Appendix A

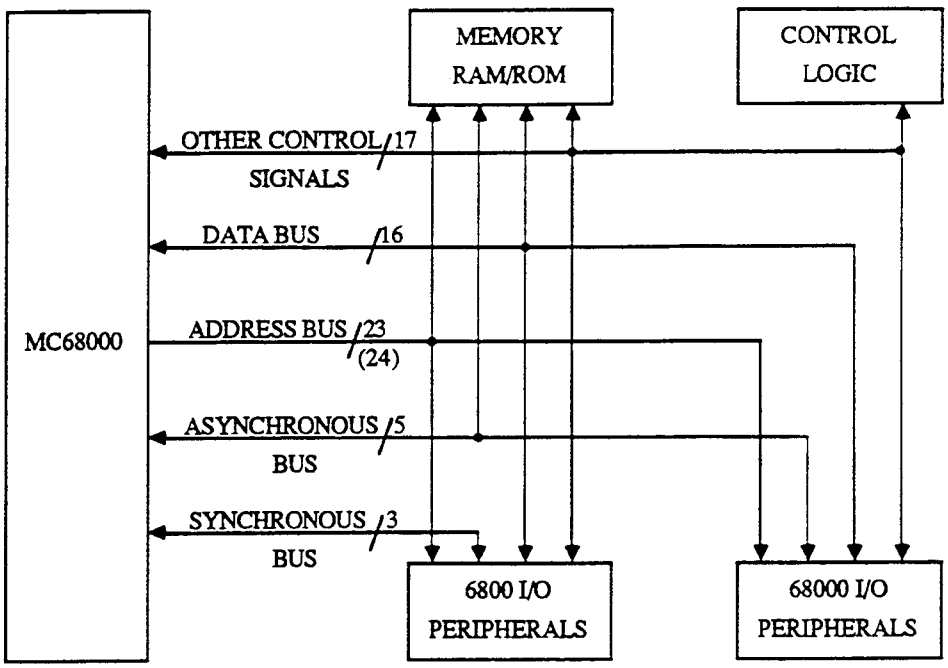
MC 68000 Course Notes

by

Motorola Inc .

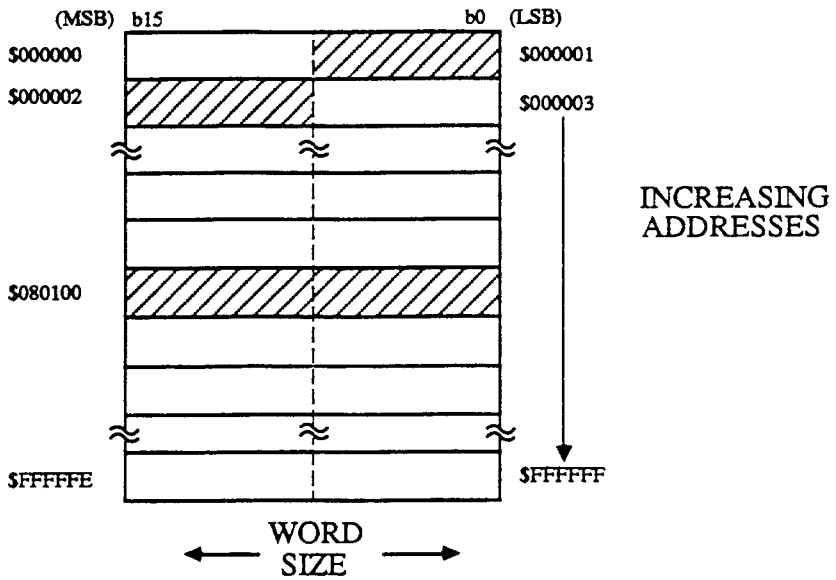
Semiconductor Products Sector

MC68000 – BASIC SYSTEM DIAGRAM



MT8-002-1

MC68000 – MEMORY DIAGRAMS



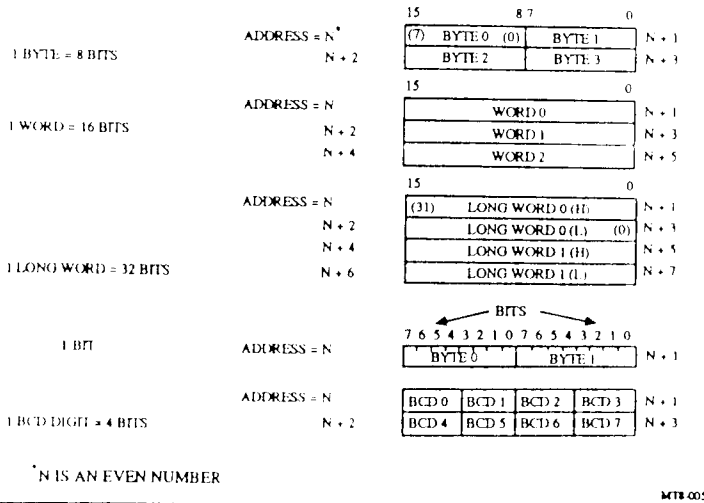
MT8-003-1

MC68000 – RULES FOR ACCESSING MEMORY

- Words and long words must be accessed from an even address.
- Bytes can be accessed from either an odd or even address.
- Op words must be accessed on even addresses.

MTB-004-1

MC68000 INTEGER DATA MEMORY FORMATS



*N IS AN EVEN NUMBER

MTB-005-1

MC68000 – CYCLE TIME DEFINITIONS

CLOCK CYCLE
The input clock period from positive edge to positive edge

BUS CYCLE
The sequence of timing events required to do a byte or a word read cycle, byte or a word write cycle, or a read-modify-write cycle

INSTRUCTION CYCLE
The sequence of timing events required to do an instruction

MTB-006-2

ASSERTION (ASSERT)

A signal (pin) is active or true independent of the actual voltage level.

NEGATION (NEGATE)

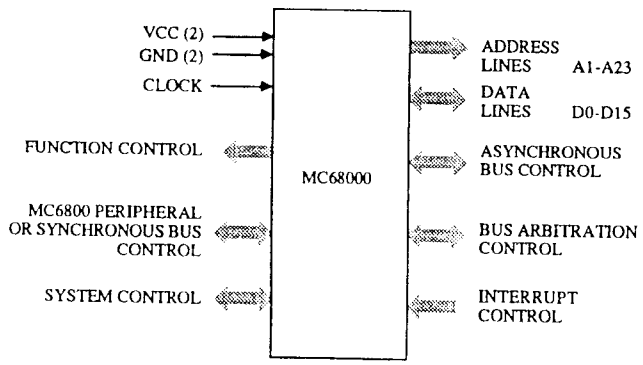
A signal (pin) is inactive or false independent of the actual voltage level.

RESCINDABLE (RESCIND)

A signal (pin) is first negated and then three-stated (high impedance).

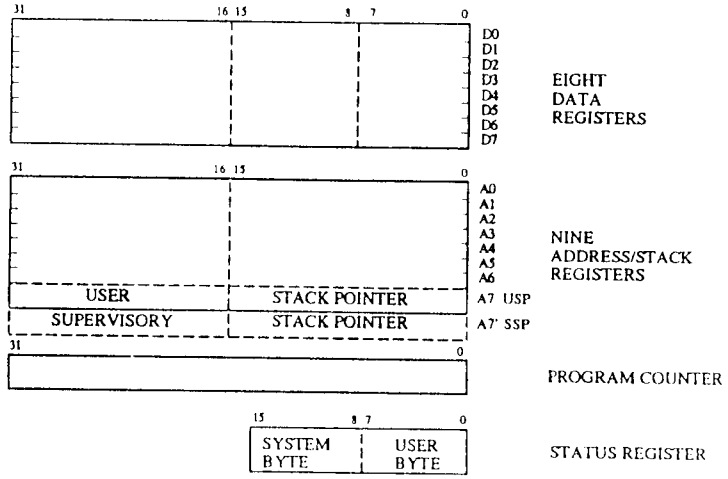
MTB-007-1

MC68000 – BASIC PIN DIAGRAM



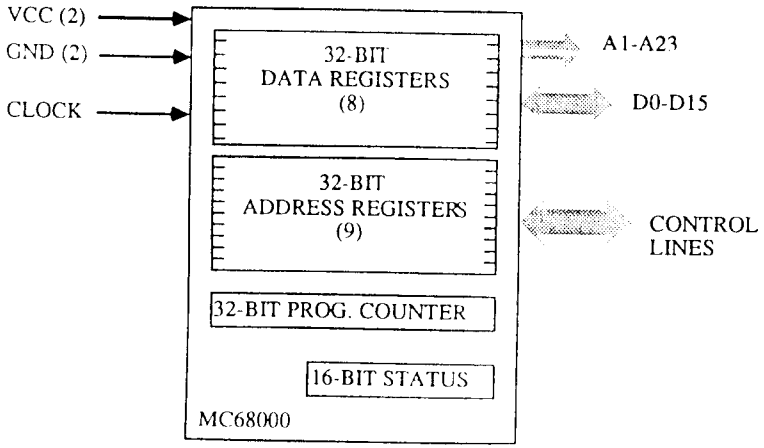
MT8-008-2

MC68000 PROGRAMMING MODEL



MT8-010-2

MC68000 – REGISTER DIAGRAM



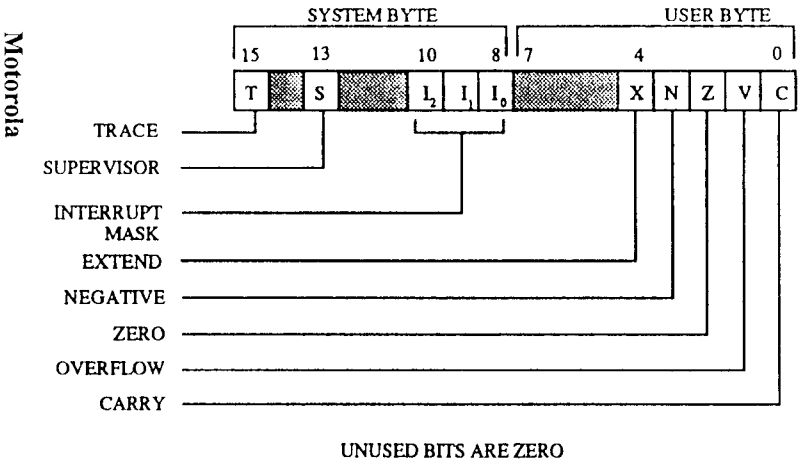
MT8-009-3

MC68000 – ADDRESS AND DATA REGISTER DIFFERENCES

	DATA REGISTER	ADDRESS REGISTER
CCR		
byte operands		
word operands		
longword operands		

MT8-511

MC68000 STATUS REGISTER



UNUSED BITS ARE ZERO

MTB-011-2

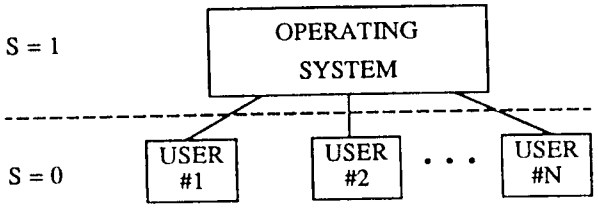
SUPERVISOR (S) BIT

TWO STATES (LEVELS) OF PRIVILEGE:

- S = 1 : SUPERVISOR (HIGHEST)
- S = 0 : USER (LOWEST)

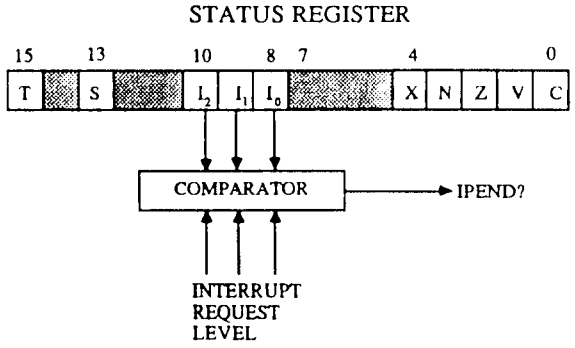
THE PRIVILEGE STATE:

- PROVIDES SECURITY
- DETERMINES WHICH OPERATIONS ARE LEGAL



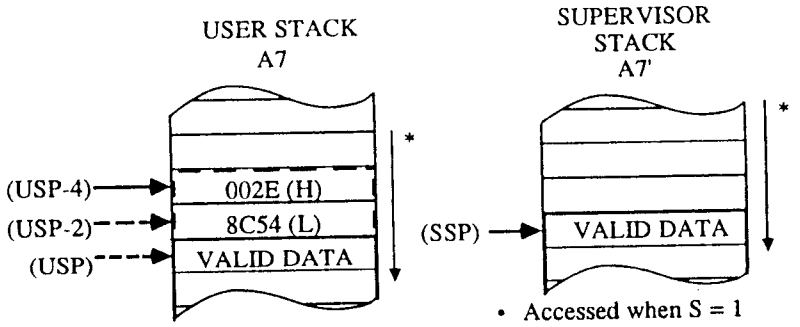
MTB-200-2

MC68000–INTERRUPT MASK



MTB-206.1-1

MC68000 – STACK POINTERS



- Accessed when S = 0
- PC is stacked on subroutine calls in user state, eg;

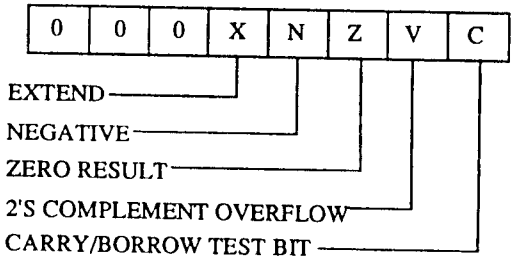
  PC = 002E8C54

* Increasing addresses

- Accessed when S = 1
- PC is stacked on subroutine calls in supervisor state
- Used for exception processing

MTB-012-1

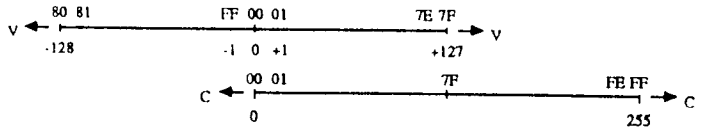
MC68000 – CONDITION CODE REGISTER

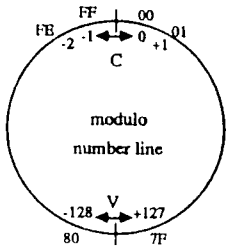


BITS INDICATE THE RESULTS OF AN INSTRUCTION

MTB-013-1

SIGNED AND UNSIGNED NUMBERS





Which condition code bits provide useful information for evaluating arithmetic results?

DATA TYPE	X	N	Z	V	C
SIGNED					
UNSIGNED					

MTB-509-2

THE CARRY FUNCTION

CARRY: 2 MAJOR USES
1) PROGRAM CONTROL : C
2) LINKING FIELDS : X

ARITHMETIC INSTRUCTIONS AFFECT BOTH X AND C, X = C

TEST INSTRUCTIONS LEAVE X UNAFFECTED BUT UPDATE C

MOST INSTRUCTIONS DON'T AFFECT X

TASK: ADD $99AABB to $EEDDCC

TEST:	ENDADDR	ARITH:	X	X	?
	– POINTER		EE	DD	CC
			+ 99	+ AA	+ BB

1. POINT TO NUMBERS
2. ADD NUMBERS (ARITH)
3. COMPARE POINTER TO END ADDR (TEST)
4. IF C BIT SET, GO TO 1
5. ELSE DONE

MTB-201-1

CONDITION CODE REGISTER EXAMPLE

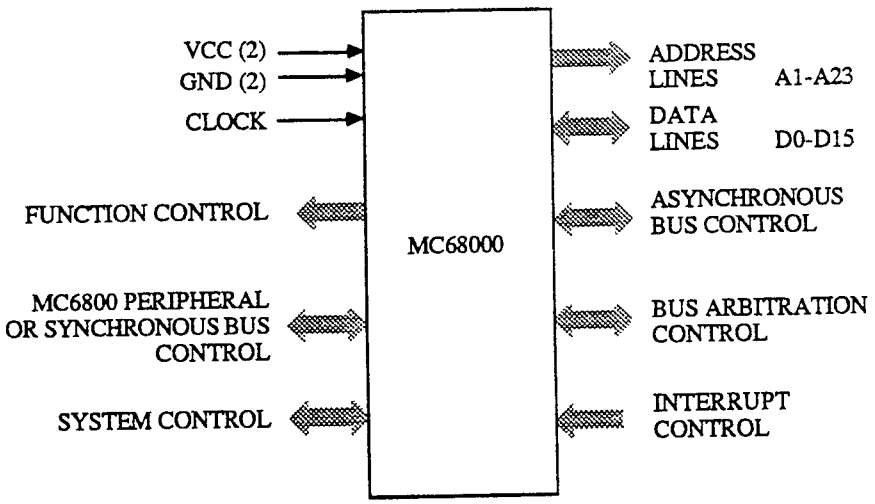
DATA REGISTER D0 = 00123456

DATA REGISTER D1 = FFFE70AA

OPERATION:	RESULT:	CCR RESULT:
ADD BYTE OF D0 REG. TO D1 REG.	D1 = BYTE SUM	X N Z V C: 0 0 0
ADD WORD OF D0 REG. TO D1 REG.	D1 = WORD SUM	X N Z V C: 0 0 0
ADD LONGWORD OF D0 REG. TO D1 REG.	D1 = LONG WORD SUM	X N Z V C: 0 0 0

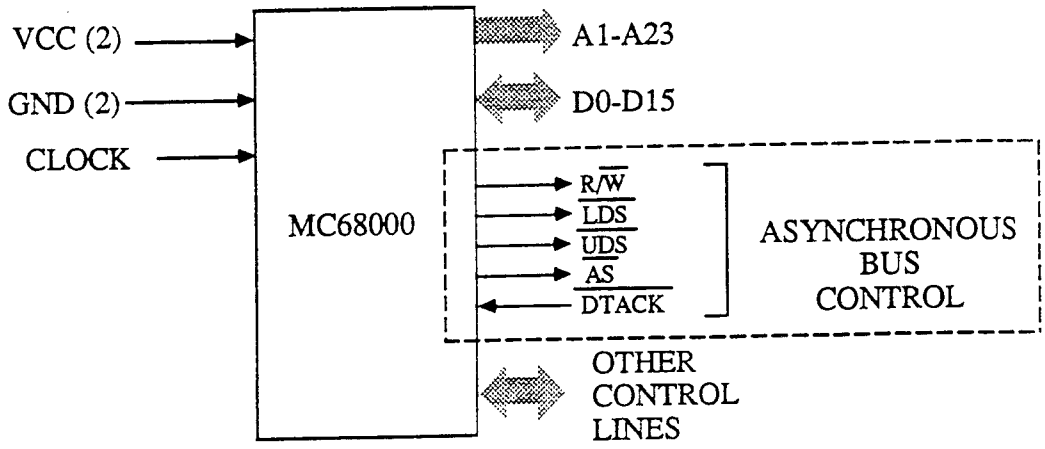
014

MC68000 – BASIC PIN DIAGRAM



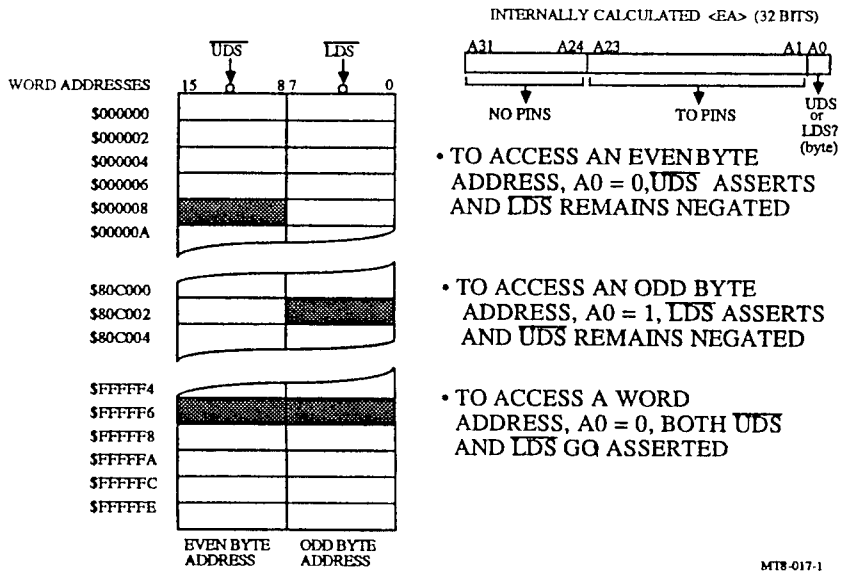
MT8-008-2

MC68000 – ASYNCHRONOUS BUS CONTROL DIAGRAM



MT8-015-1

MC68000 - MEMORY MAP



- TO ACCESS AN EVEN BYTE ADDRESS, A0 = 0, $\overline{UDS}$ ASSERTS AND $\overline{LDS}$ REMAINS NEGATED
- TO ACCESS AN ODD BYTE ADDRESS, A0 = 1, $\overline{LDS}$ ASSERTS AND $\overline{UDS}$ REMAINS NEGATED
- TO ACCESS A WORD ADDRESS, A0 = 0, BOTH $\overline{UDS}$ AND $\overline{LDS}$ GO ASSERTED

MT8-017-1

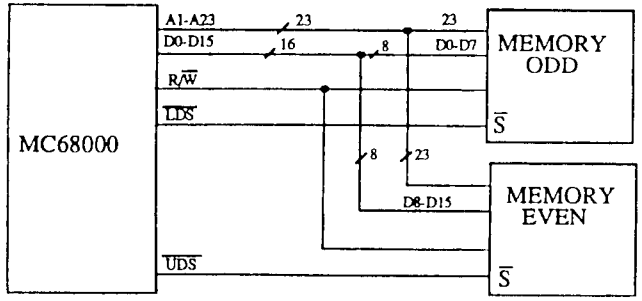
MC68000 — BYTE CONTROL

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8-D15	D0-D7
LOW	LOW	LOW	VALID WRITE DATA	VALID WRITE DATA
LOW	LOW	HIGH	VALID READ DATA	VALID READ DATA
LOW	HIGH	LOW	VALID WRITE DATA	*SAME AS D8-D15
LOW	HIGH	HIGH	VALID READ DATA	INVALID DATA
HIGH	LOW	LOW	*SAME AS D0-D7	VALID WRITE DATA
HIGH	LOW	HIGH	INVALID DATA	VALID READ DATA
HIGH	HIGH	LOW	INVALID DATA	INVALID DATA
HIGH	HIGH	HIGH	INVALID DATA	INVALID DATA

* This feature is not part of the 68000 specification, and is not assured in future versions.

