SK2_N \rightarrow SK2_N$	N - 1	N
SK1D	SK2I	$SK2_N \rightarrow SK2_N$	N - 1	N + 1
SK1D	SK2D	$SK2_N \rightarrow SK2_N$	N - 1	N - 1
SK2	SK1	$SK1_N \rightarrow SK1_N$	N	N
SK2	SK1I	$SK1_N \rightarrow SK1_N$	N + 1	N
SK2	SK1D	$SK1_N \rightarrow SK1_N$	N - 1	N
SK2	SK2	$SK2_N \rightarrow SK2_N$	N	N
SK2	SK2I	$SK2_N \rightarrow SK2_N$	N	N + 1
SK2	SK2D	$SK2_N \rightarrow SK2_N$	N	N - 1
SK2I	SK1	$SK1_N \rightarrow SK1_N$	N	N + 1
SK2I	SK1I	$SK1_N \rightarrow SK1_N$	N + 1	N + 1
SK2I	SK1D	$SK1_N \rightarrow SK1_N$	N - 1	N + 1

SOURCES			FINAL POSITIONS	
A	B-D	STACK FUNCTION	SP1	SP2
SK2I	SK2	$SK2_{N+1} \rightarrow SK2_{N+1}$	N	N + 1
SK2I	SK2I	$SK2_{N+1} \rightarrow SK2_{N+1}$	N	N + 2
SK2I	SK2D	$SK2_{N+1} \rightarrow SK2_{N+1}$	N	N
SK2D	SK1	$SK1_{N} \rightarrow SK1_{N}$	N	N - 1
SK2D	SK1I	$SK1_{N} \rightarrow SK1_{N}$	N + 1	N - 1
SK2D	SK1D	$SK1_{N} \rightarrow SK1_{N}$	N - 1	N - 1
SK2D	SK2	$SK2_{N-1} \rightarrow SK2_{N-1}$	N	N - 1
SK2D	SK2I	$SK2_{N-1} \rightarrow SK2_{N-1}$	N	N
SK2D	SK2D	$SK2_{N-1} \rightarrow SK2_{N-1}$	N	N - 2

6. A + B → D STACK OPERATIONS

SOURCES A	SOURCES B-D	STACK FUNCTION	FINAL POSITIONS SP1	FINAL POSITIONS SP2
SK1	SK1	$SK1_N + SK1_N \rightarrow SK1_N$	N	N
SK1	SK1I	$SK1_N + SK1_N \rightarrow SK1_N$	N + 1	N
SK1	SK1D	$SK1_N + SK1_N \rightarrow SK1_N$	N - 1	N
SK1	SK2	$SK1_N + SK2_N \rightarrow SK2_N$	N	N
SK1	SK2I	$SK1_N + SK2_N \rightarrow SK2_N$	N	N + 1
SK1	SK2D	$SK1_N + SK2_N \rightarrow SK2_N$	N	N - 1
SK1I	SK1	$SK1_N + SK1_{N+1} \rightarrow SK1_{N+1}$	N + 1	N
SK1I	SK1I	$SK1_N + SK1_{N+1} \rightarrow SK1_{N+1}$	N + 2	N
SK1I	SK1D	$SK1_N + SK1_{N+1} \rightarrow SK1_{N+1}$	N	N
SK1I	SK2	$SK1_N + SK2_N \rightarrow SK2_N$	N + 1	N
SK1I	SK2I	$SK1_N + SK2_N \rightarrow SK2_N$	N + 1	N + 1
SK1I	SK2D	$SK1_N + SK2_N \rightarrow SK2_N$	N + 1	N - 1
SK1D	SK1	$SK1_N + SK1_{N-1} \rightarrow SK1_{N-1}$	N - 1	N
SK1D	SK1I	$SK1_N + SK1_{N-1} \rightarrow SK1_{N-1}$	N	N
SK1D	SK1D	$SK1_N + SK1_{N-1} \rightarrow SK1_{N-1}$	N - 2	N
SK1D	SK2	$SK1_N + SK2_N \rightarrow SK2_N$	N - 1	N
SK1D	SK2I	$SK1_N + SK2_N \rightarrow SK2_N$	N - 1	N + 1
SK1D	SK2D	$SK1_N + SK2_N \rightarrow SK2_N$	N - 1	N - 1
SK2	SK1	$SK2_N + SK1_N \rightarrow SK1_N$	N	N
SK2	SK1I	$SK2_N + SK1_N \rightarrow SK1_N$	N + 1	N
SK2	SK1D	$SK2_N + SK1_N \rightarrow SK1_N$	N - 1	N
SK2	SK2	$SK2_N + SK2_N \rightarrow SK2_N$	N	N
SK2	SK2I	$SK2_N + SK2_N \rightarrow SK2_N$	N	N + 1
SK2	SK2D	$SK2_N + SK2_N \rightarrow SK2_N$	N	N - 1
SK2I	SK1	$SK2_N + SK1_N \rightarrow SK1_N$	N	N + 1
SK2I	SK1I	$SK2_N + SK1_N \rightarrow SK1_N$	N + 1	N + 1
SK2I	SK1D	$SK2_N + SK1_N \rightarrow SK1_N$	N - 1	N + 1