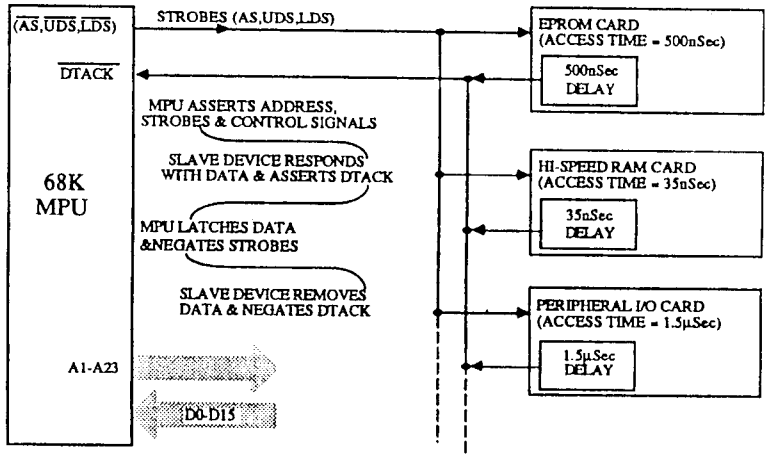
MT8-019-2

MC68000 - BYTE ADDRESSING



MT8-018

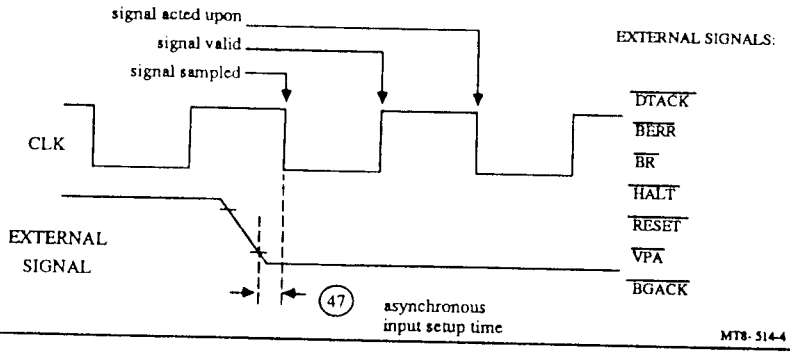
ASYNCHRONOUS TRANSFER OF DATA (READ)



MT8-016-1

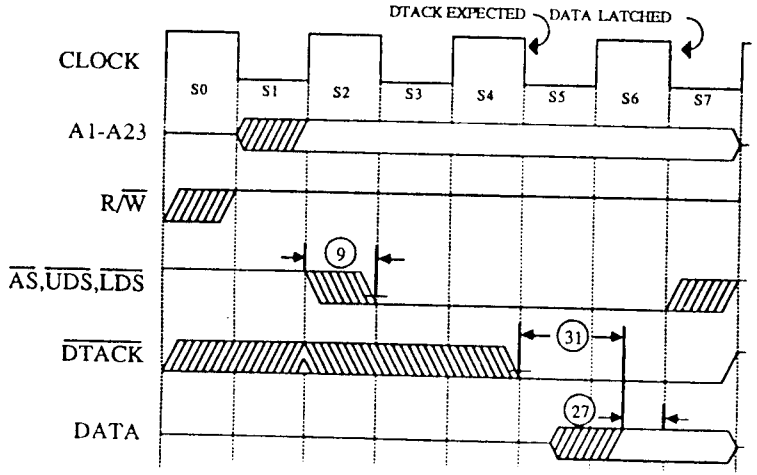
MC68000 – EXTERNAL ASYNCHRONOUS INPUTS

- All external asynchronous input signals must be synchronized to the CPU clock before being acted upon.
- This causes a clock cycle of delay before an external signal is used internally.
- If the external signal does not meet the asynchronous input setup time it may not be sampled as asserted until the next falling edge of the CPU clock.



MT8-514-4

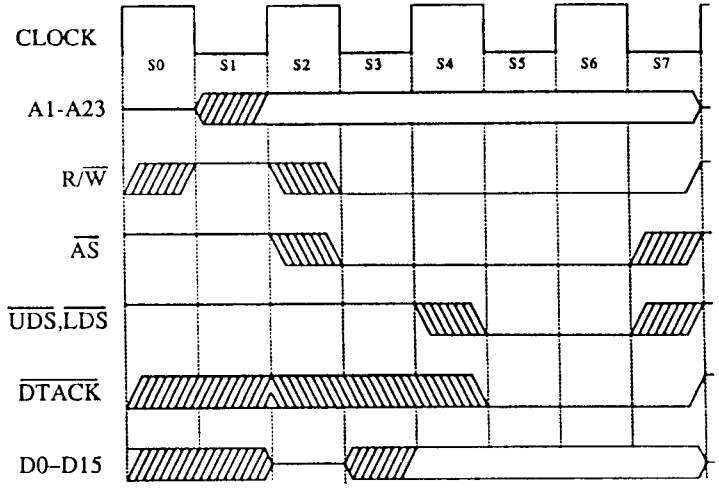
MC68000 – READ BUS CYCLE OPERATION (WORD)



What is the memory access time if the clock frequency is 10 MHz?_____

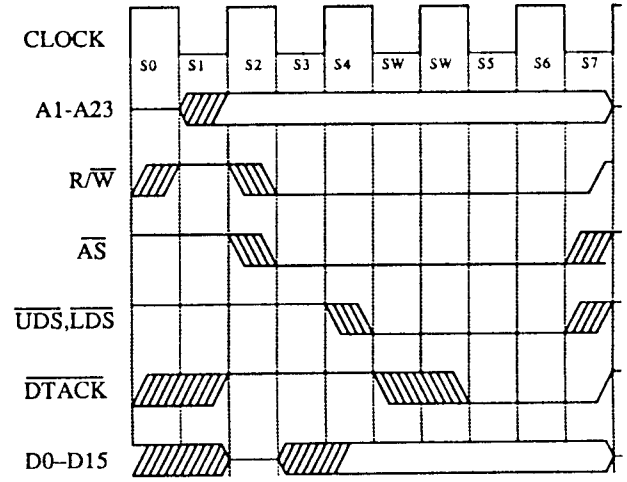
MT8-020-3

MC68000 – WRITE BUS CYCLE OPERATION (WORD)
(4 CLOCK CYCLES)



MT8-021-1

MC68000 – WRITE BUS CYCLE OPERATION (WORD)
(5 CLOCK CYCLES)



MT8-022-1

8.6 AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES

(V_{CC} = 5.0 Vdc ±5%; V_{SS} = 0 Vdc; $T_A = T_L$ to T_H; see Figures 8-6 and 8-7

Num.	Characteristic	Symbol	4 MHz		6 MHz		8 MHz		10 MHz		12.5 MHz		Unit
			Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
1	Clock Period	t_{CYC}	250	500	167	500	125	500	100	500	80	250	ns
2	Clock Width Low	t_{CL}	115	250	75	250	55	250	45	250	35	125	ns
3	Clock Width High	t_{CH}	115	250	75	250	55	250	45	250	35	125	ns
4	Clock Fall Time	t_{Cf}	–	10	–	10	–	10	–	10	–	5	ns
5	Clock Rise Time	t_{Cr}	–	10	–	10	–	10	–	10	–	5	ns
6	Clock Low to Address	t_{CLAV}	–	90	–	80	–	70	–	60	–	55	ns
6A	Clock High to FC Valid	t_{CHFCV}	–	90	–	80	–	70	–	60	–	55	ns
7	Clock High to Address Data High Impedance (Maximum)	t_{CHAZx}	–	120	–	100	–	80	–	70	–	60	ns
8	Clock High to Address/FC Invalid (Minimum)	t_{CHAZn}	0	–	0	–	0	–	0	–	0	–	ns
9[1]	Clock High to $\overline{AS}$, $\overline{DS}$ Low (Maximum)	t_{CHSLx}	–	80	–	70	–	60	–	55	–	55	ns
10	Clock High to $\overline{AS}$, $\overline{DS}$ Low (Minimum)	t_{CHSLn}	0	–	0	–	0	–	0	–	0	–	ns
11[2]	Address to $\overline{AS}$, $\overline{DS}$ (Read) Low/$\overline{AS}$ Write	t_{AVSL}	55	–	35	–	30	–	20	–	0	–	ns
11A[2,7]	FC Valid to $\overline{AS}$, $\overline{DS}$ (Read) Low/$\overline{AS}$ Write	t_{FCVSL}	80	–	70	–	60	–	50	–	40	–	ns
12[1]	Clock Low to $\overline{AS}$, $\overline{DS}$ High	t_{CLSH}	–	90	–	80	–	70	–	55	–	50	ns
13[2]	$\overline{AS}$, $\overline{DS}$ High to Address/FC Invalid	t_{SHAZ}	60	–	40	–	30	–	20	–	10	–	ns
14[2,5]	$\overline{AS}$, $\overline{DS}$ Width Low (Read)/$\overline{AS}$ Write	t_{SL}	535	–	337	–	240	–	195	–	160	–	ns
14A[2]	$\overline{DS}$ Width Low (Write)	t_{DWPW}	285	–	170	–	115	–	95	–	80	–	ns
15[2]	$\overline{AS}$, $\overline{DS}$ Width High	t_{SH}	285	–	180	–	150	–	105	–	65	–	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$ High Impedance	t_{CHSZ}	–	120	–	100	–	80	–	70	–	60	ns
17[2]	$\overline{AS}$, $\overline{DS}$ High to R/$\overline{W}$ High	t_{SHRH}	60	–	50	–	40	–	20	–	10	–	ns
18[1]	Clock High to R/$\overline{W}$ High (Maximum)	t_{CHRHx}	–	90	–	80	–	70	–	60	–	60	ns
19	Clock High to R/$\overline{W}$ High (Minimum)	t_{CHRHn}	0	–	0	–	0	–	0	–	0	–	ns
20[1]	Clock High to R/$\overline{W}$ Low	t_{CHRL}	–	90	–	80	–	70	–	60	–	60	ns
20A[8]	$\overline{AS}$ Low to R/$\overline{W}$ Valid	t_{ASRV}	–	20	–	20	–	20	–	20	–	20	ns
21[2]	Address Valid to R/$\overline{W}$ Low	t_{AVRL}	45	–	25	–	20	–	0	–	0	–	ns
21A[2,7]	FC Valid to R/$\overline{W}$ Low	t_{FCVRL}	80	–	70	–	60	–	50	–	30	–	ns
22[2]	R/$\overline{W}$ Low to $\overline{DS}$ Low (Write)	t_{RLSL}	200	–	140	–	80	–	50	–	30	–	ns
23	Clock Low to Data Out Valid	t_{CLDO}	–	90	–	80	–	70	–	55	–	55	ns
24	Clock High to R/$\overline{W}$, $\overline{VMA}$ High Impedance	t_{CHRZ}	–	120	–	100	–	80	–	70	–	60	ns
25[2]	$\overline{DS}$ High to Data Out Invalid	t_{SHDO}	60	–	40	–	30	–	20	–	15	–	ns
26[2]	Data Out Valid to $\overline{DS}$ Low (Write)	t_{DOSL}	55	–	35	–	30	–	20	–	15	–	ns
27[6]	Data In to Clock Low (Setup Time)	t_{DICL}	30	–	25	–	15	–	10	–	10	–	ns
28[2,5]	$\overline{AS}$, $\overline{DS}$ High to $\overline{DTACK}$ High	t_{SHDAH}	0	490	0	325	0	245	0	190	0	150	ns
29	$\overline{DS}$ High to Data Invalid (Hold Time)	t_{SHDI}	0	–	0	–	0	–	0	–	0	–	ns
30	$\overline{AS}$, $\overline{DS}$ High to $\overline{BERR}$ High	t_{SHBEH}	0	–	0	–	0	–	0	–	0	–	ns
31[2,6]	$\overline{DTACK}$ Low to Data In (Setup Time)	t_{DALDI}	–	180	–	120	–	90	–	65	–	50	ns

8.6 AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES (CONTINUED)

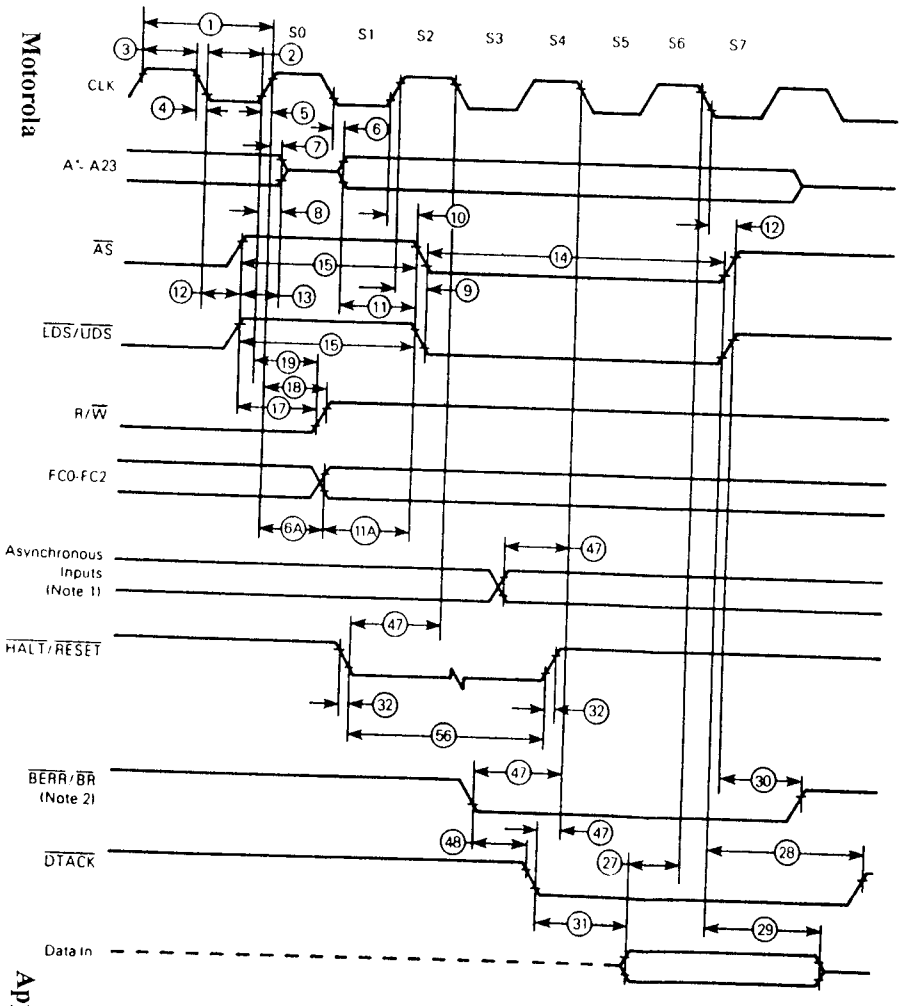
Num.	Characteristic	Symbol	4 MHz		6 MHz		8 MHz		10 MHz		12.5 MHz		Unit
			Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
32	$\overline{HALT}$ and $\overline{RESET}$ Input Transition Time	$t_{RHr, f}$	0	200	0	200	0	200	0	200	0	200	ns
33	Clock High to $\overline{BG}$ Low	t_{CHGL}	–	90	–	80	–	70	–	60	–	50	ns
34	Clock High to $\overline{BG}$ High	t_{CHGH}	–	90	–	80	–	70	–	60	–	50	ns
35	$\overline{BR}$ Low to $\overline{BG}$ Low	t_{BRLGL}	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clk. Per.
36	$\overline{BR}$ High to $\overline{BG}$ High	t_{BRHGH}	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clk. Per.
37	$\overline{BGACK}$ Low to $\overline{BG}$ High	t_{GALGH}	1.5	3.0	1.5	3.0	1.5	3.0	1.5	3.0	1.5	3.0	Clk. Per.
37A	$\overline{BGACK}$ Low to $\overline{BR}$ High (to Prevent Rearbitration)	t_{BGKBR}	30	–	25	–	20	–	20	–	20	–	ns
38	$\overline{BG}$ Low to Bus High Impedance (with $\overline{AS}$ High)	t_{GLZ}	–	120	–	100	–	80	–	70	–	60	ns
39	$\overline{BG}$ Width High	t_{GH}	1.5	–	1.5	–	1.5	–	1.5	–	1.5	–	Clk. Per.
40	Clock Low to $\overline{VMA}$ Low	t_{CLVML}	–	90	–	80	–	70	–	70	–	70	ns
41	Clock Low to E Transition	t_{CLC}	–	100	–	85	–	70	–	55	–	45	ns
42	E Output Rise and Fall Time	$t_{Er, f}$	–	25	–	25	–	25	–	25	–	25	ns
43	$\overline{VMA}$ Low to E High	t_{VMLEH}	325	–	240	–	200	–	150	–	90	–	ns
44	$\overline{AS}$, $\overline{DS}$ High to $\overline{VPA}$ High	t_{SHVPH}	0	240	0	160	0	120	0	90	0	70	ns
45	E Low to Address/$\overline{VMA}$/FC Invalid	t_{ELAI}	55	–	35	–	30	–	10	–	10	–	ns
46	$\overline{BGACK}$ Width	t_{BGL}	1.5	–	1.5	–	1.5	–	1.5	–	1.5	–	Clk. Per.
47[6]	Asynchronous Input Setup Time	t_{ASI}	30	–	25	–	20	–	20	–	20	–	ns
48[3]	$\overline{BERR}$ Low to $\overline{DTACK}$ Low	t_{BELDAL}	30	–	25	–	20	–	20	–	20	–	ns
49	E Low to $\overline{AS}$, $\overline{DS}$ Invalid	t_{ELSI}	−80	–	−80	–	−80	–	−80	–	−80	–	ns
50	E Width High	t_{EH}	900	–	600	–	450	–	350	–	280	–	ns
51	E Width Low	t_{EL}	1400	–	900	–	700	–	550	–	440	–	ns
52	E Extended Rise Time	t_{CIEHX}	–	80	–	80	–	80	–	80	–	80	ns
53	Data Hold from Clock High	t_{CHDO}	0	–	0	–	0	–	0	–	0	–	ns
54	Data Hold from E Low (Write)	t_{ELDOZ}	60	–	40	–	30	–	20	–	15	–	ns
55	R/$\overline{W}$ to Data Bus Impedance Change	t_{RLDO}	55	–	35	–	30	–	20	–	10	–	ns
56[4]	$\overline{HALT}$/$\overline{RESET}$ Pulse Width	t_{HRPW}	10	–	10	–	10	–	10	–	10	–	Clk. Per.

Notes:

1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the value given in these columns.
2. Actual value depends on clock period.
3. If #47 is satisifed for both $\overline{DTACK}$ and $\overline{BERR}$, #48 may be 0 nanoseconds.
4. For power up, the MPU must be held in $\overline{RESET}$ state for 100 ms to all stabilization of on-chip circuitry. After the system is powered up, #56 refers to the minimum pulse width required to reset the system.
5. #14, #14A, and #28 are one clock period less than the given number for T6E, BF4, and R9M mask sets.
6. If the asynchronous setup time (#47) requirements are satisfied, the $\overline{DTACK}$ low-to-data setup time (#31) requirement can be ignored. The data must only satisfy the data-in clock-low setup time (#27) for the following cycle.
7. For T6E, BF4, and R9M mask set 11A timing equals 11, and 21A equals 21. 20A may be 0 for T6E, BF4, and R9M mask sets.
8. When $\overline{AS}$ and R/$\overline{W}$ are equally loaded (±20%), subtract 10 nanoseconds from the values given in these columns.

Timing diagrams (Figures 8-6 and 8-7) are located on a fold-out page at the end of this document.

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

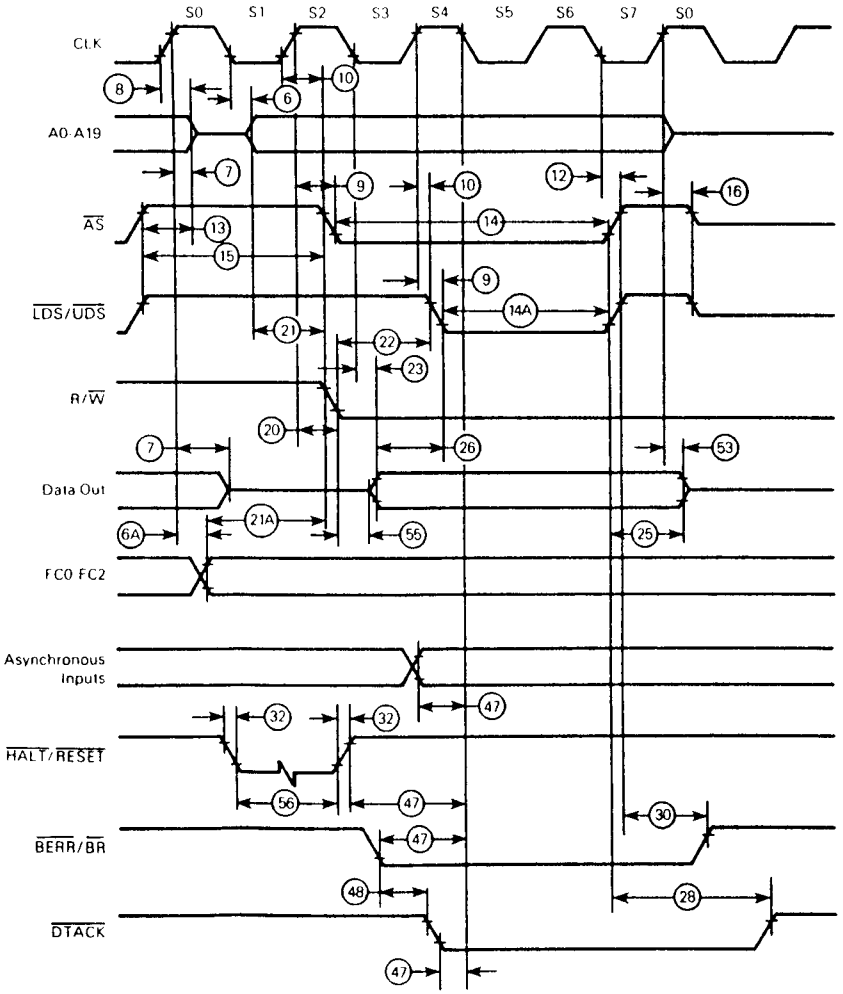


NOTES:
1. Setup time for the asynchronous inputs BGACK, IPL0-2, and VPA guarantees their recognition at the next falling edge of the clock.
2. BR need fall at this time only in order to insure being recognized at the end of this bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

Figure 8-6. Read Cycle Timing Diagram

Foldout 1

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

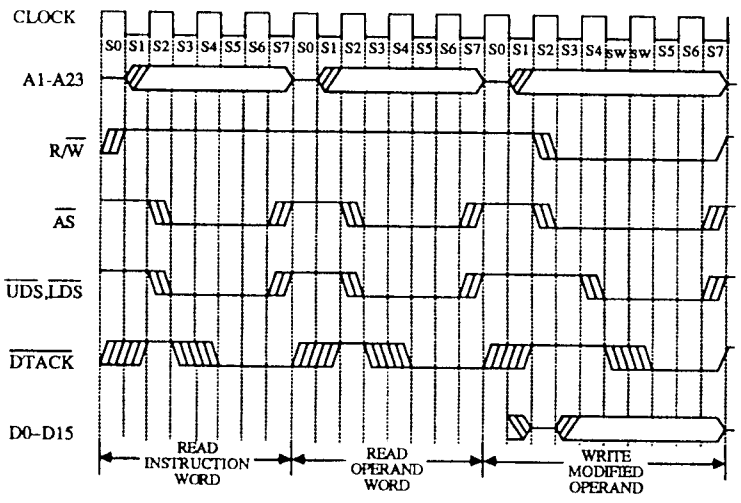


NOTES:
1. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.
2. Because of loading variations, R/W may be valid after AS even though both are initiated by the rising edge of S2 (Specification 20A).

Figure 8-7. Write Cycle Timing Diagram

Foldout 2

MC68000 – READ/MODIFY/WRITE INSTRUCTION (WORD)



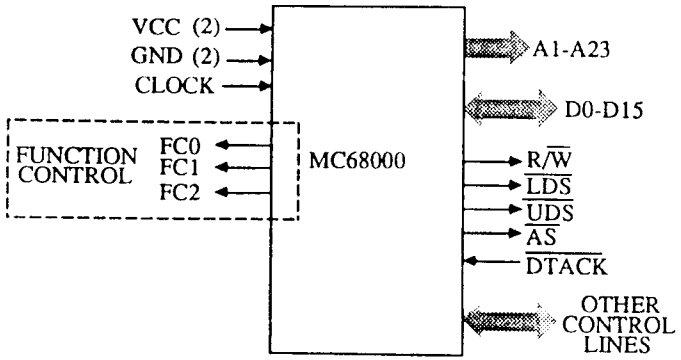
MT8-023-2

MC68000 – FUNCTION CONTROL STATES

FC2	FC1	FC0	STATE	MODE
0	0	0	RESERVED – MOTOROLA	USER
0	0	1	DATA SPACE	USER
0	1	0	PROGRAM SPACE	USER
0	1	1	RESERVED – USER	USER
1	0	0	RESERVED – MOTOROLA	SUPERVISOR
1	0	1	DATA SPACE	SUPERVISOR
1	1	0	PROGRAM SPACE	SUPERVISOR
1	1	1	INTERRUPT ACKNOWLEDGE	SUPERVISOR

MT8-025-1

MC68000 – FUNCTION CONTROL PIN DIAGRAM



MT8-024-1

MC68000 — PROGRAM vs. DATA

THE FUNCTION CONTROL OUTPUTS INDICATE PROGRAM WHEN:

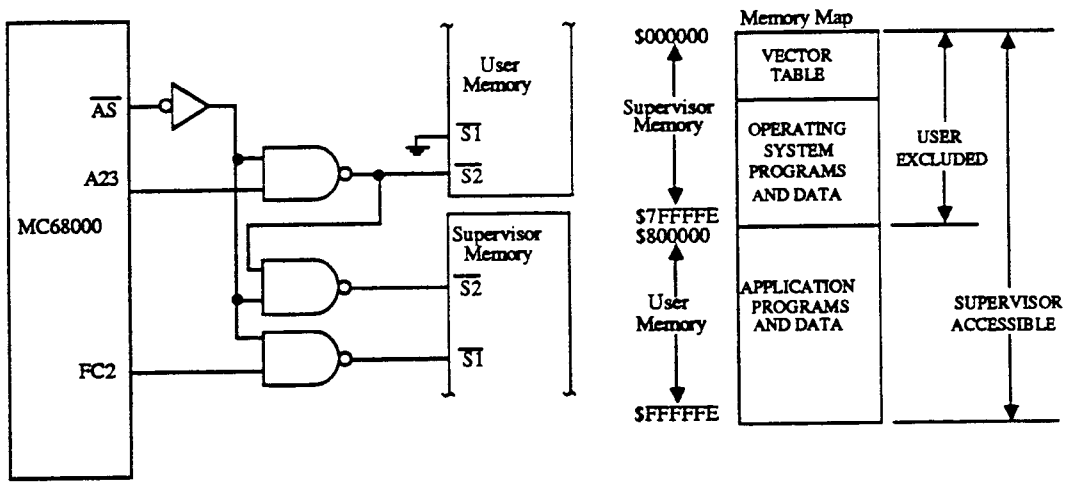
PC IS THE ADDRESS SOURCE
RESET VECTORS ARE FETCHED

THE FUNCTION CONTROL OUTPUTS INDICATE DATA WHEN:

MOST OPERANDS ARE READ (PC IS NOT ADDRESS SOURCE)
ALL OPERANDS ARE WRITTEN
VECTORS OTHER THAN RESET ARE FETCHED

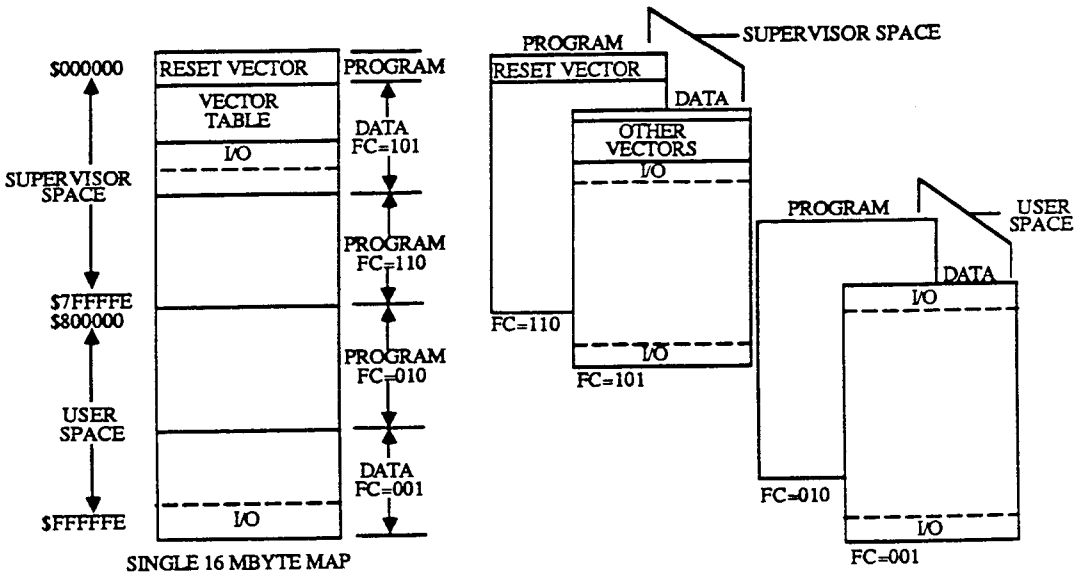
MT8-026

MC68000 – SIMPLE MEMORY PROTECTION (USER PROHIBITED FROM ACCESSING SUPERVISOR MEMORY)



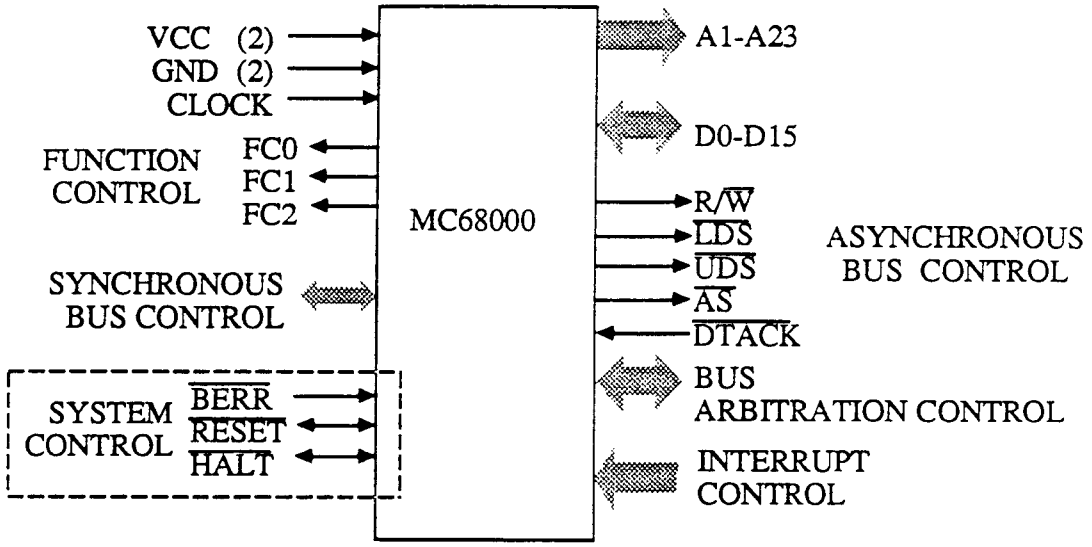
MT8-027-1

MC68000 – USING FUNCTION CONTROL OUTPUTS TO PARTITION AND MANAGE MEMORY



MT8-028-1

MC68000 – SYSTEM CONTROL PIN DIAGRAM



MT8-059-2

MC68000–BERR CHARACTERISTICS

- USED TO SIGNAL CPU THAT AN ERROR EXISTS DURING A BUS CYCLE
- WHEN $\overline{\text{BERR}}$ IS ASSERTED:
  - CPU ABORTS BUS CYCLE
  - DATA IS IGNORED
  - ADDRESS AND DATA 3-STATED
  - BUS CONTROL PINS ARE NEGATED
- AFTER $\overline{\text{BERR}}$ NEGATES, BUS ERROR EXCEPTION PROCESSING OCCURS (TYPICALLY).
- TYPICAL USES OF BUS ERROR:
  - BUS TIMEOUT
  - NON-CORRECTABLE (HARD) PARITY ERRORS
- USED WITH $\overline{\text{HALT}}$ FOR RERUN

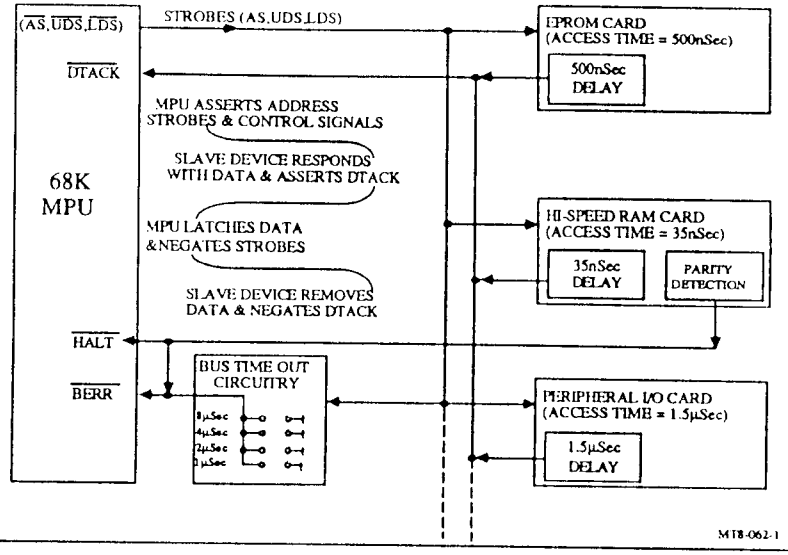
MT8-203-2

MC68000 – RESET CHARACTERISTICS

- EXTERNAL RESET INPUT
  1. POWER-ON
     $\overline{\text{RESET}}$ AND $\overline{\text{HALT}}$ ASSERTED FOR ≥ 100 MILLISEC
  2. PUSH-BUTTON
     $\overline{\text{RESET}}$ AND $\overline{\text{HALT}}$ ASSERTED FOR ≥ 10 CLOCKS
     AFTER $\overline{\text{RESET}}$ NEGATES ⌐, RESET EXCEPTION
     PROCESSING OCCURS.
- INTERNAL RESET OUTPUT (DUE TO RESET INSTRUCTION)
  $\overline{\text{RESET}}$ ASSERTS FOR 124 CLOCKS
  $\overline{\text{HALT}}$ ASSERTED DURING RESET CAUSES RESET
  EXCEPTION PROCESSING
  AFTER COMPLETION OF THE RESET INSTRUCTION,
  THE 68000 GOES TO THE NEXT INSTRUCTION;
  NO EXCEPTION PROCESSING OCCURS.

MTB-060-1

BERR — HALT DURING ASYNCHRONOUS DATA TRANSFER



MTB-062-1

MC68000 – HALT CHARACTERISTICS

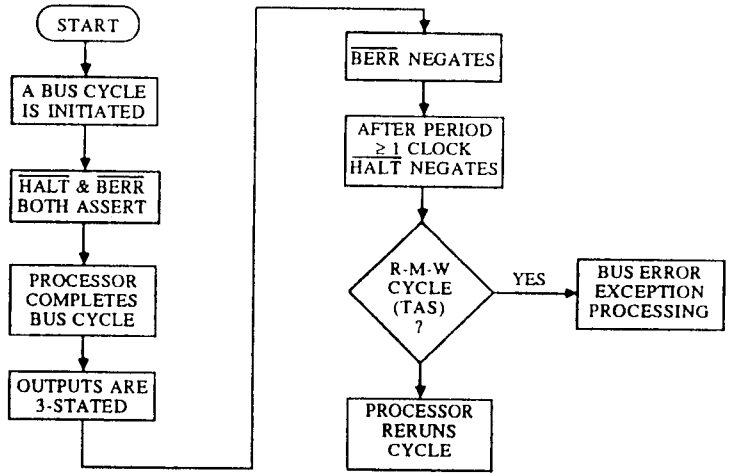
- WHEN $\overline{\text{HALT}}$ INPUT IS ASSERTED DURING EXECUTION OF AN INSTRUCTION:

  THE PROCESSOR STOPS EXECUTION UPON
  COMPLETION OF THE CURRENT BUS CYCLE
  ADDRESS AND DATA BUS ARE 3-STATED
  BUS CONTROL PINS ARE NEGATED; DMA CONTROL PINS
  ARE AVAILABLE

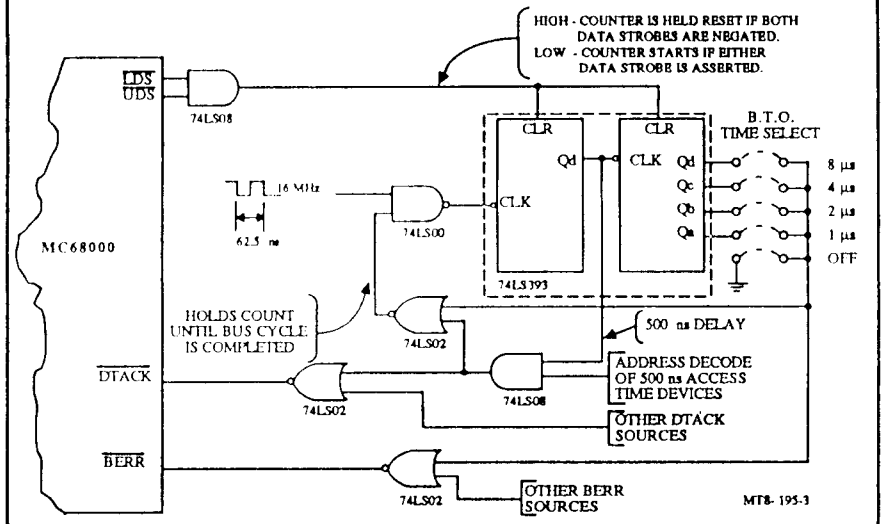
- WHEN $\overline{\text{HALT}}$ INPUT AND $\overline{\text{BERR}}$ ARE ASSERTED, A RERUN CYCLE CAN BE INITIATED (EXCEPT FOR TAS).
- WHEN $\overline{\text{HALT}}$ INPUT AND $\overline{\text{RESET}}$ ARE ASSERTED, A POWER-ON RESET, OR A PUSH BUTTON RESET HAS OCCURRED.
- WHEN $\overline{\text{HALT}}$ OUTPUT ASSERTS, A DOUBLE BUS FAULT HAS OCCURRED.

MTB-061-2

MC68000 – RERUN CYCLE



MC68000 – BUS TIME-OUT AND DTACK DELAY EXAMPLE

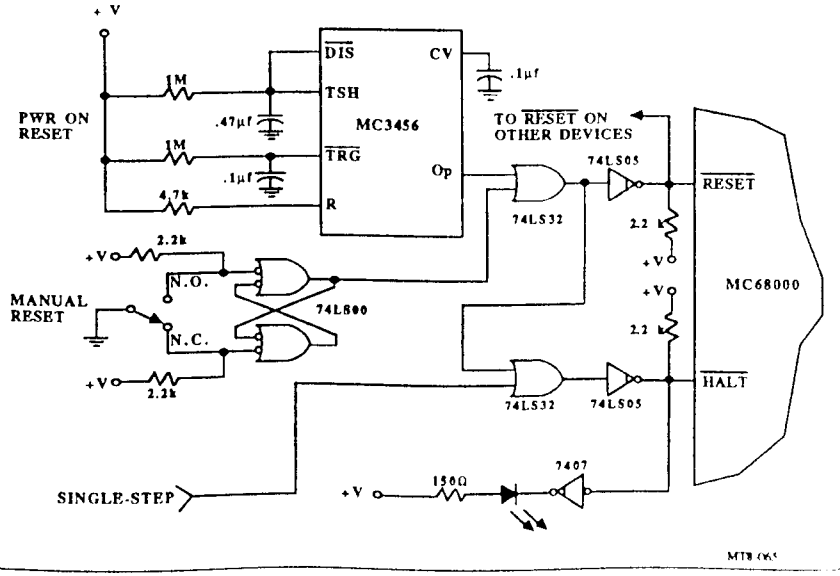


MC68000 – BUS ERROR AND HALT

BERR	HALT	RESULTING OPERATION
LOW	LOW	BUS ERROR - RERUN CURRENT CYCLE
LOW	HIGH	BUS ERROR - EXCEPTION PROCESSING
HIGH	LOW	SINGLE BUS CYCLE OPERATION
HIGH	HIGH	NORMAL OPERATION

MTB-064

MC68000 – RESET AND HALT



MC68000 – PROCESSING STATES

NORMAL	INSTRUCTION EXECUTION (INCLUDING STOP)
EXCEPTION	INTERRUPTS TRAPS TRACING
HALTED	HARDWARE HALT DOUBLE BUS ERROR DOUBLE ILLEGAL ADDRESS ERROR

MT8-073-1

MC68000—EXCEPTION DEFINITION

EXCEPTION – A DEVIATION FROM NORMAL PROCESSING SEQUENCE DUE TO AN INTERNAL INSTRUCTION OR ERROR CONDITION, OR AN EXTERNAL REQUEST OR ERROR CONDITION.

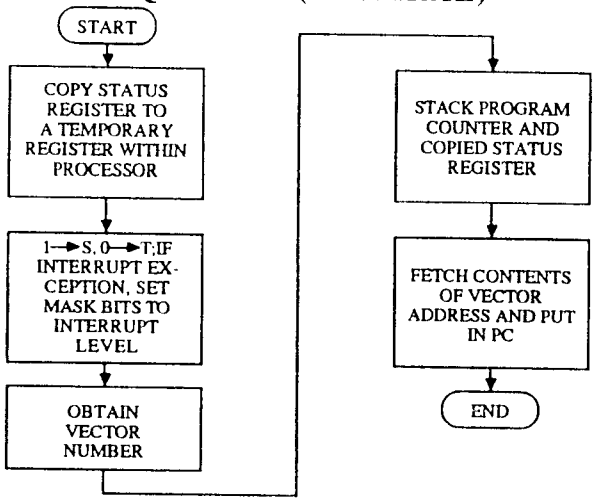
MT8-074-1

MC68000 – EXCEPTIONS

INTERNAL	INSTRUCTIONS (TRAP,TRAPV,CHK,DIV) ADDRESS ERRORS TRACE MODE
EXTERNAL	INTERRUPTS BUS ERROR RESET

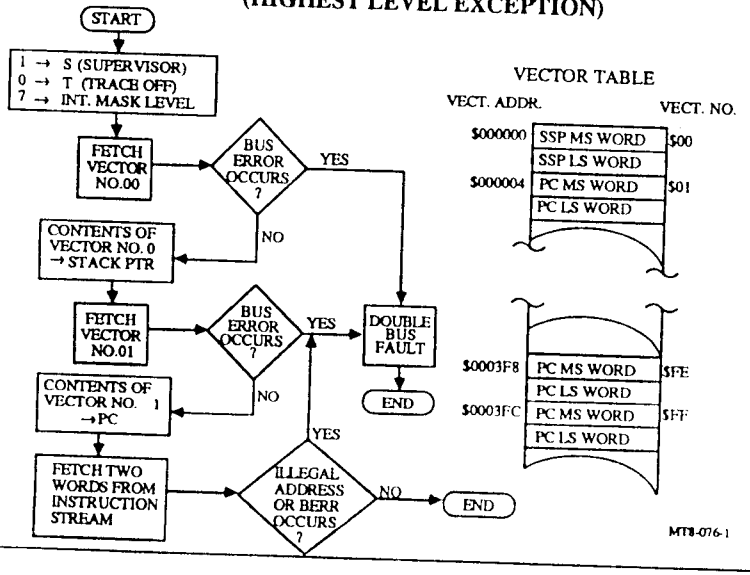
MT8-075-1

MC68000 – EXCEPTION PROCESSING SEQUENCE (GENERAL)



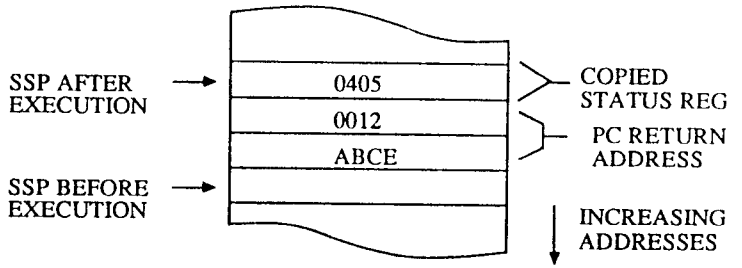
MT8-077

MC68000 – RESET EXCEPTION PROCESSING
(HIGHEST LEVEL EXCEPTION)



MT8-076-1

MC68000 – STACK OPERATION ON EXCEPTION
(OTHER THAN RESET, BUS ERROR & ILLEGAL ADDRESS)*



*NOTE: ADDITIONAL WORDS ARE STACKED FOR BUS ERROR AND ILLEGAL ADDRESS EXCEPTIONS

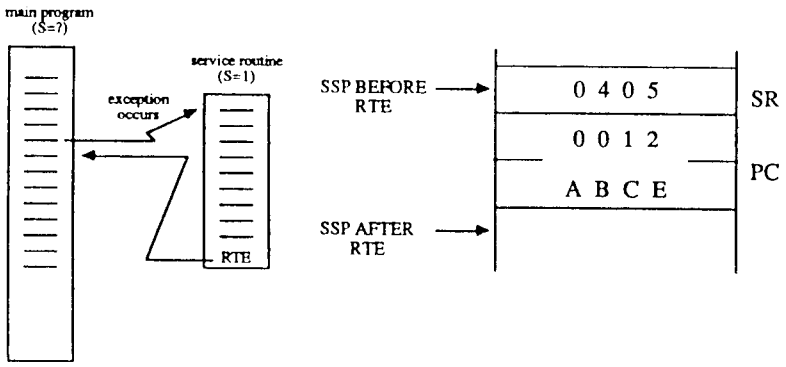
WHAT MODE WAS THE MPU OPERATING IN PRIOR TO THIS EXCEPTION? _____

MT8-078

RTE

Usually the last instruction in Exception Service Routine.