SOURCES			FINAL POSITIONS	
A	B-D	STACK FUNCTION	SP1	SP2
SK2I	SK2	$SK2_N + SK2_{N+1} \rightarrow SK2_{N+1}$	N	N + 1
SK2I	SK2I	$SK2_N + SK2_{N+1} \rightarrow SK2_{N+1}$	N	N + 2
SK2I	SK2D	$SK2_N + SK2_{N+1} \rightarrow SK2_{N+1}$	N	N
SK2D	SK1	$SK2_N + SKI_N \rightarrow SK1_N$	N	N - 1
SK2D	SK1I	$SK2_N + SK1_N \rightarrow SK1_N$	N + 1	N - 1
SK2D	SK1D	$SK2_N + SK1_N \rightarrow SK1_N$	N - 1	N - 1
SK2D	SK2	$SK2_N + SK2_{N-1} \rightarrow SK2_{N-1}$	N	N - 1
SK2D	SK2I	$SK2_N + SK2_{N-1} \rightarrow SK2_{N-1}$	N	N
SK2D	SK2D	$SK2_N + SK2_{N-1} \rightarrow SK2_{N-1}$	N	N - 2

7. ALU - I/O FIELD - BITS 5-10

MNEMONIC	HEX CODE	FUNCTION
TAD	0 0	A + B 8 bit Add Co = 0
ADI	0 1	A + B 8 bit Add Co = 1
ADTC	0 2	A + B 8 bit Add Co = TC
ADC	0 3	A + B 8 bit Add Co = C
A	0 8	A + 0 8 bit Add Co = 0
AI	0 9	A + 0 8 bit Add Co = 1
ATC	0 A	A + 0 8 bit Add Co = TC
AC	0 B	A + 0 8 bit Add Co = C
NA	3 4	$\overline{A}$ + 0 8 bit Add Co = 0
NAI	3 5	$\overline{A}$ + 0 8 bit Add Co = 1
NATC	3 6	$\overline{A}$ + 0 8 bit Add Co = TC
NAC	3 7	$\overline{A}$ + 0 8 bit Add Co = C
XOR	1 0	A + B Exclusive OR
OR	1 7	A $\wedge$ B OR Logical
AND	1 8	A $\vee$ B AND Logical
SRZ	2 4	Shift A Right with 0
SRTC	2 6	Shift A Right with TC
SRC	2 7	Shift A Right with C
SRSE	2 5	Shift A Right with Sign Extention
SD1	2 8	Swap Digits A <3:0>, B <7:4> → D <7:0>
SD2	2 F	Swap Digits B<3:0>, A <7:4> →D <7:0>
DCZ	3 0	2421 Decimal Correction with Co = 0
MPI	3 8	Initiate Input and B + 0 with Co = 0
MPII	3 9	Initiate Input and B + 0 with Co = 1
MPIT	3 A	Initiate Input and B + 0 with Co = TC
MPIC	3 B	Initiate Input and B + 0 with Co = C
MPO	3 C	Initiate Output and B + 0 with Co = 0
MPOI	3 D	Initiate Output and B + 0 with Co = 1
MPOT	3 E	Initiate Output and B + 0 with Co = TC
MPOC	3 F	Initiate Output and B + 0 with Co = C