Pops values off supervisor stack INTO status register and PC



MT8-501-1

MC68000 – EXCEPTION VECTORS (1 of 2)

VECTOR ADDRESS HEX		VECTOR NO. HEX
0	RESET	0
4	RESET (CONT)	1
8	BUS ERROR	2
C	ILLEGAL ADDRESS	3
10	ILLEGAL INSTRUCTION	4
14	DIVIDE BY ZERO	5
18	CHK INSTRUCTION	6
1C	TRAPV INSTRUCTION	7
20	PRIVILEGE INSTRUCTION	8
24	TRACE	9
28	LINE 1010 EMULATOR	A
2C	LINE 1111 EMULATOR	B

MT8-079-1

MC68000 – EXCEPTION VECTORS (2 of 2)

VECTOR ADDRESS HEX		VECTOR NO. HEX
30 – 38	UNASSIGNED RESERVED	0C – 0E
3C	UNINITIALIZED INTERRUPT	0F
40 – 5C	UNASSIGNED RESERVED	10 – 17
60	SPURIOUS INTERRUPT	18
64	AUTO VECTOR #1	19
68	AUTO VECTOR #2	1A
6C	AUTO VECTOR #3	1B
70	AUTO VECTOR #4	1C
74	AUTO VECTOR #5	1D
78	AUTO VECTOR #6	1E
7C	AUTO VECTOR #7	1F
80 – BC	TRAP INSTRUCTIONS	20 – 2F
C0 – FC	UNASSIGNED RESERVED	30 – 3F
100 – 3FC	USER INTERRUPTS	40 – FF

MT8-080-2

MC68000 – ILLEGAL INSTRUCTION EXAMPLE

MOVE D0,#$1000

OP WORD

0011	100111	000	000
MOVE WORD	IMMEDIATE	DATA REGISTER DIRECT	REGISTER NUMBER "0"

NOTE: 68000 ASSEMBLER WILL FLAG THIS EXAMPLE INSTRUCTION AS AN ERROR

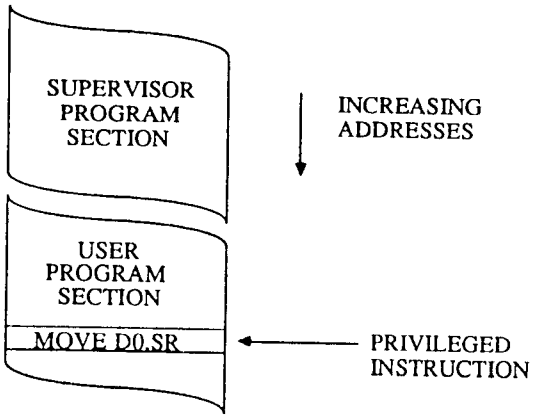
MT8-081

MC68000 – PRIVILEGE STATES

STATE	S BIT	OPERATIONAL RESTRICTIONS	
USER	0	PRIVILEGED INSTRUCTIONS WHICH ARE NOT ALLOWED:	
		INSTRUCTION	OPERATION
		RESET	RESET EXTERNAL DEVICES
		RTE	RETURN FROM EXCEPTION
		STOP	STOP PROGRAM EXECUTION
		ORI TO SR	LOGICAL OR TO STATUS REGISTER
		MOVE USP	MOVE USER STACK POINTER
		ANDI TO SR	LOGICAL AND TO STATUS REGISTER
		EORI TO SR	LOGICAL EOR TO STATUS REGISTER
		MOVE EA TO SR	LOAD NEW STATUS REGISTER
SUPERVISOR	1	NO RESTRICTIONS – ALL INSTRUCTIONS MAY BE EXECUTED	

MT8-084-1

MC68000 – PRIVILEGE VIOLATION EXAMPLE



MT8-082

TRACE MODE

IF T = 1

STATUS REGISTER

T

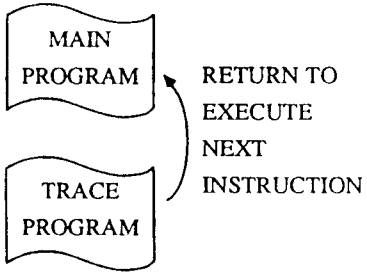
AFTER EACH INSTRUCTION

RETURN EXAMPLE:

SSP → 8000 → SR

0000 – 2000 → PC

ADDRESS OBTAINED FROM VECTOR TABLE



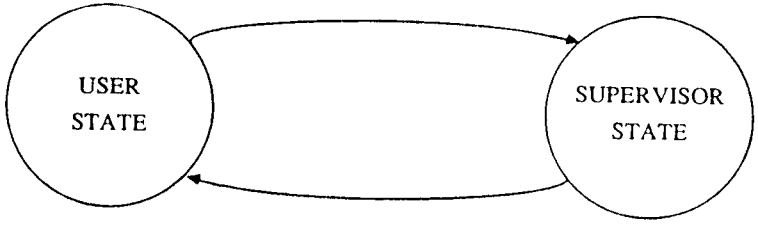
RETURN TO EXECUTE NEXT INSTRUCTION

1. IF, UPON COMPLETION OF AN INSTRUCTION, T = 1, GO TO TRACE EXCEPTION PROCESSING.
2. EXECUTE TRACE EXCEPTION SEQUENCE.
3. EXECUTE TRACE SERVICE ROUTINE.
4. AT THE END OF THE SERVICE ROUTINE, EXECUTE RETURN FROM EXCEPTION (RTE).

MT8-085

MC68000 – USER/SUPERVISOR MODES

TRANSITION ONLY MAY OCCUR DURING EXCEPTION PROCESSING

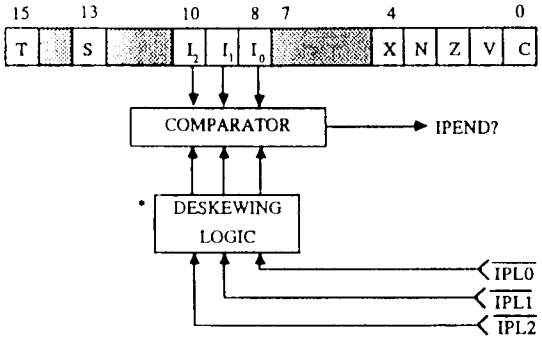


TRANSITION MAY BE MADE BY:
RTE; MOVE, ANDI, EORI TO STATUS WORD

MT8-083

INTERRUPT-PENDING LOGIC (IPEND)

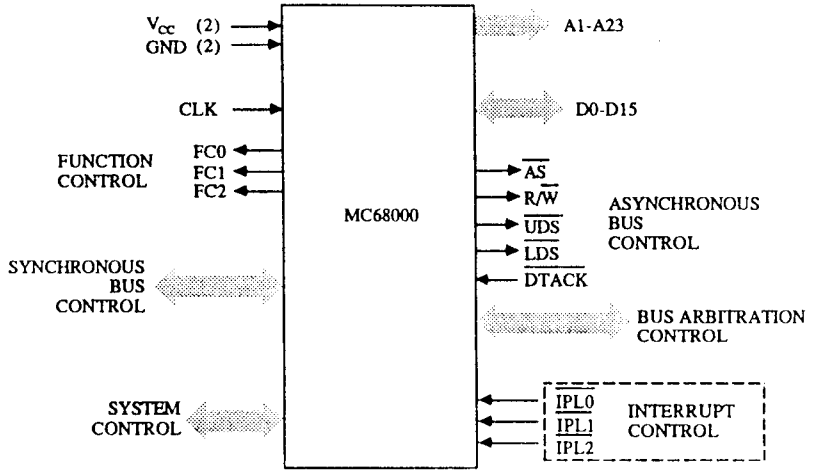
STATUS REGISTER



• INTERRUPT LEVEL IS DESKEWED BEFORE BEING PRESENTED TO COMPARATOR.

MT8-206-1

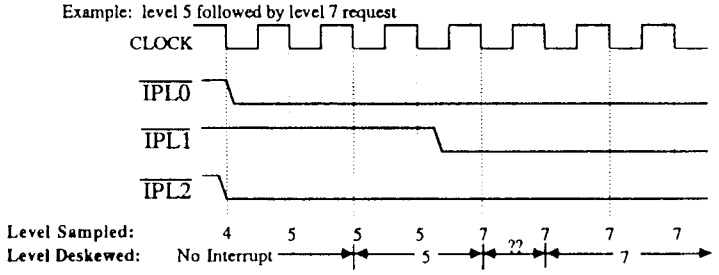
MC68000 SIGNAL LINES



MT8-086-3

MC68000 – INTERRUPTS, DESKEWING LOGIC

- Interrupts are internally deskewed when the interrupt request remains at the same level for two consecutive falling edges of the input clock.
- Interrupts are continuously deskewed on every falling edge of the clock (they are not latched).



MT8-205-1

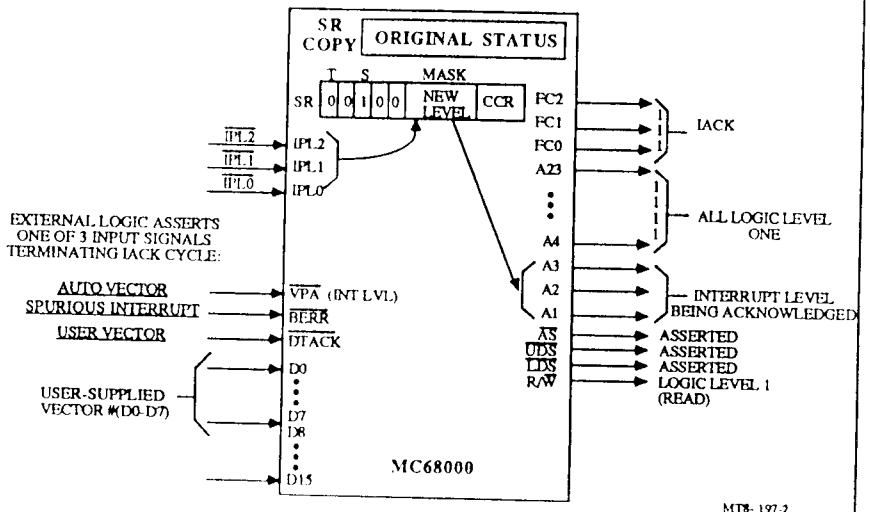
MC68000 – INTERRUPT PRIORITY

REQUESTED INTERRUPT LEVEL	STATE OF PINS $\overline{IPL2}$	$\overline{IPL1}$	$\overline{IPL0}$	MASK LEVEL REQUIRED FOR RECOGNITION
NO. INT.	hi	hi	hi	N/A
1	hi	hi	lo	0
2	hi	lo	hi	1 OR LOWER
3	hi	lo	lo	2 OR LOWER
4	lo	hi	hi	3 OR LOWER
5	lo	hi	lo	4 OR LOWER
6	lo	lo	hi	5 OR LOWER
7	lo	lo	lo	NOT MASKED

- Interrupts allowed only above mask level.
- Level 7 may not be masked out – becomes NMI.
- Levels applied to the pins are inverted with respect to corresponding interrupt mask level.

MT8-205-1-2

MC68000 – IACK BUS CYCLE



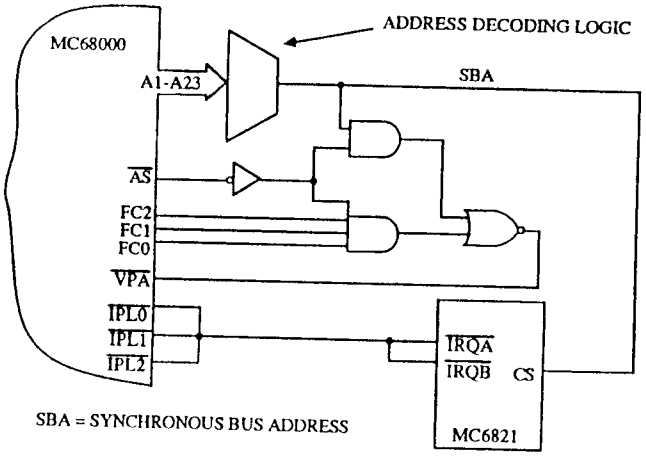
MTB-197-2

MC68000 INTERRUPT RECOGNITION

- IPEND sampling is instruction dependent.
- To ensure an interrupt request is recognized, the request must be held until acknowledge.

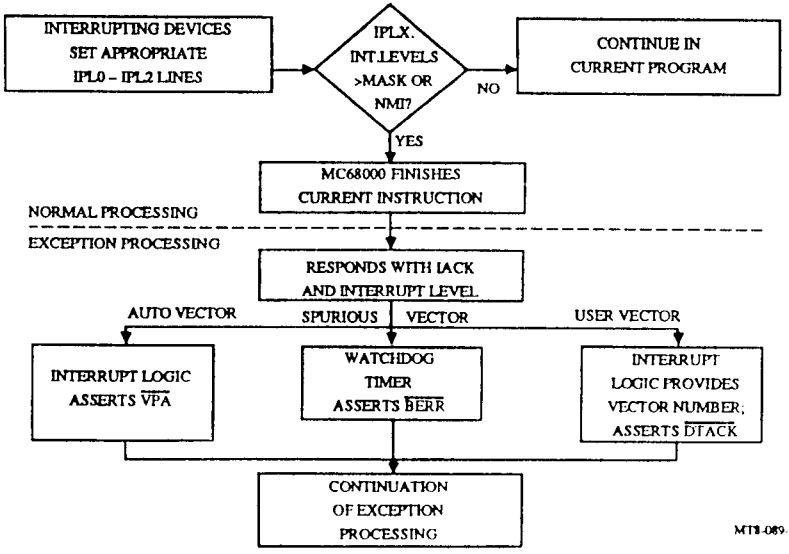
MTB-530-1

MC68000 – SIMPLE INTERRUPT STRUCTURE (AUTO VECTOR)



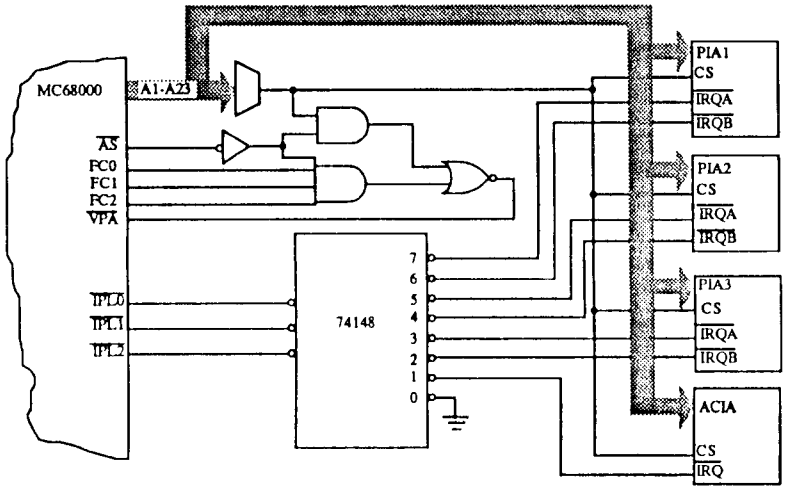
MTB-090-1

MC68000 – BASIC HARDWARE INTERRUPT SEQUENCE



MTB-089-1

MC68000 – MULTIPLE INTERRUPTS 6800 PERIPHERALS



MT8-091-1

MC68000 – USER INTERRUPTS

INTERRUPTING DEVICE RETURNS AN 8-BIT VECTOR NUMBER WHICH IS USED TO CALCULATE THE VECTOR ADDRESS

VECTOR NUMBERS $40-$FF
VECTOR ADDRESSES $100-$3FC

NOTE: USER INTERRUPTING DEVICE IS NOT LIMITED TO ACCESSING ONLY USER INTERRUPT VECTORS

MT8-093

MC68000 - MULTIPLE INTERRUPTS, AUTO VECTORS

VECTOR ADDRESS HEX		VECTOR NO. HEX
	0010 2346	
64		19
	0020 0000	
68		1A
6C	PIA3 IRQA AUTO VECTOR	1B
70	PIA2 IRQB AUTO VECTOR	1C
74	PIA2 IRQA AUTO VECTOR	1D
78	PIA1 IRQB AUTO VECTOR	1E
7C	PIA1 IRQA AUTO VECTOR	1F
$102346	ACIA IRQ SERVICE ROUTINE	
$200000	PIA3 IRQB SERVICE ROUTINE	

MT8-092-1

CALCULATION OF EXCEPTION VECTOR ADDRESS

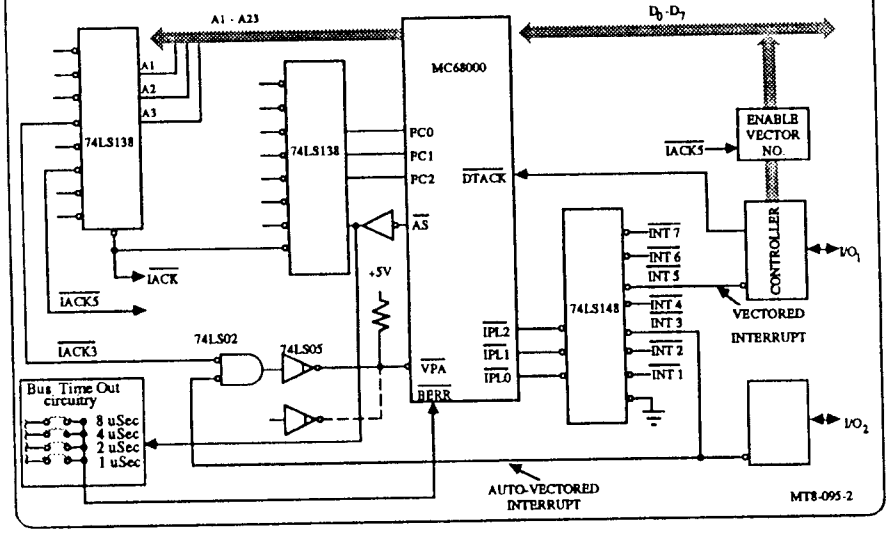
VECTOR NUMBER X 4 = VECTOR ADDRESS

$0F X 4 = $3C

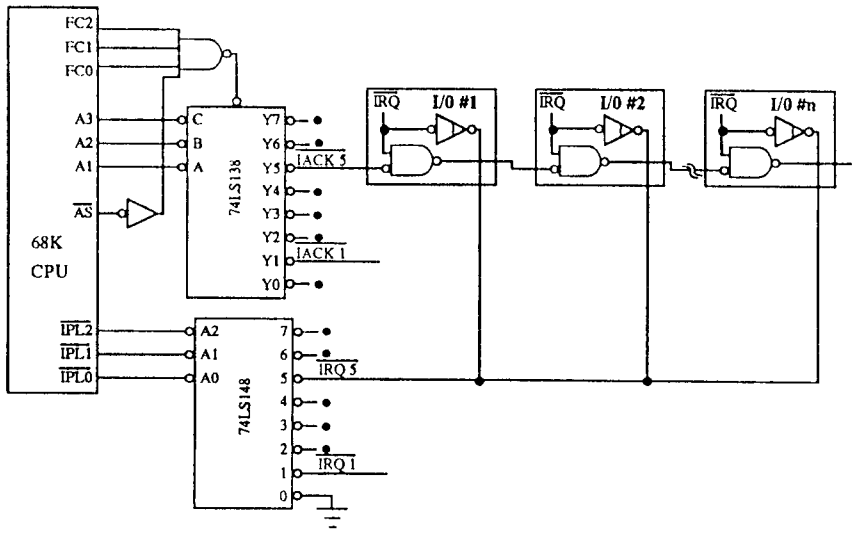
WHEN THE ABOVE VECTOR NUMBER HAS BEEN OBTAINED FROM AN INTERRUPTING DEVICE, WHAT TYPE OF SERVICE HAS BEEN REQUESTED?_____

MT8-094

MC68000 – VECTORED AND AUTO-VECTORED INTERRUPT LOGIC



IACK DAISY-CHAIN



MC68000 – SPURIOUS INTERRUPT

- SPURIOUS INTERRUPT VECTOR IS USED WHEN A BUS ERROR OCCURS DURING INTERRUPT ACKNOWLEDGE.

- PC AND SR ARE STACKED NORMALLY, I.E., AS THOUGH THE INTERRUPT VECTOR FETCH HAD OCCURRED PROPERLY.

MT8-096

MC68000 – BUS ERROR EXAMPLES

ERROR DETECTION:

- MPU ADDRESS REFERENCES NONRESIDENT MEMORY OR INPUT/OUTPUT DEVICE-BUS TIMEOUT
- PARITY ERRORS - RERUN CURRENT BUS CYCLE
- SPURIOUS INTERRUPTS-BERR DURING IACK

MEMORY MANAGEMENT:

- MPU ADDRESS TRANSLATION FAULTS
- ADDRESS SPACE VIOLATIONS-USER ACCESSING SUPERVISOR SPACES
- SECURITY VIOLATIONS OF READ OR WRITE PROTECTED MEMORY

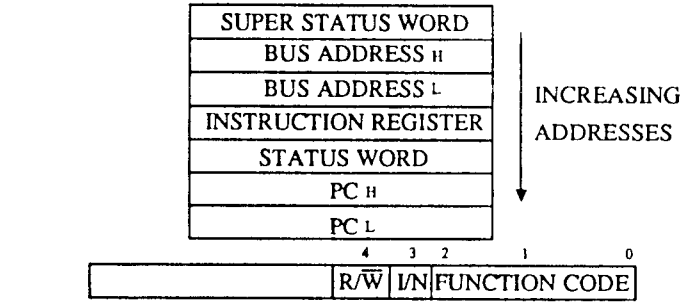
MT8-097-1

MC68000 – ILLEGAL ADDRESS EXAMPLES

- MOVE.W D0,$2001
- MOVE.W D0,$06(A0)
  WHERE A0 HAS ODD NUMBER
- MOVE.L D0,$05(A0)
  WHERE A0 HAS EVEN NUMBER
- MOVE.L D0,$48(A0,D7.W)
  WHERE A0 HAS EVEN NUMBER
  AND D7 HAS ODD NUMBER

MT8-098-1

MC68000 – ILLEGAL ADDRESS AND BUS ERROR STACKING



R/$\overline{W}$ = 1, READ
R/$\overline{W}$ = 0, WRITE

I/N = 0, NORMAL INSTRUCTION OR GROUP 2 EXCEPTION PROCESSING

I/N = 1, GROUP 0 AND GROUP 1 EXCEPTION PROCESSING

MT8-099-2

MC68000 – EXCEPTION PRIORITIES

GROUP 0	• RESET
	• ADDRESS ERROR
	• BUS ERROR
GROUP 1	• TRACE
	• INTERRUPT
	• ILLEGAL INSTRUCTIONS
	• UNIMPLEMENTED INSTRUCTIONS
	• PRIVILEGED INSTRUCTIONS
GROUP 2	• TRAP
	• TRAPV
	• CHK
	• ZERO DIVIDE

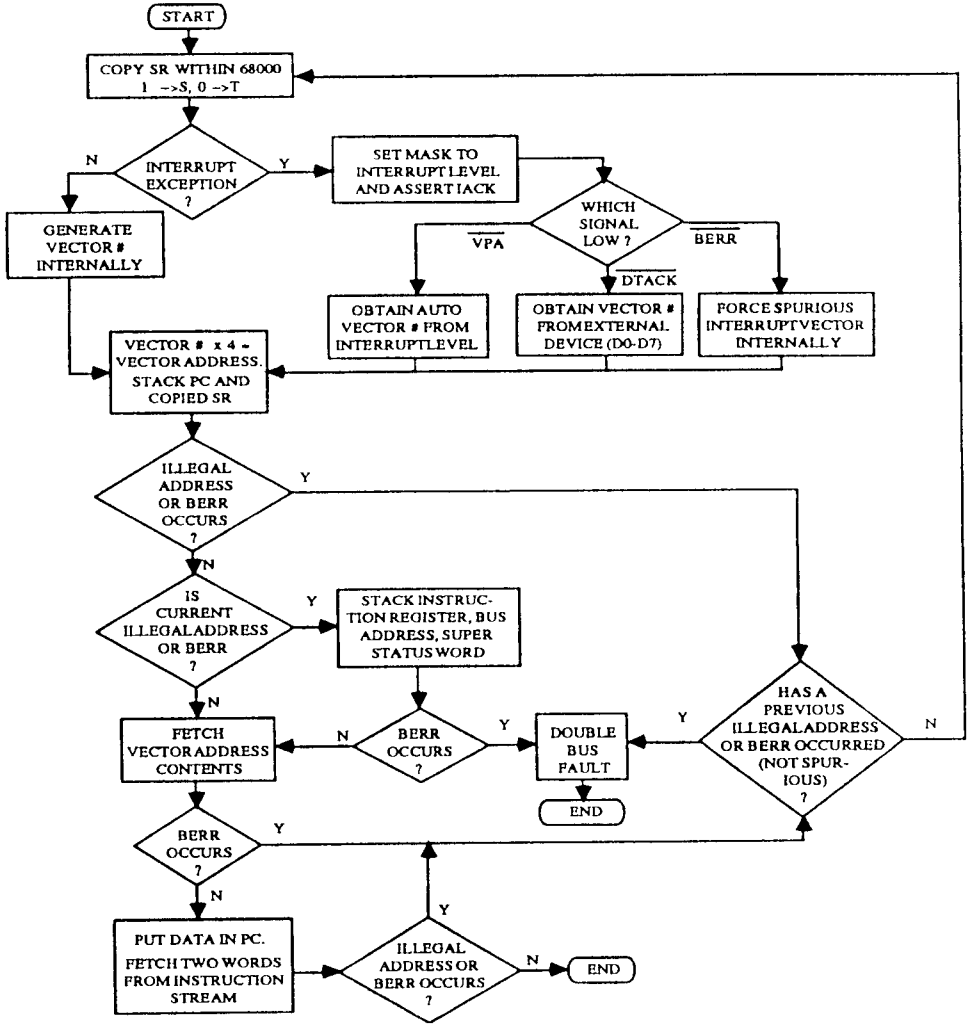
MT8-100-2

MC68000 - RECOGNITION TIMES OF EXCEPTIONS

- END OF A CLOCK CYCLE
  - RESET
  - ADDRESS ERROR
  - BUS ERROR
- END OF A BUS CYCLE
  - ILLEGAL INSTRUCTION
  - UNIMPLEMENTED INSTRUCTION
  - PRIVILEGE VIOLATION
- END OF AN INSTRUCTION CYCLE
  - TRACE EXCEPTION
  - INTERRUPT EXCEPTION
- WITHIN AN INSTRUCTION CYCLE
  - TRAP
  - TRAPV
  - CHK
  - ZERO DIVIDE

MT8-101-3

MC68000–EXCEPTION PROCESSING SEQUENCE (NOT RESET)



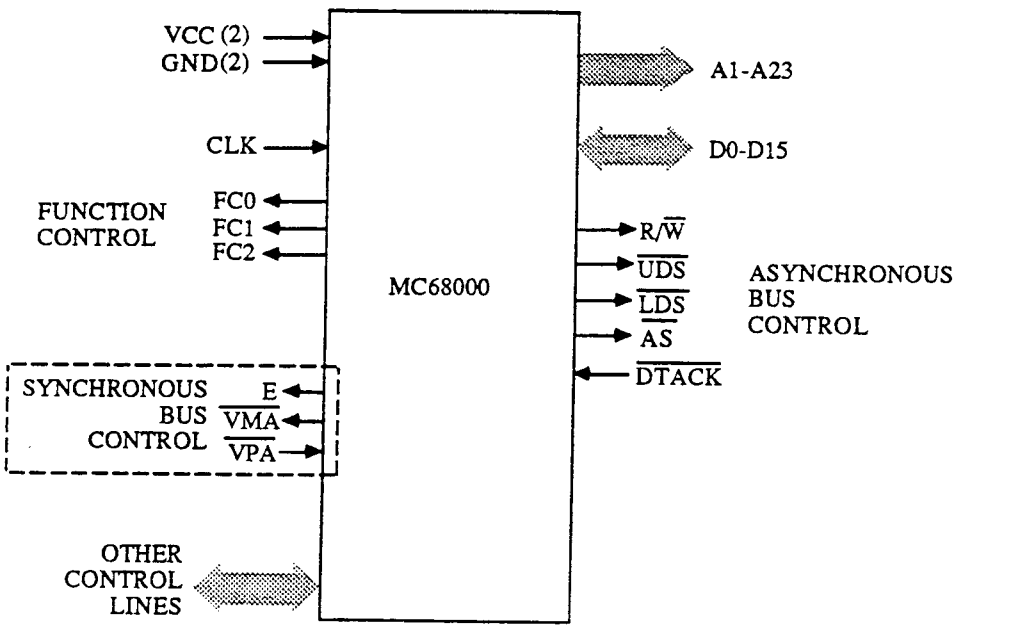
MT8-102-1

MC68000 – 6800 PERIPHERALS

• MC6821	PIA	PERIPHERAL INTERFACE ADAPTER
• MC6840	PTM	PROGRAMMABLE TIMER MODULE
• MC6845	CRTC	CRT CONTROLLER
• MC6847	VDG	VIDEO DISPLAY GENERATOR
• MC6850	ACIA	ASYNCHRONOUS COMMUNICATION INTERFACE ADAPTER
• MC6852	SSDA	SERIAL SYNCHRONOUS DATA ADAPTER
• MC6854	ADLC	ADVANCED DATA LINK CONTROLLER
• MC68488	GPIA	IEEE-488 BUS INTERFACE ADAPTER

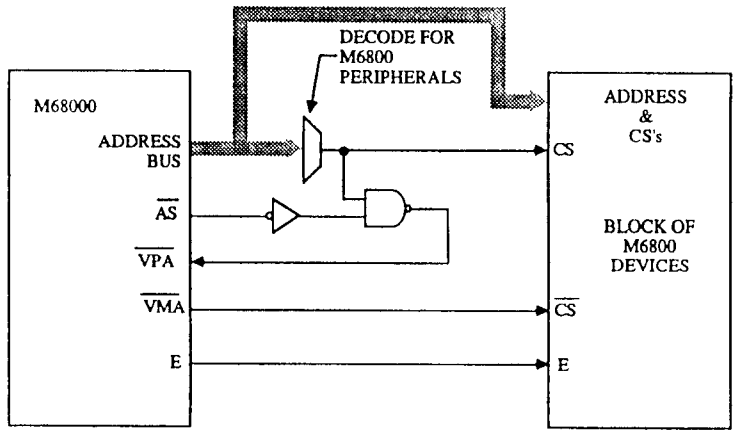
MT8-160-2

MC68000 – MC6800 PERIPHERAL CONTROL PIN DIAGRAM



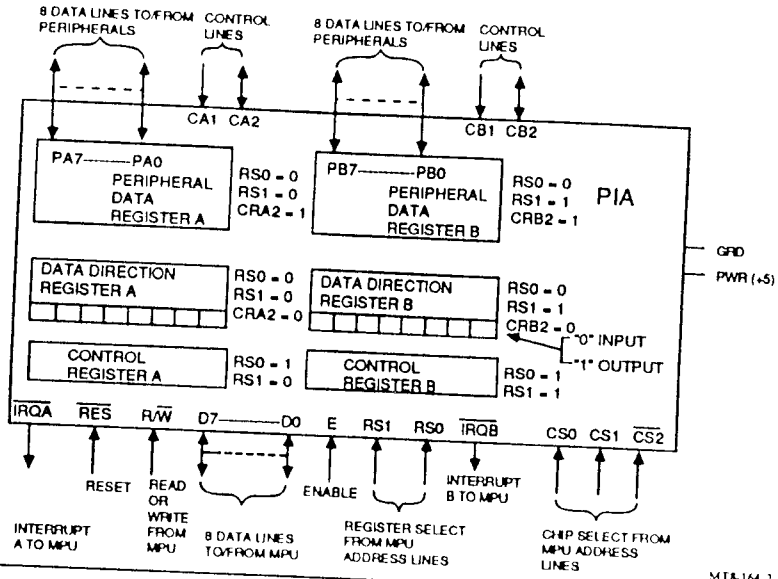
MT8-161-2

MC68000 – CONNECTION OF 6800 PERIPHERALS



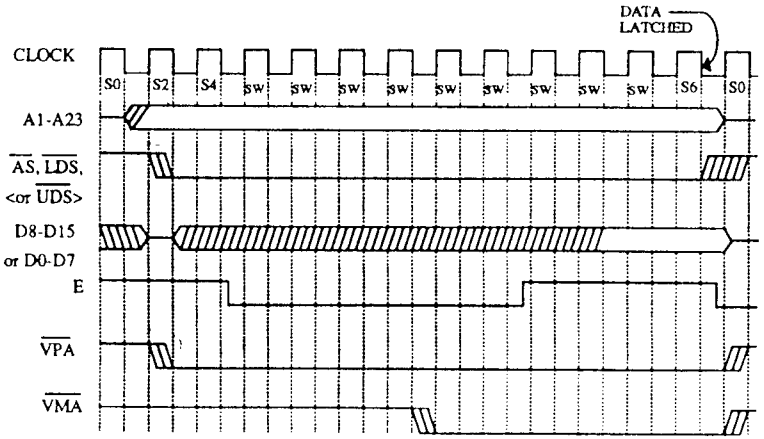
MT8-162-1

MC6821 – PIA DIAGRAM



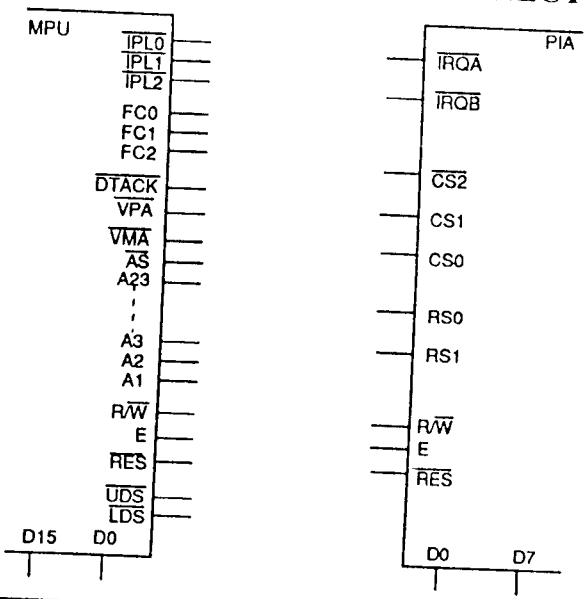
MT8-164-2

MC68000 – 6800 PERIPHERAL CYCLE (READ)



MT8-163-2

MC68000 – PIA INTERCONNECT



MT8-166-1

6.1 DATA TRANSFER OPERATION

Three signals on the processor provide the M6800 interface. They are: enable (E) valid memory address ($\overline{VMA}$), and valid peripheral address ($\overline{VPA}$). Enable corresponds to the E or phase 2 signal in existing M6800 systems. The bus frequency is one tenth of the incoming MC68000 clock frequency. The timing of E allows 1 megahertz peripherals to be used with 8 megahertz MC68000s. Enable has a 60/40 duty cycle; that is, it is low for six input clocks and high for four input clocks. This duty cycle allows the processor to do successive $\overline{VPA}$ accesses on successive E pulses.

M6800 cycle timing is given in Figures 6-2, 6-3, 8-7, and 8-8. At state zero (S0) in the cycle, the address bus is in the high-impedence state. A function code is asserted on the function code output lines. One-half clock later, in state 1, the address bus is released from the high-impedence state.

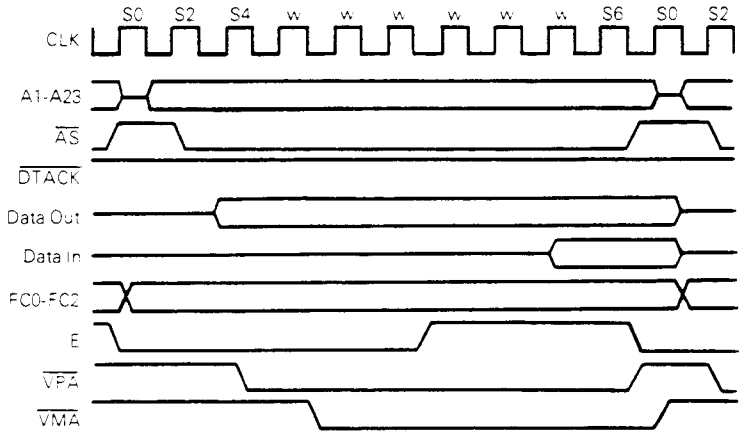


Figure 6-2. MC68000 to M6800 Peripheral Timing — Best Case

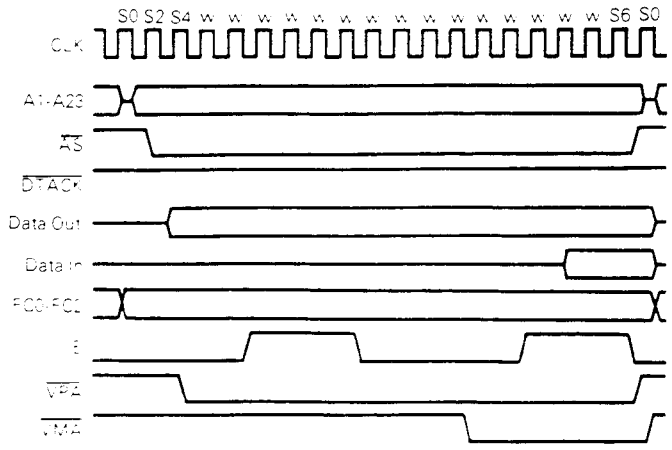
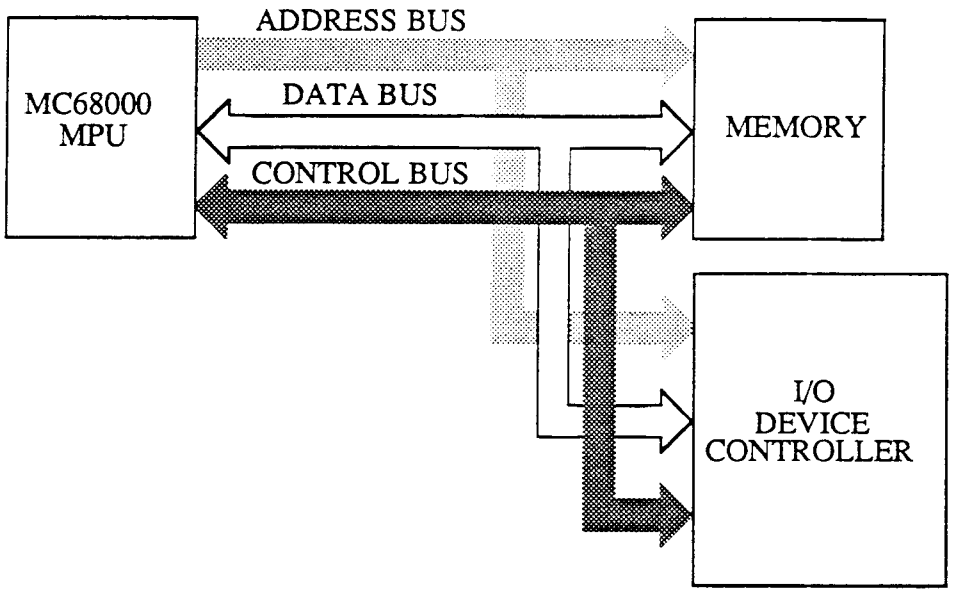


Figure 6-3. MC68000 to M6800 Peripheral Timing — Worst Case

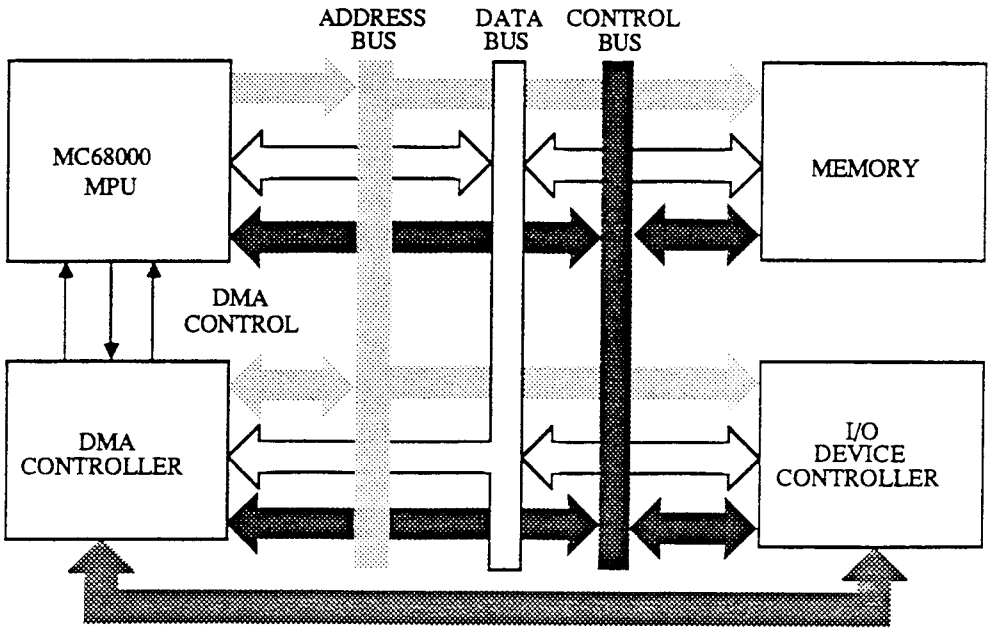
MC68000 – DMA INTRODUCTION



MT8-066-1

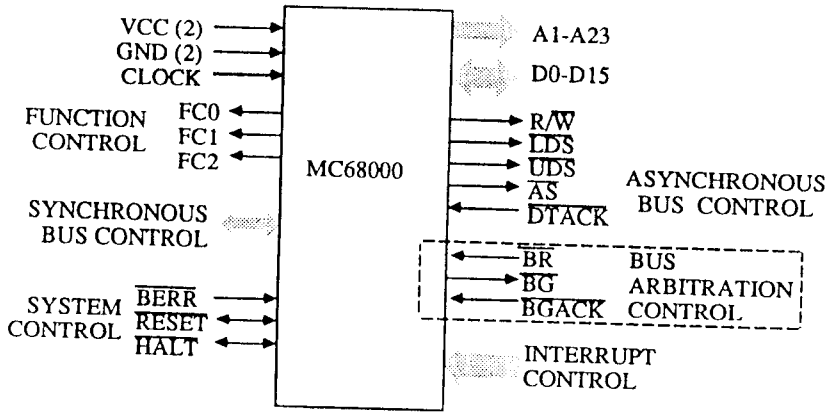
MC68000 – DMA CONCEPT

DIRECT MEMORY ACCESS



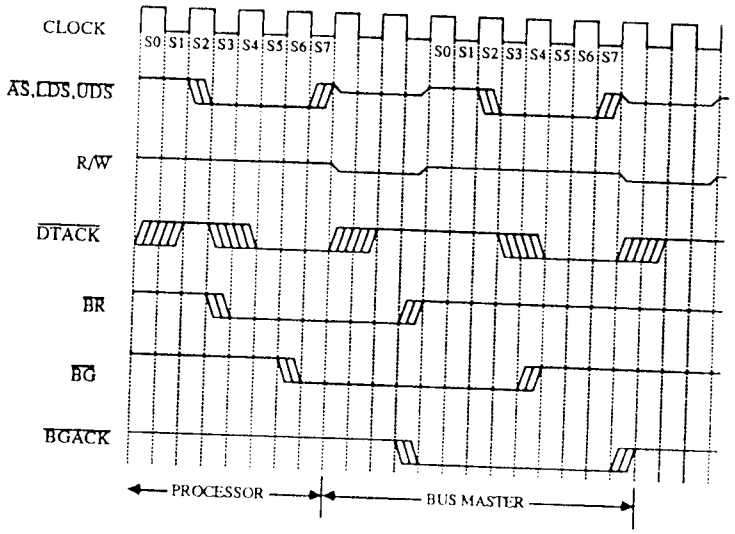
MT8-067-2

MC68000 – BUS ARBITRATION CONTROL PIN DIAGRAM



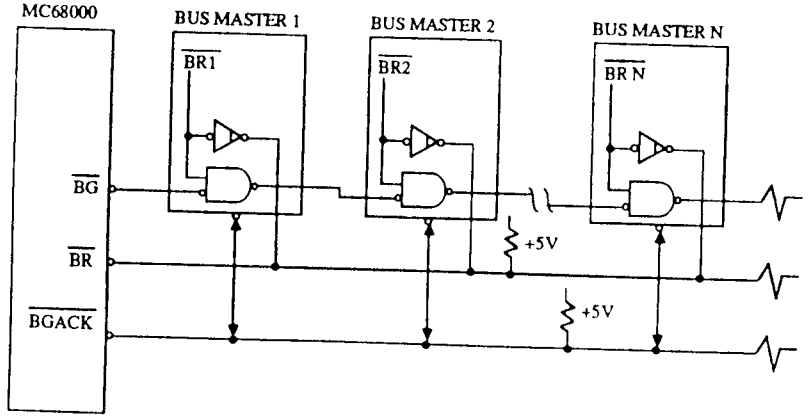
MT8-068-2

MC68000 – BUS REQUEST TIMING



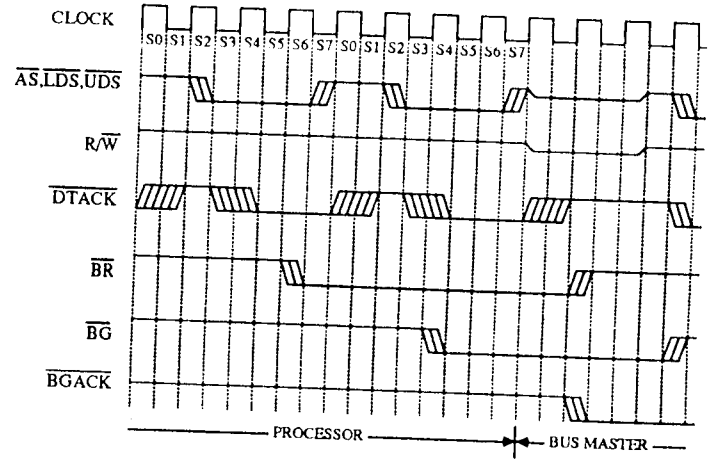
MT8-070-2

DAISY-CHAINED BUS ARBITRATION



MT8-069

MC68000 – BUS REQUEST TIMING DIAGRAM



71-3

MC68000 – BUS ARBITRATION CONTROL CHARACTERISTICS

- $\overline{\text{BGACK}}$ SHOULD NOT BE ASSERTED UNTIL: — SIGNAL STATE CERTIFIES THAT:
  1. $\overline{\text{BG}}$ IS ASSERTED _____
  2. $\overline{\text{AS}}$ IS NEG. (RESCINDED) _____
  3. $\overline{\text{DTACK}}$ IS NEGATED _____
  4. $\overline{\text{BGACK}}$ IS NEGATED _____

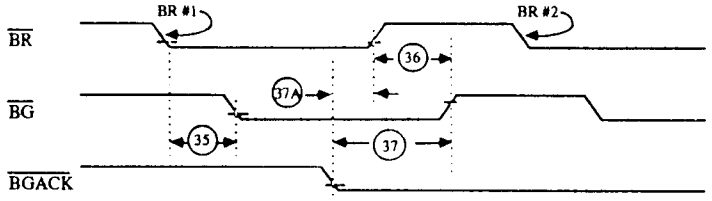
- IF $\overline{\text{BR}}$ IS ASSERTED PRIOR TO THE NEGATION OF $\overline{\text{BGACK}}$, THEN THE MC68000 ISSUES ANOTHER $\overline{\text{BG}}$. NO EXTERNAL BUS CYCLES ARE PERFORMED IN BETWEEN.

- IF $\overline{\text{BGACK}}$ IS NOT ASSERTED AND $\overline{\text{BR}}$ IS NEGATED, THE MC68000 CONTINUES PROCESSING

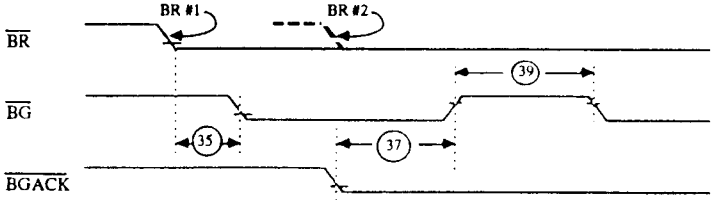
MT8-072-2

MC68000 – Who's on Deck?