Notes:

1. Temporary carry bits are not modified by logical operations.

2. The decimal correction operations do not modify the TC.

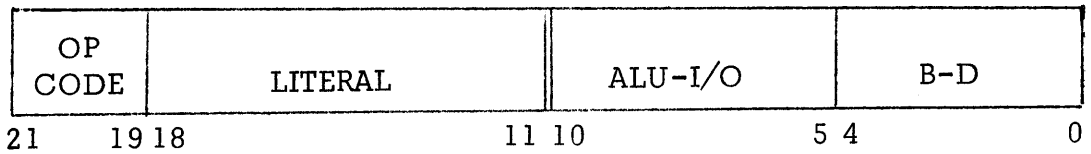
3. When doing MPO or MPI operations the Address is derived as follows:

   (DEH) Address High = A source

   (DEL) Address Low = B-D source

4. The decimal correction operation will correct the contents of the specified A Source.

8. LITERAL FORMAT (L) - FORMAT 2



The Literal Format utilizes the same field assignment for the B-D and ALU-I/O fields as described above. In doing I/O the following definitions apply for Literals:

MPI Initiate Memory/Peripheral Input and B <7:0> → D < 7:0 >

LITERAL → DEH

B < 7:0 > → DEL

MPO Initiate Memory/Peripheral Out and B < 7:0 > → D < 7:0 >

LITERAL → DEH

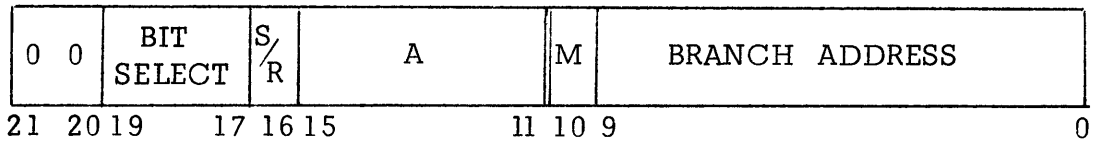
B < 7:0 > → DEL

The memory/peripheral out operations may be initiated with the MPO instruction and terminated by the AK I/O line. This allows the implementation of I/O output transfers with single microinstructions. The Literal may be any 8-bit value. The Literal (Bits 11-18) must be specified in complement form.

9. OPCODE FIELD - BITS 19-21

MNEMONIC	HEX CODE	FUNCTION
LN	5	Null - No Operation
LMR	3	Microreturn
LCC	2	CoCall
LLR	4	Return From Microsubroutine Loop

10. BRANCH FORMAT (B) - FORMAT 3



The Branch Instruction Fields have the following definitions:

Bit Select (Bits 17-19) Specifies one out of 8 bits in the selected register to Branch On. Only one bit may be selected at a time.

S/R (Bit 16) Specifies whether to branch on 1 or 0 condition.

S/R	
1	Branch On 1
0	Branch On 0

A (Bits 11-15) As specified above. The A Source Field may be any register in the machine as well as the constants 0 and -1. Branching on the -1 condition will provide the Unconditional Branch capability.

M (Bit 10) Mark during the Branch

BA (Bits 0-9) Branch Address, 10-bit polynomial coded count.

11. BRANCH CONDITIONAL

The above fields are combined to form various OpCodes as depicted below:

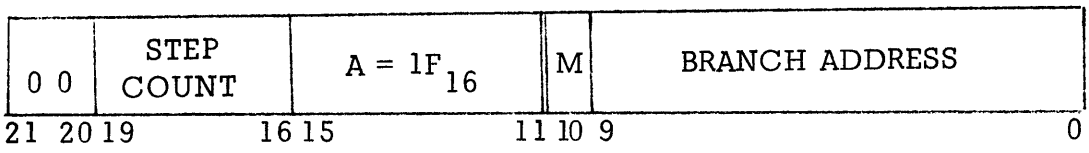
MNEMONIC	M	S/R	FUNCTION
BR	0	0	Branch On Reset Condition
BS	0	1	Branch On Set Condition
BMR	1	0	Branch And Mark On Reset Condition
BMS	1	1	Branch And Mark On Set Condition

These operations are further defined as follows. The Branch Control (BC) line represents the state of the selected bit.

MNEMONIC	BC = 0	BC = 1
BR	BA → RAR	RAR + 1 → RAR
BS	RAR + 1 → RAR	BA → RAR
BMR	BA → RAR, RAR + 1 → RRAS, RRAS↓ SC → SCS, SCS↓	RAR + 1 → RAR
BMS	RAR + 1 → RRAR	BA → RAR, RAR + 1 → RRAS, RRAS↓ SC → SCS, SCS↓

12. BRANCH ON AND LOAD STEP COUNTER

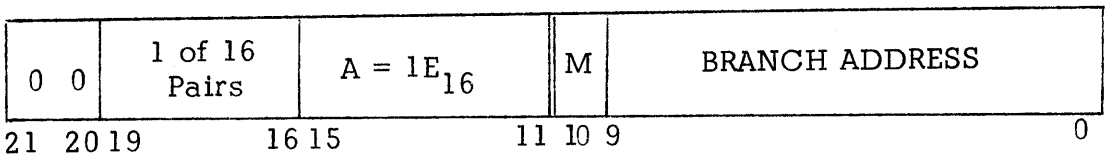
A special form of the Branch Instruction is used to implement the Step Counter operations. The instruction form is generated by utilizing the A Field zero source code (1F) as follows:



MNEMONIC	M	FUNCTION
BC	0	Branch and Load Step Counter
BMC	1	Branch and Mark and Load Step Counter

13. BRANCH AND LOAD STACK POINTERS

Another special form of the Branch Instruction is used to load 1 of 16 pairs of values into Stack Pointer 1 or 2. The instruction form is generated by utilizing the A Field Source Code 1E as follows:

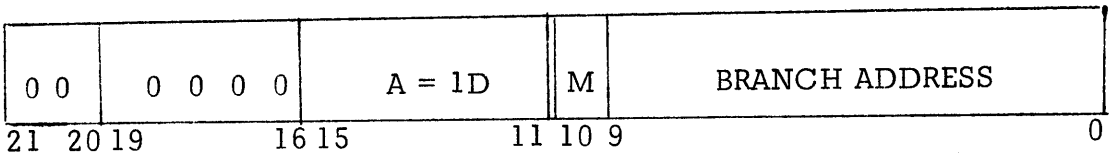


The 16 pairs of values are mask programmable on the RALU chip 1.

MNEMONIC	M	FUNCTION
BP	0	Branch and Load Stack Pointers
BMP	1	Branch and Mark and Load Stack Pointers

14. BRANCH UNCONDITIONAL

The Branch Unconditional is implemented with the following form



MNEMONIC	M	FUNCTION
BU	0	Branch Unconditional
BMU	1	Branch and Mark Unconditional

Note:

1. The Step Counter has seven levels of First-In Last-Out Stack (SCS). When the Branch and Mark (BMC) operation is given, the Step Counter (MC) is pushed into the stack. When the CoCall or Microreturn is given, the top of the stack is unconditionally popped into the Step Counter. When the Return From Microsubroutine Loop is given, and the Step Counter equals zero, the top of the Step Counter Stack (SCS) is popped into the Step Counter.

15. MNEMONIC MICROASSEMBLER FORMATS

FORMAT 1 - REGISTER CONTROL (R)

LABEL	OPCODE	A,ALU-I/O, B-D
EXAMPLE:		
LOCIO	RFSI	RO, TAD, SKID

FORMAT 2 - LITERAL (L)

LABEL	OPCODE	LITERAL, ALU-I/O, B-D
EXAMPLE:		
LOCIO	LCC	8F, A, SP2

FORMAT 3 - BRANCH (B)

LABEL	OPCODE	BIT SELECT, A, BA
EXAMPLE:		
LOCIO	BMR	5, R7, LOC12

If Step Counter OpCode OpCodes are used:

LABEL	OPCODE	STEP COUNT, BA
EXAMPLE:		
LOCIO	BMC	7, LOC12