MULTIPLE BUS REQUESTS – NON-OVERLAPPING



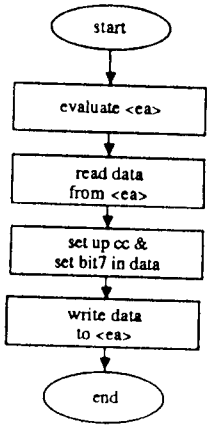
MULTIPLE BUS REQUESTS – OVERLAPPING



MT8- 519-2

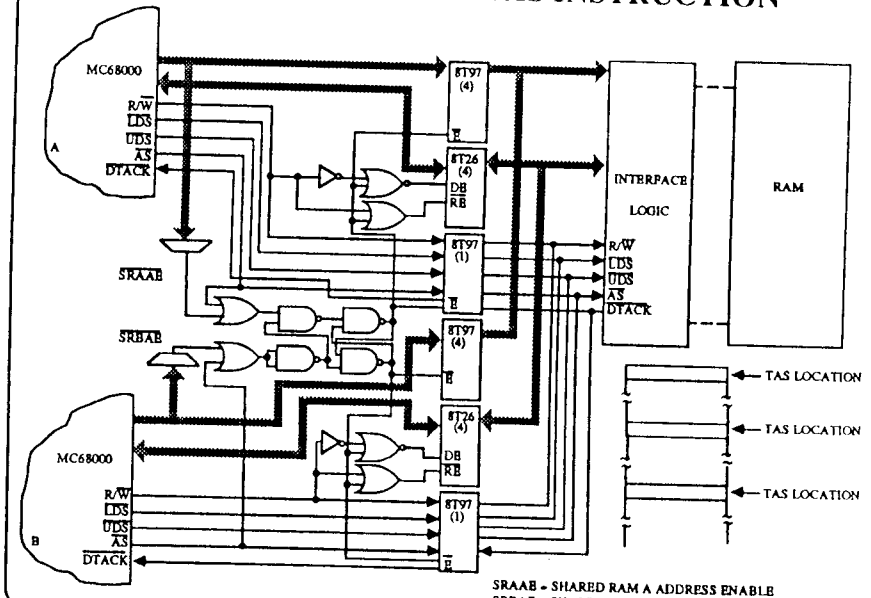
MC68000–TAS INSTRUCTION

TAS	8	(DESTINATION) TESTED→CC 1→DESTINATION AT BIT 7	TAS <ea>	DATA ALTERABLE



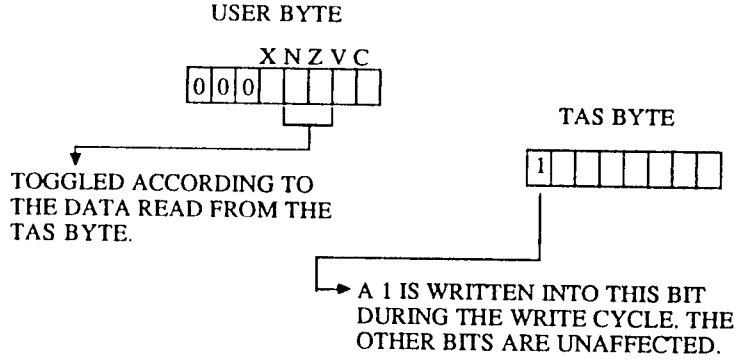
MT8 677-1

MC68000 – USING THE TAS INSTRUCTION



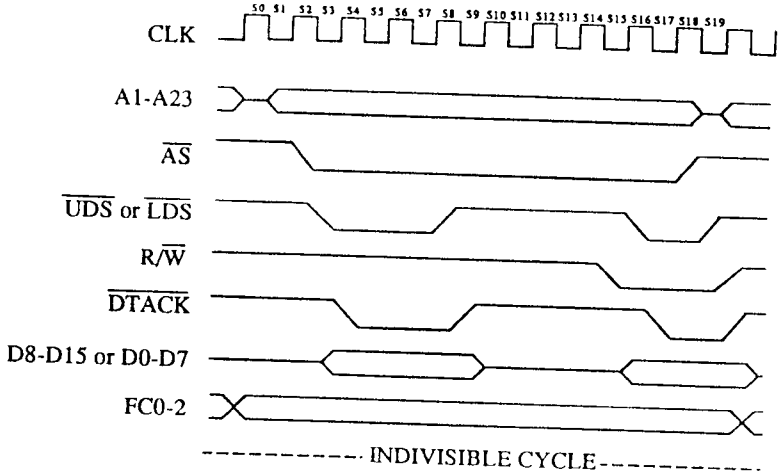
MT8- 153-1

MC68000 – USING THE TAS INSTRUCTION



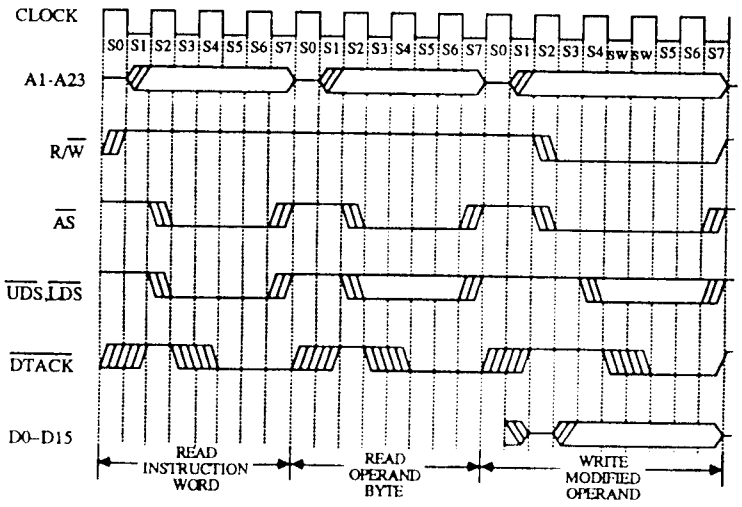
MTB-154

READ-MODIFY-WRITE CYCLE TIMING DIAGRAM (TAS)



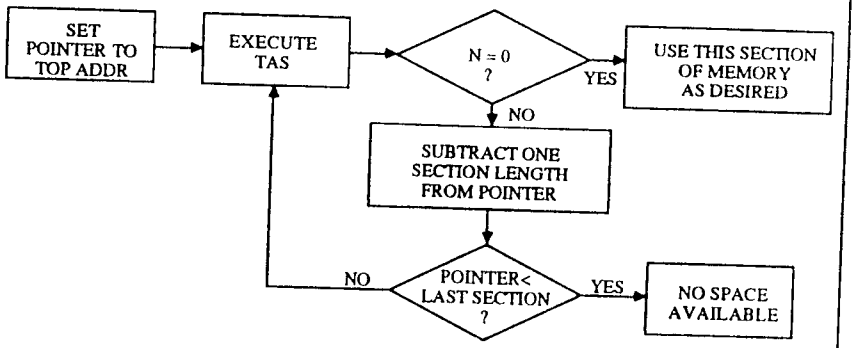
MTB-156-2

MC68000 – READ/MODIFY/WRITE INSTRUCTION (BSET)



MTB-023.1-2

MC68000 – USING THE TAS INSTRUCTION
ALLOCATING MEMORY

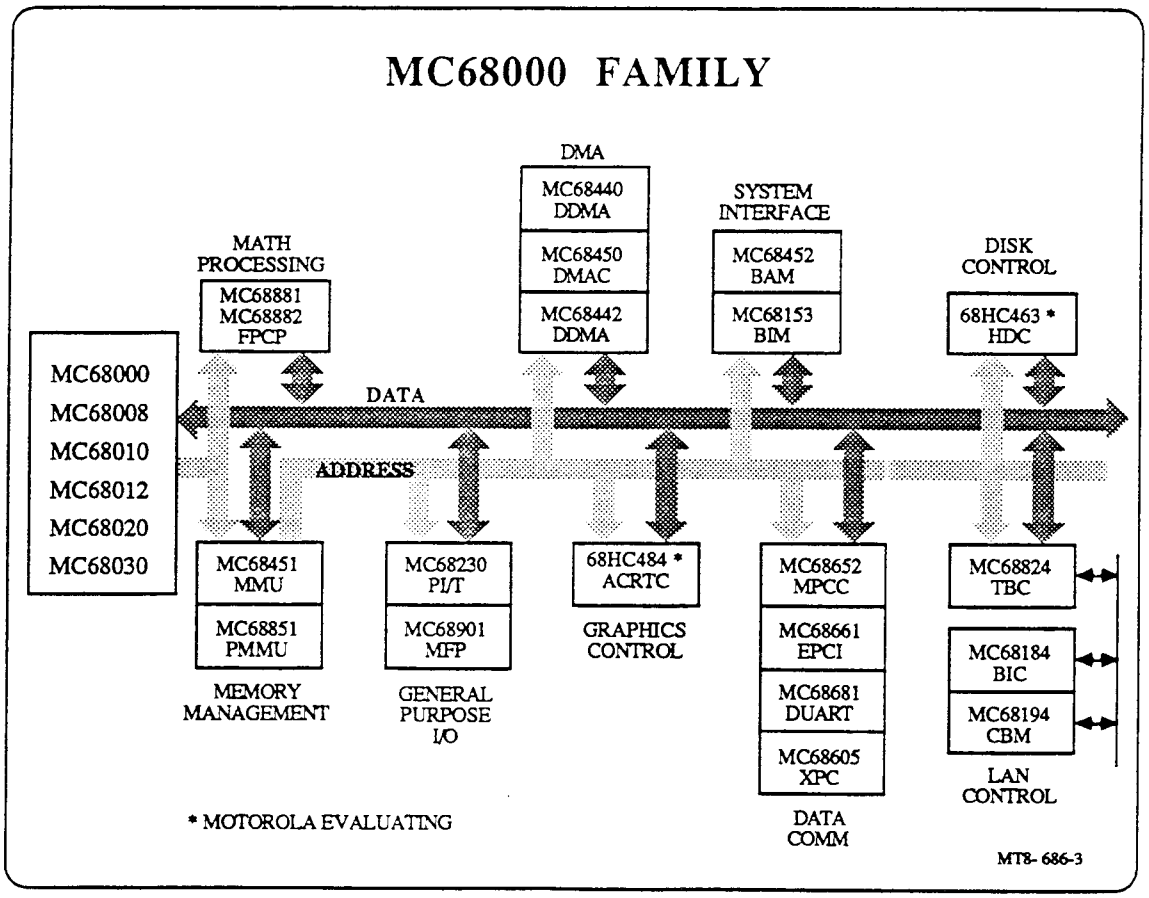


MTB-157-1

MC68000 – USING THE TAS INSTRUCTION

N	Z	LOCATION CONTENTS
0	0	0 (NON-ZERO) ⟶ 1 (NON-ZERO)
0	1	00000000 ⟶ 10000000
1	0	1XXXXXXX ⟶ 1XXXXXXX
1	1	NOT POSSIBLE

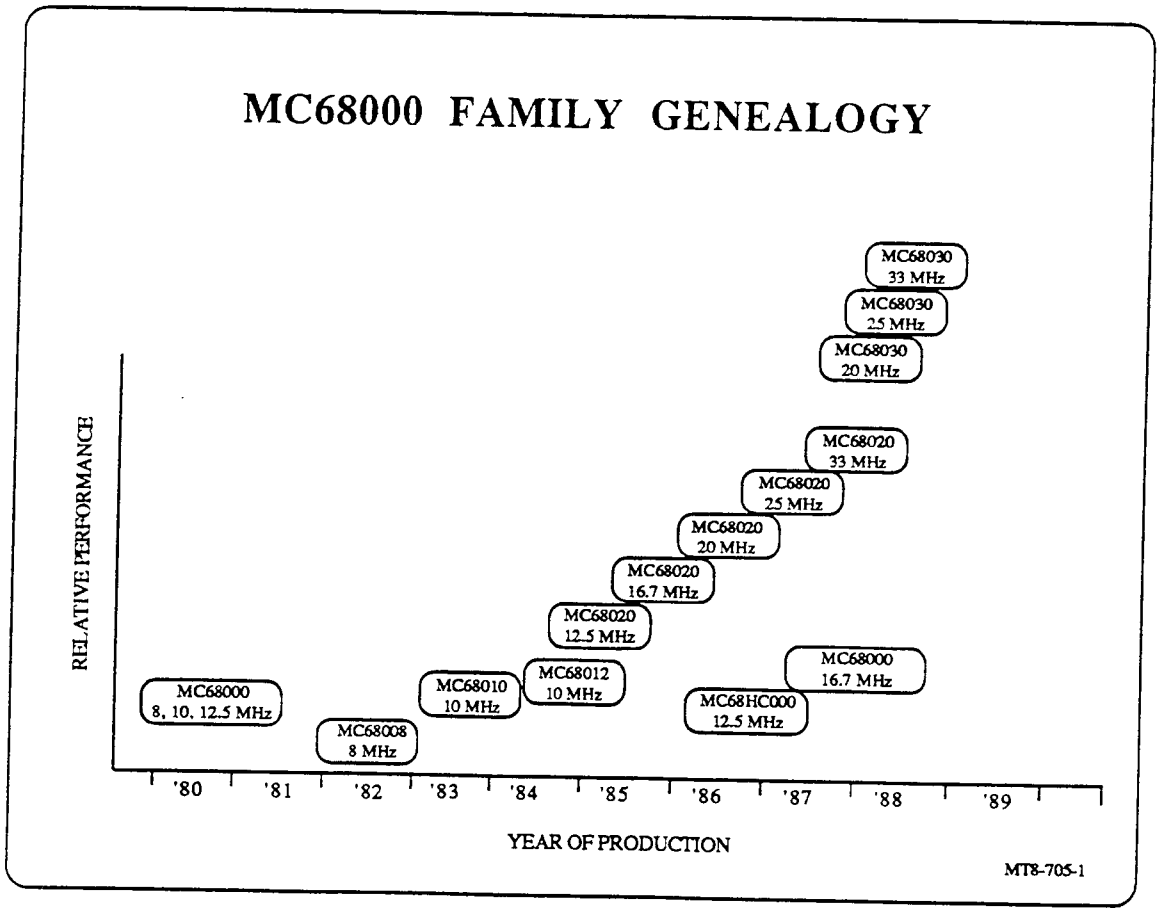
MT8-158-1


MC68000 FAMILY
DMA
MC68440 DDMA
MC68450 DMAC
MC68442 DDMA
SYSTEM INTERFACE
MC68452 BAM
MC68153 BIM
MATH PROCESSING
MC68881 MC68882 FPCP
DISK CONTROL
68HC463 * HDC
MC68000
MC68008
MC68010
MC68012
MC68020
MC68030
DATA
ADDRESS
MC68451 MMU
MC68851 PMMU
MEMORY MANAGEMENT
MC68230 PI/T
MC68901 MFP
GENERAL PURPOSE I/O
68HC484 * ACRTC
GRAPHICS CONTROL
MC68652 MPCC
MC68661 EPCI
MC68681 DUART
MC68605 XPC
DATA COMM
MC68824 TBC
MC68184 BIC
MC68194 CBM
LAN CONTROL
* MOTOROLA EVALUATING
MT8-686-3

MC68000 FAMILY PERIPHERALS

MC68451	MMU	Ideal MMU for non-demand paged MC68000, MC68010 systems.
MC68851	PMMU	32-bit demand virtual paged MMU for MC68020 based systems. Feautres variable page sizes and 45 ns translations.
MC68440	DDMA	Dual channel, high speed Direct Memory Access Controller. Capable of 5 MB/s data rates, and bandwidth controlled auto requesting (LRAR).
MC68450	DMAC	Four channel Direct Memory Access Controller. Capable of very complex "chained" data transfers.
MC68442	DDMA	32-bit version of DDMA. Support for 4 gigabyte range of MC68020. Pin compatible with MC68440/MC68450.
MC68605	XPC	Implements 1984 CCITT X.25 LAPB. Independently generates link level commands and responses using two 22 byte FIFOs and on-chip DMA. Or intelligent HDLC controller.
MC68652	MPCC	Single channel byte control and bit oriented. CRC (error correction) circuitry.
MC68661	EPCI	Universal sync/async. Double buffered reciever/transmitter. Internal baud rate clock.
MC68681	DUART	Dual channel. Quad buffered rcvr. Double buffered xmtr. Independent baud rate selection.
MC68824	TBC	Implements IEEE 802.4 Token Bus Media Access Control which GM MAP specifies as layer 2
MC68184	BIC	Macrocell implementation of digital portion of IEEE 802.4 Broadband Physical Layer.
MC68194	CBM	Bipolar implementation of IEEE 802.4 Carrierband Physical layer.
68HC463	HDC	Hard disk controller, 4 ST506 or 8 SMD drives. (Motorola evaluating)
MC68230	PI/T	Uni/bidirectional, 8/16 bit, double buffered parallel interface. 24-bit timer w/ prescalar.
MC68901	MFP	Single channel USART. 8 source interrupt controller. 8 parallel I/O lines. Four 8-bit timers.
68HC484	ACRTC	8/16-bit CPU interface. 500ns/pixel. 2k x 2k pixels for 16 colors, 4k x 4k for monochrome. (Motorola evaluating)
MC68153	BIM	Routes interrupts from 4 independent sources to any of 7 M68000 MPU interrupt levels.
MC68452	BAM	Arbitrates access of an M68000 system bus between up to 8 local masters.

MT8-685-1


MC68000 FAMILY GENEALOGY
MC68030 33 MHz
MC68030 25 MHz
MC68030 20 MHz
MC68020 33 MHz
MC68020 25 MHz
MC68020 20 MHz
MC68020 16.7 MHz
MC68020 12.5 MHz
MC68000 16.7 MHz
MC68HC000 12.5 MHz
MC68012 10 MHz
MC68010 10 MHz
MC68008 8 MHz
MC68000 8, 10, 12.5 MHz
RELATIVE PERFORMANCE
'80
'81
'82
'83
'84
'85
'86
'87
'88
'89
YEAR OF PRODUCTION
MT8-705-1

EFFECTIVE ADDRESS

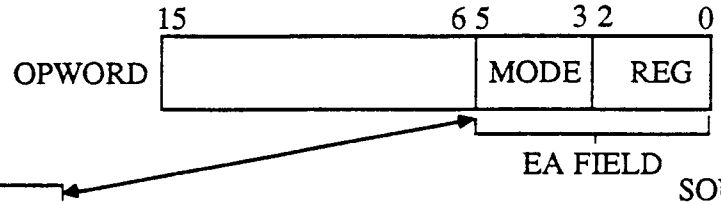
MOST INSTRUCTIONS OPERATE ON AN EFFECTIVE ADDRESS <ea> .

THE <ea> IS SUPPLIED AS A PART OF THE INSTRUCTION, BY THE PROGRAMMER.

THE <ea> IS SPECIFIED BY THE ADDRESSING MODE, ENCODED IN THE OPERATION WORD.

MT8-030

EFFECTIVE ADDRESS



EA	MODE	REG	ADDRESSING MODE	SOURCE SYNTAX
1	000	REG#	DATA REGISTER DIRECT	Dn
2	001	REG#	ADDRESS REGISTER DIRECT	An
3	010	REG#	ADDRESS REGISTER INDIRECT (ARI)	(An)
4	011	REG#	ARI WITH POSTINCREMENT	(An) +
5	100	REG#	ARI WITH PREDECREMENT	-(An)
8	111	000	ABSOLUTE SHORT	$XXXX
9	111	001	ABSOLUTE LONG	$XXXXXXXX
12	111	100	IMMEDIATE	#XXXX

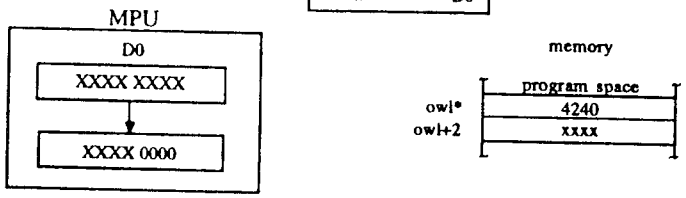
MT8-720-1

REGISTER DIRECT

USED TO ACCESS AN INTERNAL REGISTER

- EA=Dn or An
- AN EXAMPLE

CLR.W D0



- opword decode

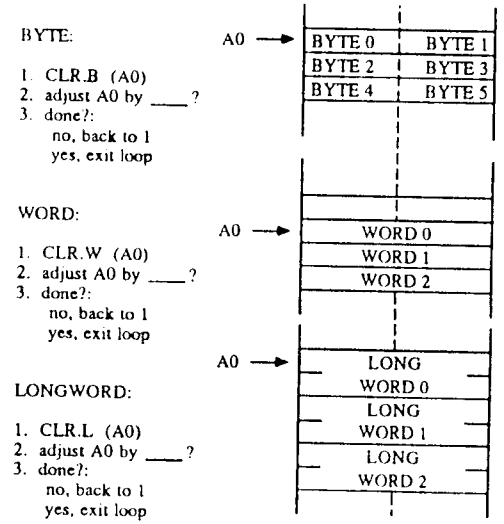
GIVEN	DATA SIZE	MODE	REG
0100 0010	01	000	000
4 2	WORD	DIRECT	D0

EA

* opword location

MTB-032-1

ADJUSTING ADDRESS REGISTERS



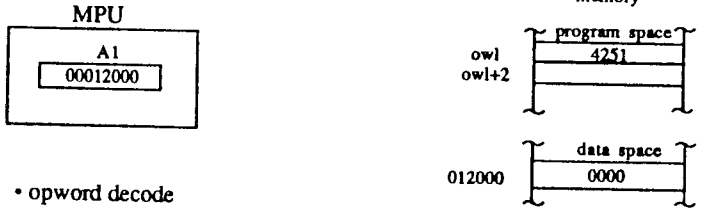
MTB-202-1

ADDRESS REGISTER INDIRECT

USED AS A VARIABLE REFERENCE POINTER TO MEMORY

- EA=(An)
- An Example

CLR.W (A1)



- opword decode

GIVEN	SIZE	MODE	REG
42	01	010	001
	Word	Indirect	A1

EA

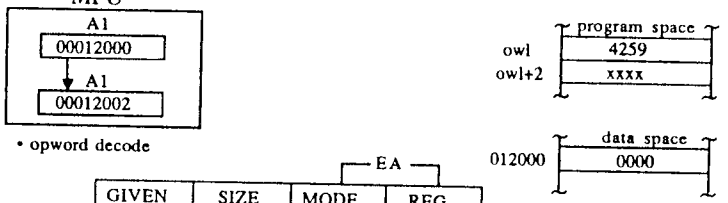
MTB-033-2

ADDRESS REGISTER INDIRECT WITH POST INCREMENT

USED TO PROCESS SEQUENTIAL DATA, STRINGS, BLOCK MOVES; PERFORM UNSTACKING OPERATIONS

- EA=(An)* An is incremented after use
- An Example

CLR.W (A1)+ *



- opword decode

GIVEN	SIZE	MODE	REG
01000010	01	011	001
	Word	Indirect Post-Increment	A1

EA

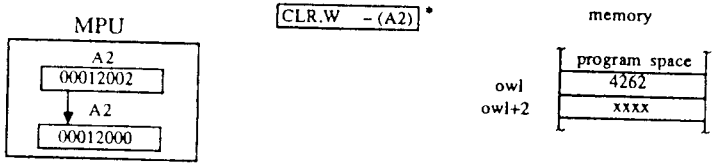
*Post increment : by 1 for a byte data size
by 2 for a word data size
by 4 for a longword data size

MTB-035-4

ADDRESS REGISTER INDIRECT WITH PREDECREMENT

USED TO PROCESS SEQUENTIAL DATA, STRINGS, MULTIPLE PRECISION DATA; PERFORM STACKING OPERATIONS

- EA=(An - P)* predecrement An before use.
- An Example



- opword decode

	EA		
GIVEN	SIZE	MODE	REG
01000010	01	100	010
	Word	Indirect Predecrement	A2

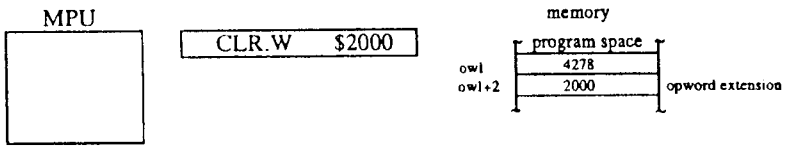
*Predecrement: by 1 for a byte data size
by 2 for a word data size
by 4 for a longword data size

MT8-034-3

ABSOLUTE SHORT

DEFINES A PERMANENT ADDRESS WITHIN A 64 KBYTE RANGE: THE LOCATION MAY BE ONE REQUIRING FREQUENT REFERENCES

- EA = XXXX where 0 ≤XXXX≤$7FFF OR $FF8000≤XXXX≤$FFFFFF
- AN EXAMPLE



- OPWORD DECODE

GIVEN	SIZE	MODE	REG
01000010	01	111	000
	WORD	ABSOLUTE SHORT	

002000 data space 0000

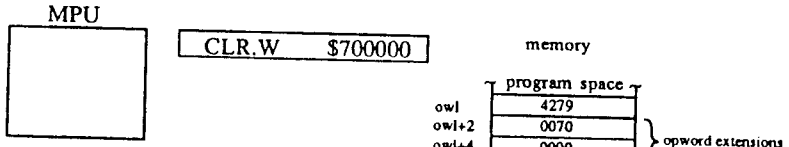
- ABSOLUTE SHORT ADDRESS IS SIGN EXTENDED TO A LONG WORD.

MT8-042-2

ABSOLUTE LONG

USED TO SPECIFY A PERMANENT OR PREASSIGNED ADDRESS WHICH CAN REFERENCE ANY LOCATION WITHIN ENTIRE MAP

- EA = XXXXXX where 0 ≤ XXXXXX ≤ $FFFFFF
- AN EXAMPLE



- OPWORD DECODE

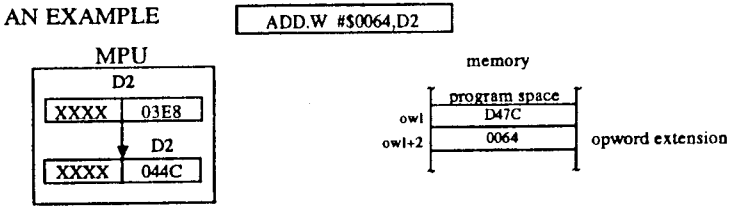
GIVEN	SIZE	MODE	REG
01000010	01	111	001
	WORD	ABSOLUTE LONG	

MT8-041-3

IMMEDIATE

USED TO SPECIFY CONSTANT VALUES; INITIALIZE MPU REGISTERS, EXTERNAL MEMORY LOCATIONS AND I/O DEVICES

- EA = #XXXX
- AN EXAMPLE



- OPWORD DECODE

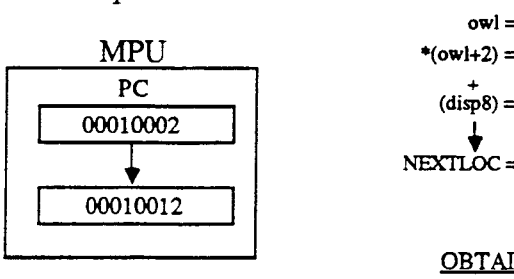
			EA	
GIVEN	DATA REG	SIZE	MODE	REG
1101	010	001	111	100
	D2	WORD	IMMEDIATE	

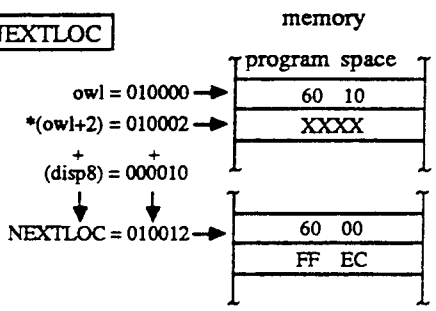
MT8-043-1

PROGRAM COUNTER RELATIVE FOR BRANCHES

- EA = PC + d
- An Example

BRA NEXTLOC





OBTAIN EA OF SECOND BRANCH INST:

*(owl+2) =

\+
(disp) =

NEW PC =

- OPWORD DECODE

GIVEN	GIVEN	d_8
0110	0000	00010000

d_{16} if $d_8 = 0$

* OPWORD LOCATION +2 is the relative address to which the displacement is added.

The displacement is SIGNED and is EXTENDED to a long word before it is added to the PC.

MT8-044-2

INSTRUCTIONS WITH TWO EFFECTIVE ADDRESSES

- GENERAL FORM
  INST SOURCE EA, DESTINATION EA
- GENERAL OPWORD

GIVEN	DEST EA		SOURCE EA	
INST	REG	MODE	MODE	REG

- EXAMPLES

  MOVE.W (A3), D1

0011	001	000	010	011

  MOVE.L A5, – (A1)

0010				

MT8-045

DATA

MOVE.s	Source, Destination (Source) ⟶ Destination
EXG.L	Rx, Ry (Rx) ⟷ (Ry)
SWAP.W	Dn Dnlow ⟷ Dnhigh
EXT.s	Dn Sign Extend Dn Byte to Word Sign Extend Dn Word to Long Word

MT8-050-1

MOVE INSTRUCTION

MOVE.s <ea> , <ea>

SOURCE DESTINATION

- WHERE 's' SPECIFIES LENGTH OF DATA BEING TRANSFERRED

B – BYTE

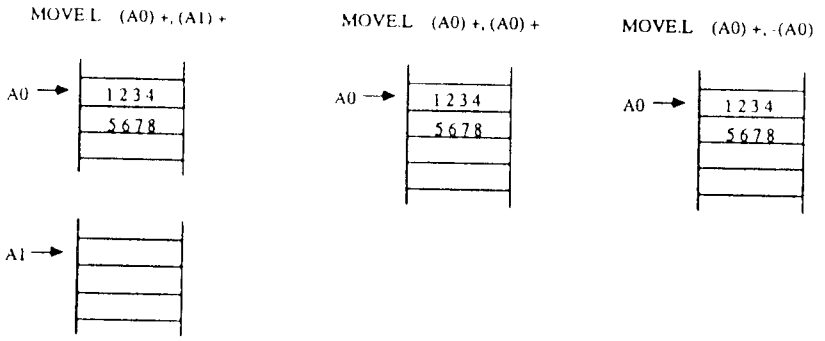
W – WORD (DEFAULT)

L – LONG WORD

MT8-051-1

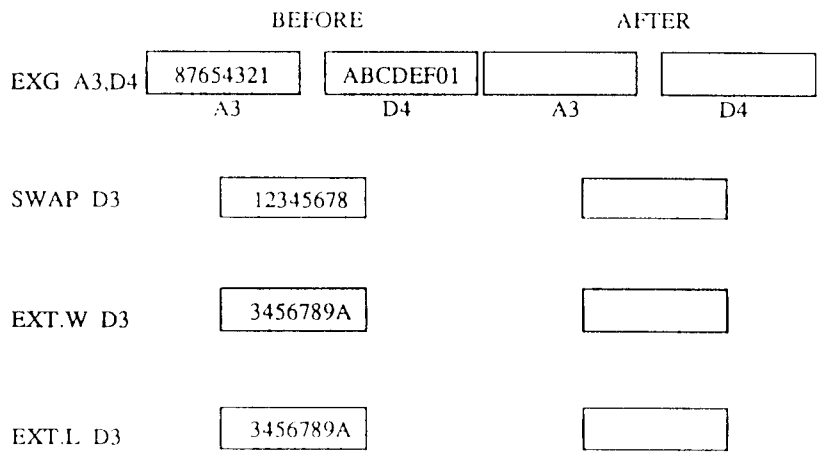
MC68000 – POST-INCREMENT AND PRE-DECREMENT BEHAVIOR

- SHOW RESULTING MEMORY CONTENTS
- SHOW RESULTING POINTER LOCATION



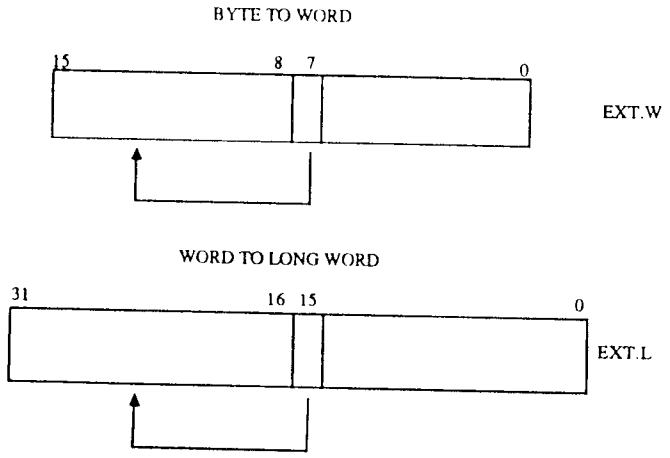
MTB-207

MC68000 – DATA INSTRUCTION EXAMPLES



MTB-522

MC68000 – EXT INSTRUCTION



MTB-052

INTEGER ARITHMETIC

ADD.s* Source, Destination
(Destination) + (Source) → Destination

SUB.s Source, Destination
(Destination) – (Source) → Destination

CMP.s Source, Destination
(Destination) – (Source)

TST.s Destination
(Destination) – 0

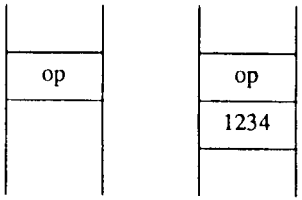
CLR.s Destination
0 → Destination

* s = Data Size

MTB-047-

ADD/SUB INSTRUCTION EXAMPLES

```
ADD.W   (A0),D1
ADD.W   D1,(A0)
ADD.B   D1,(A0)
SUB.W   D1,(A0)
SUB.W   D1,$1234
SUB.W   D1,#$1234
TST.B   D0
CMP.B   #0,D0
```



MTB-680

EXAMPLE RESULTS OF MULTIPLY OPERATION

MULS #2,D0

FFFE
x 0002
FFFFFFFC

D0 before: XXXX FFFE

D0 after:

MULU #2,D0

FFFE
x 0002
0001FFFC

D0 before: XXXX FFFE

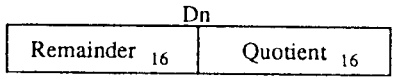
D0 after:

MTB-521

ARITHMETIC

MULU.W SOURCE, Dn
MULS.W SOURCE, Dn
$(Dn)_{16} * (SOURCE)_{16} \rightarrow Dn_{32}$

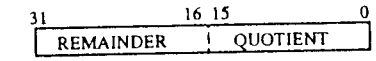
DIVU.W SOURCE, Dn
DIVS.W SOURCE, Dn
$(Dn)_{32} / (SOURCE)_{16} \rightarrow Dn$



MTB-048

MC68000 – DIVIDE INSTRUCTION DIVU, DIVS

- DIVIDEND IS A 32 BIT DATA REGISTER
- DIVISOR IS A 16 BIT EFFECTIVE ADDRESS (DATA)
- THE RESULT IS IN THE SPECIFIED DATA REGISTER



- IF THE DIVISOR IS ZERO, AN EXCEPTION OCCURS
- IF OVERFLOW IS DETECTED,

$V \longrightarrow 1$

AND OPERANDS ARE UNAFFECTED

MTB-049-1

EXAMPLE RESULTS OF DIVIDE OPERATION

DIVS #2,D3

Q FFFF R 0000

0002) FFFFFFFE

D3 before: FFFF FFFE

D3 after: 0000 FFFF

DIVU #2,D3

Q 7FFFFFFF R 0

0002) FFFFFFFE

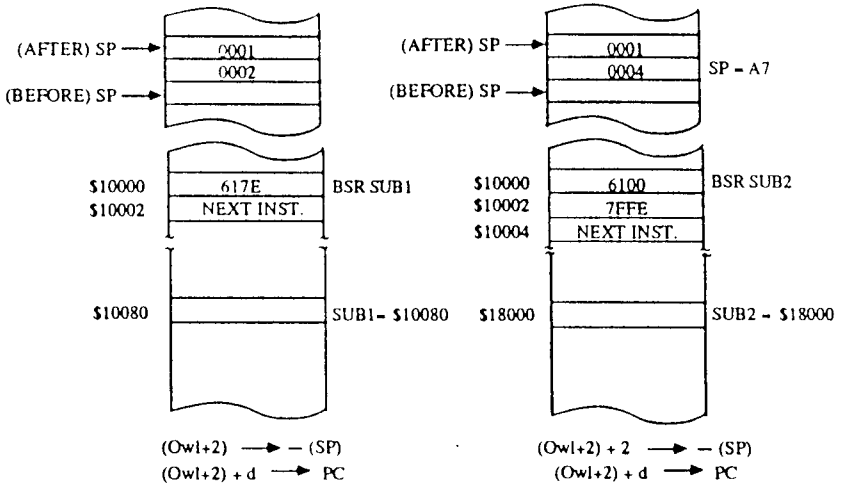
D3 before: FFFF FFFE

D3 after: FFFF FFFE

V = 1

MT8-681-1

MC68000 – BSR INSTRUCTION



MT8-054-2

CONTROL

NOP	No Operation
Bcc	<Label> IF cc is true Then d + Owl + 2 → PC Else Next Instruction -32768 ≤ d ≤ + 32766
BRA	<Label> d + Owl + 2 → PC
BSR	<Label> PC → -(A7) d + Owl + 2 → PC
RTS	(A7) + → PC

MT8-053-1

CONDITIONAL BRANCH INSTRUCTIONS

Bcc	CONDITION	CCR TEST	r = NEGATIVE
BMI	MINUS	N = 1	r = POSITIVE
BPL	PLUS	N = 0	r = SIGN ERR
*BVS	OVERFLOW	V = 1	r = SIGN OK
*BVC	NO OVERFLOW	V = 0	
*BLT	LESS	[N⊕V] = 1	d < s
*BGE	GREATER OR EQUAL	[N⊕V] = 0	d ≥ s
*BLE	LESS OR EQUAL	[Z + (N⊕V)] = 1	d ≤ s
*BGT	GREATER	[Z + (N⊕V)] = 0	d > s
BEQ	EQUAL	Z = 1	d = s
BNE	NOT EQUAL	Z = 0	d ≠ s
BHI	HIGHER	[C + Z] = 0	d > s
BLS	LOWER OR SAME	[C + Z] = 1	d ≤ s
BCC (BHS)	CARRY CLEAR	C = 0	d ≥ s
BCS (BLO)	CARRY SET	C = 1	d < s

EXAMPLE: CMP.X s,d (d - s = r); CCR BITS SET UP

Bcc LABEL Bcc TESTS RESULTS OF CMP.X IN CCR;

SPECIFY APPROPRIATE Bcc TO DETERMINE IF:

*2's COMPLEMENT ARITHMETIC

MT8-055-1

ADD EXAMPLES

1. EXECUTE THE FOLLOWING PROGRAM AND SHOW RESULTS.

DATA REG D0: 9C2F8A70

CCR: 0 0 0 1 1

			X N Z V C
MOVE.W	#$FF00, D0		
ADD.W	#$0300, D0		
ADD.B	#$FE, D0		

2. REPLACE D0 WITH (A0) IN THE ABOVE PROGRAM AND SHOW THE RESULTS.

(A0) → MEMORY

CCR X N Z V C

MTB-056

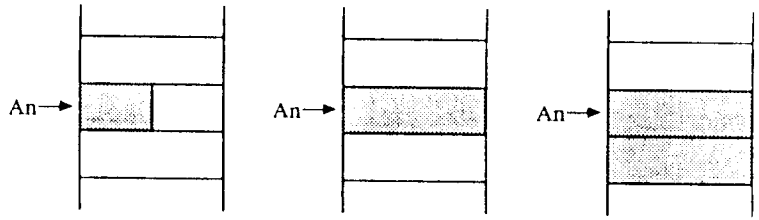
SUMMARY

	X	N	Z	V	C	
ADD.X	•	•	•	•	•	d + s → d
SUB.X	•	•	•	•	•	d – s → d
CMP.X	–	•	•	•	•	d – s
TST.X	–	•	•	0	0	<EA> – 0
CLR.X	–	0	1	0	0	0 → <EA>
MULU.W	–	•	•	0	0	Dn * <EA> → Dn
MULS.W	–	•	•	0	0	Dn * <EA> → Dn
DIVU.W	–	•	•	•	0	Dn ÷ <EA> → Dn
DIVS.W	–	•	•	•	0	Dn ÷ <EA> → Dn
MOVE.X	–	•	•	0	0	s → d
EXG	–	–	–	–	–	Rx ↔ Ry
SWAP	–	•	•	0	0	Dn low ↔ Dn high
EXT.X	–	•	•	0	0	BIT 7 EXTENDS TO BIT 15 BIT 15 EXTENDS TO BIT 31
NOP	–	–	–	–	–	NO OPERATION
Bcc	–	–	–	–	–	PC + d → PC
BRA	–	–	–	–	–	PC + d → PC
BSR	–	–	–	–	–	RETURN ADD → -(A7) PC + d → PC
RTS	–	–	–	–	–	(A7) + → PC

• UPDATES
– NO CHANGES
d DESTINATION
s SOURCE
EA EFFECTIVE ADDRESS

MTB-057

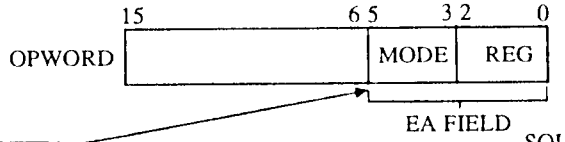
MC68000 – LOCATION OF OPERANDS IN MEMORY



The <ea> locates the most significant byte of the operand in memory REGARDLESS of operand size

MTB-504-3

EFFECTIVE ADDRESS SUMMARY



EA	MODE	REG	ADDRESSING MODE	SOURCE SYNTAX
1	000	REG#	DATA REGISTER DIRECT	Dn
2	001	REG#	ADDRESS REGISTER DIRECT	An
3	010	REG#	ADDRESS REGISTER INDIRECT (ARI)	(An)
4	011	REG#	ARI WITH POSTINCREMENT	(An) +
5	100	REG#	ARI WITH PREDECREMENT	-(An)
8	111	000	ABSOLUTE SHORT	$XXXX
9	111	001	ABSOLUTE LONG	$XXXXXXXX
12	111	100	IMMEDIATE	#XXXX

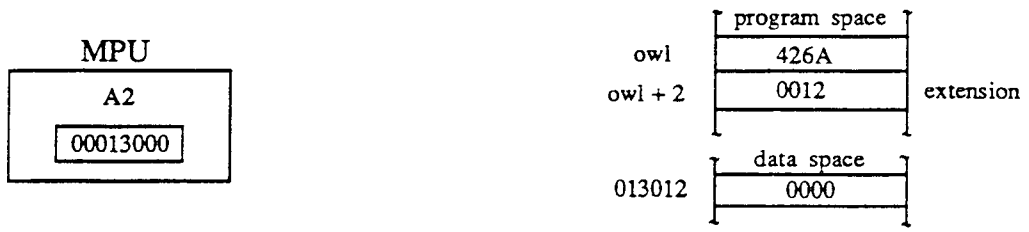
MTB-701

ADDRESS REGISTER INDIRECT WITH DISPLACEMENT

USED TO REFERENCE ELEMENTS WITHIN AN ARRAY; ACCESS INDIVIDUAL I/O LOCATIONS WITHIN A BLOCK OF I/O DEVICES

- EA = (An) + displacement (word)
- An Example

```
CLR.W    $12(A2)
```



- OP WORD DECODE

	EA		
GIVEN	SIZE	MODE	REG
01000010	01	101	010
	word	ADD. REG. IND. with DISP.	A2

- EXTENSION

0000	0000	0001	0010

•Displacement is always a word sign Extended to a long word.

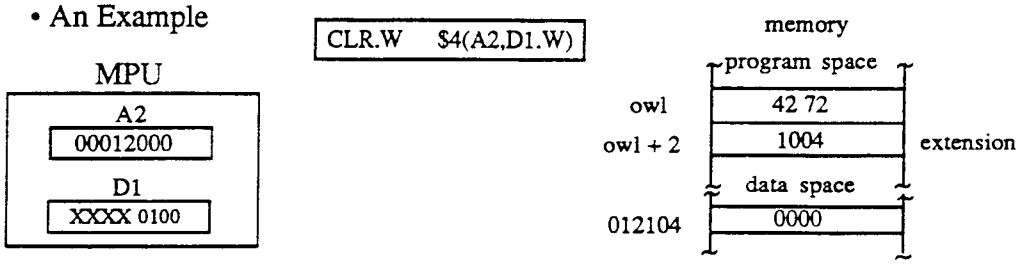
MT8-036-2

ADDRESS REGISTER INDIRECT with INDEX (and DISPLACEMENT)

USED TO REFERENCE DATA WITHIN COMPLEX STRUCTURES;
Eg: ACCESSING SINGLE OR MULTIPLE FIELDS WITHIN MULTIPLE RECORD ARRAYS

- EA = (An) + (Rx) + displacement (byte)
- An Example

```
CLR.W    $4(A2,D1.W)
```



- OPWORD DECODE

	EA		
GIVEN	SIZE	MODE	REG
01000010	01	110	010
	WORD	ADD. REG. IND. WITH INDEX	A2

- EXTENSION WORD DECODE

D/A	REG#	SIZE	FIXED	DISPLACEMENT
0	001	0	000	0000 0100

• Reference Programmer's Manual

MT8-038-3

ADDRESS REGISTER INDIRECT WITH INDEX (AND DISPLACEMENT)

- EA CALCULATION

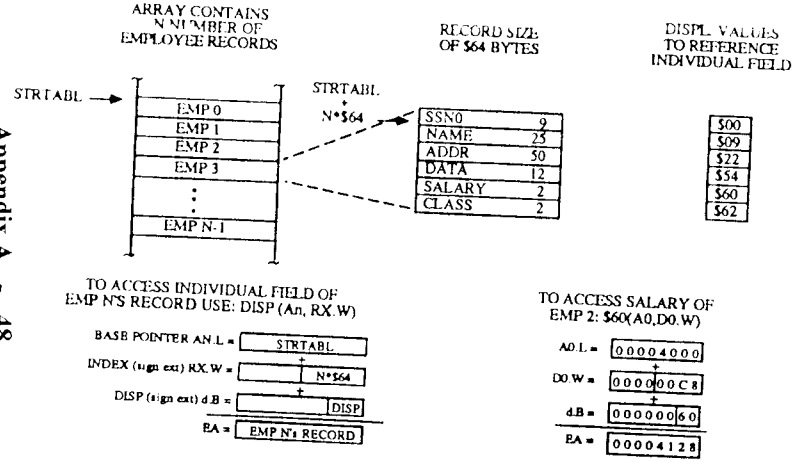
	A2.L = 0001 2000
(sign extended)	D1.W = 0000 0100
(sign extended)	d.B = 0000 0004
	EA = 0001 2104

- The Index Register (Rx) can be a word or a long word (2's Complement Notation).
- The displacement is always a byte (2's Complement Notation).

MTB-039

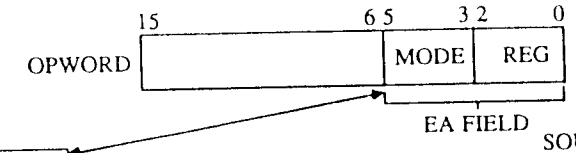
MC68000 – USING ARI with INDEX and DISPLACEMENT

ACCESSING DATA WITHIN A MULTIPLE RECORD ARRAY



MTB-040-2

EFFECTIVE ADDRESS



EA MODE	REG	ADDRESSING MODE	SOURCE SYNTAX
1 000	REG#	DATA REGISTER DIRECT	Dn
2 001	REG#	ADDRESS REGISTER DIRECT	An
3 010	REG#	ADDRESS REGISTER INDIRECT (ARI)	(An)
4 011	REG#	ARI WITH POSTINCREMENT	(An) +
5 100	REG#	ARI WITH PREDECREMENT	-(An)
6 101	REG#	ARI WITH DISPLACEMENT	d(An)
7 110	REG#	ARI WITH INDEX	d(An, Rx)
8 111	000	ABSOLUTE SHORT	$XXXX
9 111	001	ABSOLUTE LONG	$XXXXXXXX
12 111	100	IMMEDIATE	#XXXX

MTB-058-2

DATA MOVEMENT INSTRUCTIONS

INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
MOVE	8,16,32	(SOURCE) → DESTINATION	MOVE.s <ea>,<ea>	SOURCE-ALL DEST - DATA ALTERABLE
MOVEA	16,32	(SOURCE) → DESTINATION	MOVEA.s <ea>, An	ALL
MOVE from SR	16	SR → DESTINATION	MOVE.W SR, <ea>	DATA ALTERABLE
MOVE to CCR	16	(SOURCE) → CCR	MOVE.W <ea>, CCR	DATA
*MOVE to SR	16	(SOURCE) → SR	MOVE.W <ea>, SR	DATA
*MOVE USP	32	USP → An or An → USP	MOVE.L USP, An MOVE.L An , USP	—
MOVEQ	32	IMMEDIATE DATA → DESTINATION	MOVEQ.L # data, Dn	—
EXG	32	Rx ↔ Ry	EXG.L Rx,Ry	—
SWAP	16	Dnlw ↔ Dnhw	SWAP Dn	—

*PRIVILEGED

s = size of data
B=8 bits
W= 16 bits
L= 32 bits

MT8-104-3

MC68000 – EFFECTIVE ADDRESSING MODE CATEGORIES

EFFECTIVE ADDRESS MODES	DATA	MEMORY	CONTROL	ALTERABLE
Dn	X			X
An				X
(An)	X	X	X	X
(An)+	X	X		X
-(An)	X	X		X
d_{16} (An)	X	X	X	X
d_8 (An,Rx)	X	X	X	X
XXXX	X	X	X	X
XXXXXX	X	X	X	X
#XXXX	X	X		

MT8-105-1

INTEGER ARITHMETIC INSTRUCTIONS

INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
ADD	8,16,32	(DESTINATION) + (SOURCE) → DESTINATION	ADD.s Dn,<ea>	ALTERABLE MEMORY
			ADD.s <ea>, Dn	ALL
ADDA	16,32	(DESTINATION) + (SOURCE) → DESTINATION	ADDA.s <ea>, An	ALL
ADDI	8,16,32	(DESTINATION) + IMMEDIATE DATA → DESTINATION	ADDI.s #<data>,<ea>	DATA ALTERABLE
ADDQ	8,16,32	(DESTINATION) + IMMEDIATE DATA → DESTINATION	ADDQ.s #<data>,<ea>	ALTERABLE

s= size of data

B = 8 bits

W = 16 bits

L = 32 bits

MTB-106-2

INTEGER ARITHMETIC INSTRUCTIONS

INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
CMP	8,16,32	(OPERAND2) - (OPERAND1)	CMP.s <ea>, Dn	ALL
CMPA	16,32	(OPERAND2) - (OPERAND1)	CMPA.s <ea>, An	ALL
CMPI	8,16,32	(OPERAND) - IMMEDIATE DATA	CMPI.s #<data>,<ea>	DATA ALTERABLE
CMPM	8,16,32	(OPERAND2) - (OPERAND1)	CMPM.s (Ay) +,(Ax) +	—
TST	8,16,32	(DESTINATION) - 0 (DESTINATION) TESTED → CC	TST.s <ea>	DATA ALTERABLE

s = size of data

B = 8 bits

W = 16 bits

L = 32 bits

MTB-108-1

INTEGER ARITHMETIC INSTRUCTIONS

INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
SUB	8,16,32	(DESTINATION) - (SOURCE) → DESTINATION	SUB.s Dn,<ea>	ALTERABLE MEMORY
			SUB.s <ea>, Dn	ALL
SUBA	16,32	(DESTINATION) - (SOURCE) → DESTINATION	SUBA.s <ea>, An	ALL
SUBI	8,16,32	(DESTINATION) - IMMEDIATE DATA → DESTINATION	SUBI.s#<data>,<ea>	DATA ALTERABLE
SUBQ	8,16,32	(DESTINATION) - IMMEDIATE DATA → DESTINATION	SUBQ.s #<data>,<ea>	ALTERABLE

s = size of data

B = 8 bits

W = 16 bits

L = 32 bits

MTB-107-1

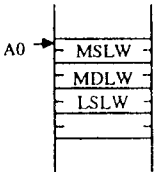
STRING COMPARISONS

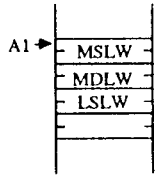
LP CMPM.L (A0)+,(A1)+

Done?

no - IF source = destination
THEN go to LP
ELSE return

yes - return





MSLW - MSLW MDLW - MDLW LSLW - LSLW

MSLW - Most Significant LongWord

MDLW - Middle LongWord

LSLW - Least Significant LongWord

MTB-210

INTEGER ARITHMETIC INSTRUCTIONS

INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
EXT	16,32	(DESTINATION) Sign-extended → DESTINATION	EXT.s Dn	—
MULS	16	(SOURCE) * (DESTINATION) → DESTINATION	MULS.W <ea>, Dn	DATA
MULU	16	(SOURCE) * (DESTINATION) → DESTINATION	MULU.W <ea>, Dn	DATA
NEG	8,16,32	0 – (DESTINATION) → DESTINATION	NEG.s <ea>	DATA ALTERABLE
CLR	8,16,32	0 → DESTINATION	CLR.s <ea>	DATA ALTERABLE
DIVS	16	(DESTINATION) ÷ (SOURCE) DESTINATION	DIVS.W <ea>, Dn	DATA
DIVU	16	(DESTINATION) ÷ (SOURCE) DESTINATION	DIVU.W <ea>, Dn	DATA

s = size (B, W, or L)

MT8-110-1

SHIFT AND ROTATE INSTRUCTIONS

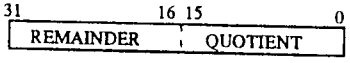
INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
ASL, ASR	8,16,32	ASL: C, X ← OPERAND ← 0	ASd.s Dx, Dy	—
			ASd.s #<data>,Dy	—
	16	ASR: OPERAND → C, X	ASd.W <ea>	MEMORY ALTERABLE
LSL, LSR	8,16,32	LSL: C, X ← OPERAND ← 0	LSd.s Dx, Dy	—
			LSd.s #<data>,Dy	—
	16	LSR: 0 → OPERAND → C, X	LSd.W <ea>	MEMORY ALTERABLE
ROL, ROR	8,16,32	ROL: C ← OPERAND	ROd.s Dx, Dy	—
			ROd.s #<data>,Dy	—
	16	ROR: OPERAND → C	ROd.W <ea>	MEMORY ALTERABLE

d = direction (L or R) s = size (B, W, or L)

MT8-114-2

MC68000 – DIVIDE INSTRUCTION DIVU, DIVS

- DIVIDEND IS A 32 BIT DATA REGISTER
- DIVISOR IS A 16 BIT EFFECTIVE ADDRESS (DATA)
- THE RESULT IS IN THE SPECIFIED DATA REGISTER



- IF THE DIVISOR IS ZERO, AN EXCEPTION OCCURS
- IF OVERFLOW IS DETECTED,

V → 1

AND OPERANDS ARE UNAFFECTED

MT8-049

MC68000 – MULTIPLE SHIFTS

*MULTIPLE SHIFT IN MEMORY

```
ROL.W   (A0)   13 (5 CLOCK CYCLE WRITE)
ROL.W   (A0)   13
  •
  •
  •
  •
```

N TIMES
TOTAL # CLOCK CYCLES = 13N

*BRING MEMORY TO Dn FOR A MULTIPLE SHIFT

```
MOVE.W   (A0), D1     8
ROL.W    #N,D1      6+2N
MOVE.W   D1, (A0)     9
```

TOTAL # CLOCK CYCLES = 23 + 2N

$23 + 2N < 13N$

$23 < 11N$

$2.1 < N$

*REFER TO APPENDIX D IN USER'S MANUAL

MT8-115

LOGIC INSTRUCTIONS

INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
AND	8,16,32	(SOURCE) ∧ (DESTINATION) → DESTINATION	AND.s Dn,<ea>	ALTERABLE MEMORY
			AND.s <ea>, Dn	DATA
ANDI	8,16,32	IMMEDIATE DATA ∧ (DESTINATION) → DESTINATION	ANDI.s #<data>,<ea>	DATA ALTERABLE OR CCR OR *STATUS REG.
EOR	8,16,32	(SOURCE) ⊕ (DESTINATION) → DESTINATION	EOR.s Dn,<ea>	DATA ALTERABLE
EORI	8,16,32	IMMEDIATE DATA ⊕ (DESTINATION) → DESTINATION	EORI.s #<data>,<ea>	DATA ALTERABLE OR CCR OR *STATUS REG.
NOT	8,16,32	(DESTINATION) → DESTINATION	NOT.s <ea>	DATA ALTERABLE
OR	8,16,32	(SOURCE) V (DESTINATION) → DESTINATION	OR.s Dn, <ea>	ALTERABLE MEMORY
			OR.s <ea>, Dn	DATA
ORI	8,16,32	IMMEDIATE DATA V (DESTINATION) → DESTINATION	ORI.s #<data>,<ea>	DATA ALTERABLE OR CCR OR *STATUS REG.

*PRIVILEGED – REF. USERS MANUAL

s = size (B, W, or L)

MT8-116

MC68000 – STOP INSTRUCTION

Example:

•
•
STOP #$2500
next instruction
•
•

SR_{before} ☐

SR_{after} ☐

Note: output signals same as HALTed state

Ways to exit STOP	Where to?
• Interrupt > MASK after	
• RESET	
• T=1 in SR before	
• S=0 in SR before	
• S=0 in SR after	

MT8-523

PROGRAM CONTROL INSTRUCTION (1 of 2)

INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
Bcc	8,16	If cc then PC + d → PC else proceed	Bcc <label>	PC REL
BRA	8,16	PC + d → PC	BRA <label>	PC REL
BSR	8,16	PC → -(SP);PC + d → PC	BSR <label>	PC REL
JMP	—	DESTINATION → PC	JMP <ea>	CONTROL
JSR	—	PC → -(SP); DESTINATION → PC	JSR <ea>	CONTROL
NOP	—	PC + 2 → PC	NOP	—
*RESET	—	RESET EXTERNAL DEVICES	RESET	—
*RTE	—	(SP) + → SR;(SP) + → PC	RTE	—
RTR	—	(SP) + → CC; (SP) + → PC	RTR	—
RTS	—	(SP) + → PC	RTS	—
*STOP	16	IMMEDIATE DATA → SR; STOP PROGRAM EXECUTION	STOP #<data>	—

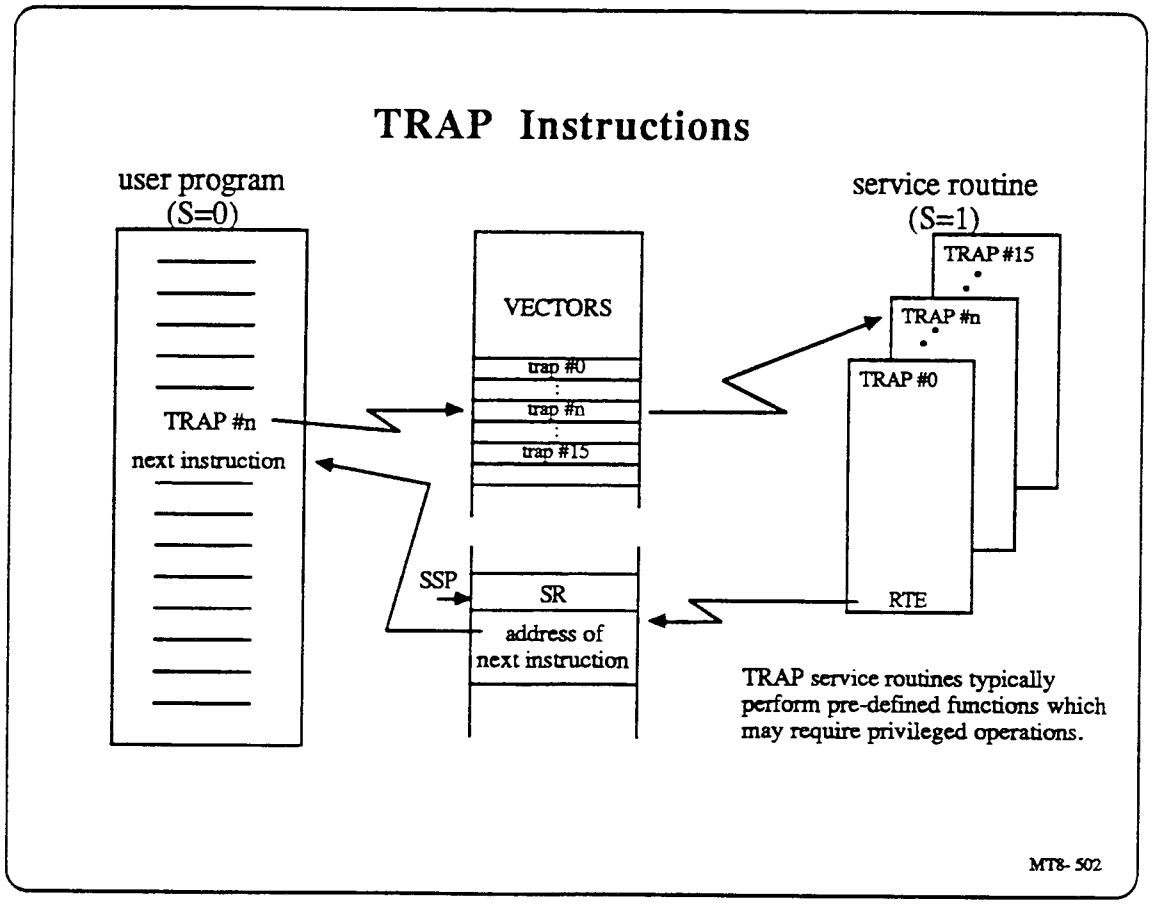
*PRIVILEGED

MT8-118-1

PROGRAM CONTROL INSTRUCTIONS (2 of 2)

INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION
TRAP	—	PC → -(SSP); SR → -(SSP); (VECTOR) → PC	TRAP #<data>
TRAPV	—	If V=1 then TRAP: else proceed	TRAPV

MT8-119


TRAP Instructions
user program
(S=0)
service routine
(S=1)
TRAP #15
VECTORS
TRAP #n
trap #0
TRAP #0
TRAP #n
trap #n
next instruction
trap #15
SSP
SR
RTE
address of
next instruction
TRAP service routines typically
perform pre-defined functions which
may require privileged operations.
MT8-502

POSITION INDEPENDENT CONCEPT

- Position Independent programs should execute from any load address.
- This is a relocatable program at the machine code level.
- To make a program Position Independent, the MC68000 instruction set must be able to use the Program Counter as an effective address register.
- The program counter relative addressing mode meets this requirement.

MT8-128

POSITION INDEPENDENT CODE

WHY?:

Operating systems that have no address translation hardware (MMU) must select program load addresses based on available memory at run time.

The programmer cannot select or determine what RAM area will be used.

A customer that has purchased a software product should have complete freedom to choose where that program will reside in a computer system.

HOW?:

IF: A PROGRAM is defined as a contiguous set of addresses containing code and data that accomplishes a unit of work,

THEN: Any addresses used by the PROGRAM to access code or data within the PROGRAM, must be generated with a program-counter-relative addressing mode.

ALSO: Any addresses used by the PROGRAM to access devices with fixed addresses in memory, must be generated without using a program-counter-relative addressing mode.

MT8-526

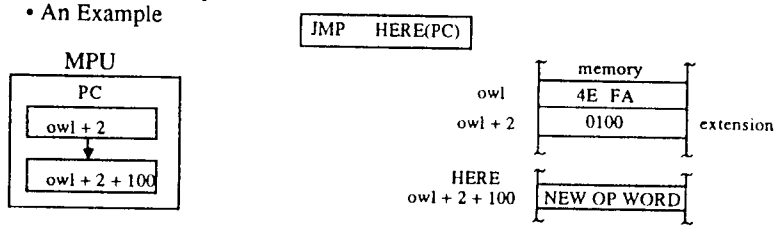
MC68000 – MULTIPLE TABLE LOOKUP PROGRAM EXAMPLE

```
*  MULTIPLE TABLE LOOKUP (NOT POSITION INDEPENDENT)
*
*USING LOOKUP TABLES, THE ASCII, DECIMAL,
*AND GRAY CODE VALUES OF HEX DIGITS
*IN D0 ARE STORED AT LOCATION TO WHICH
*A1 IS POINTING. A0 POINTS TO THE SET
*OF LOOKUP TABLES.
*
00001000                          ORG      $1000
00001000  307C1012                MOVE.W #ASCTABLE,A0     *A0 POINTS TO ASC
00001004  12F00000                MOVE.B 0(A0,D0),(A1) +  *ASCII VALUE GOES
00001008  12F00010                MOVE.B $10(A0,D0),(A1) + *BCD VALUE GOES
0000100C  12F00020                MOVE.B $20(A0,D0),(A1) + *GRAY VALUE GOES
00001010  4E75                    RTS
00001012  303132333435  ASCTABLE  DC.B   $30,$31,$32,$33,$34,$35,$36,$37,$38,$39
0000101C  414243444546            DC.B   $41,$42,$43,$44,$45,$46
00001022  000102030405  BCDTABLE  DC.B   0,1,2,3,4,5,6,7,8,9,$10,$11,$12,$13,$14,$15
00001032  000103020607  GRCTABLE  DC.B   0,1,3,2,6,7,5,4,$C,$D,$F,$E,$A,$B,9,8
                                  END
```

MT8-135-2

PROGRAM COUNTER WITH DISPLACEMENT

- EA = (PC) + displacement
- An Example



- OP WORD DECODE

GIVEN	MODE	REG
0100 1110 11	111	010

EA

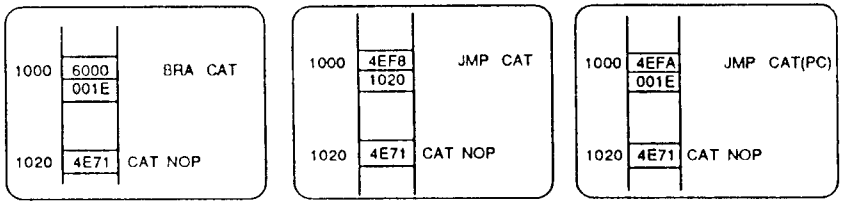
- DISPLACEMENT EXTENSION

0000	0001	0000	0000

- Displacement is always a word SIGN EXTENDED to a long word.

MT8-129-2

PROGRAM COUNTER RELATIVE VS ABSOLUTE CODE



MT8-679

PC WITH INDEX AND DISPLACEMENT

- EA Calculation

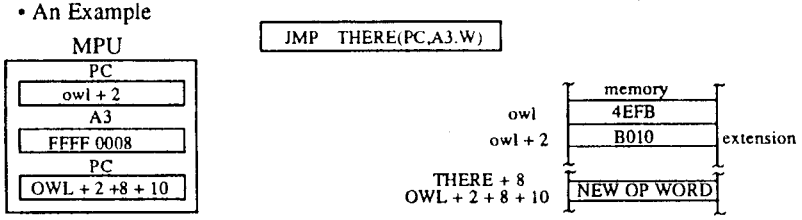
PC.L =	owl + 2
A3.W =	0000 0008
d.B =	0000 0010
EA =	owl + 2 + 8 + 10

- The Index Register (Rx) can be a word or a long word. It is in 2's complement notation.
- The displacement is always a byte. It is in 2's complement notation.

MT8-131

PROGRAM COUNTER WITH INDEX (and displacement)

- EA = (PC) + (Rx) + displacement
- An Example



- OP WORD DECODE

	EA	
GIVEN	MODE	REG
0100 1110 11	111	011

- EXTENSION DECODE

D/A	REG#	SIZE	GIVEN	←displacement→
1	011	0	000	0001 0000

MT8-130-2

MC68000 – EFFECTIVE ADDRESSING MODE CATEGORIES

EFFECTIVE ADDRESS MODES	DATA	MEMORY	CONTROL	ALTERABLE
Dn	X			X
An				X
(An)	X	X	X	X
(An) +	X	X		X
-(An)	X	X		X
d_{16}(An)	X	X	X	X
d_8 (An,Rx)	X	X	X	X
XXX.W	X	X	X	X
XXX.L	X	X	X	X
d_{16}(PC)	X	X	X	
d_8 (PC,Rx)	X	X	X	
#XXXX	X	X		

MT8-132-2

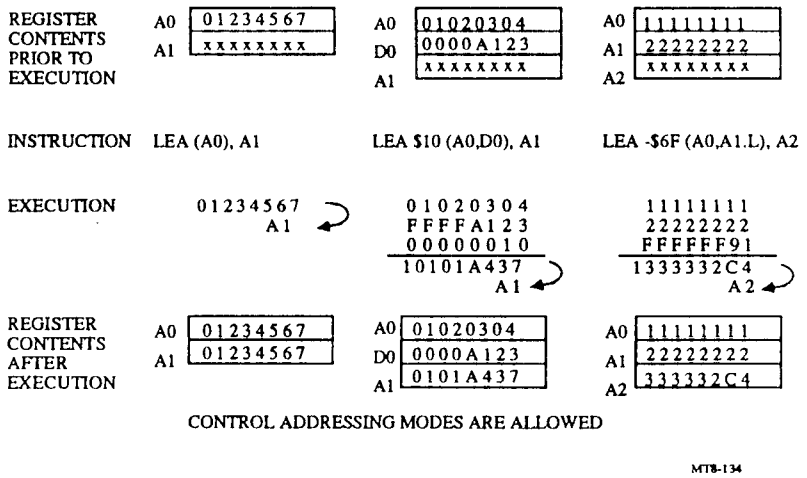
USEFUL INSTRUCTIONS FOR POSITION INDEPENDENT PROGRAMS

	Operand Size	Operation	Notation	Allowable Address Category
LEA	32	< ea > → An	LEA < ea >, An	Control

	Operand Size	Operation	Notation	Allowable Address Category
PEA	32	< ea > → -(SP)	PEA < ea >	Control

MT8-133

MC68000 – LEA INSTRUCTION



CONTROL ADDRESSING MODES ARE ALLOWED

MT8-134

MC68000 – MULTIPLE TABLE LOOKUP PROGRAM EXAMPLE

```
                                *    MULTIPLE TABLE LOOKUP
                                *
                                *USING LOOKUP TABLES, THE ASCII, DECIMAL,
                                *AND GRAY CODE VALUES OF A HEX DIGIT
                                *IN D0 ARE STORED IN LOCATIONS TO WHICH
                                *A1 IS POINTING. A0 POINTS TO THE SET
                                *OF LOOKUP TABLES.
                                *
            00001000                ORG $1000
00001000 41FA0010               LEA ASCTABLE (PC), A0     *A0 POINTS TO ASCTABLE
00001004 12F00000               MOVE.B 0(A0,D0), (A1)+    *ASCII VALUE GOES TO TEMP 1
00001008 12F00010               MOVE.B $10 (A0,D0),(A1)+  *BCD VALUE GOES TO TEMP 2
0000100C 12F00020               MOVE.B $20 (A0,D0),(A1)+  *GRAY VALUE GOES TO TEMP 3
00001010 4E75                   RTS
00001012 303132333435 ASCTABLE  DC.B   $30,$31,$32,$33,$34,$35,$36,$37,$3
0000101B 394142434445           DC.B   $39,$41,$42,$43,$44,$45,$46
00001022 000102030405 BCDTABLE  DC.B   0,1,2,3,4,5,6,7,8,9,$10,$11,$12,$13
00001030 1415                   DC.B   $14,$15
00001032 000103020607 GRCTABLE  DC.B   0,1,3,2,6,7,5,4,$C,$D,$F,$E,$A
0000103F 0B0908                 DC.B   $B,9,8
                                END
```

MT8 136-1

MC68000 – WRITING PC RELATIVE

You can't:

MOVE.B D1,TABLE(PC)

or

MOVE.B D1,TABLE(PC,D0.W)

But you can:

LEA TABLE(PC),A0
MOVE.B D1,(A0)

or

LEA TABLE(PC,D0.W),A0
MOVE.B D1,(A0)

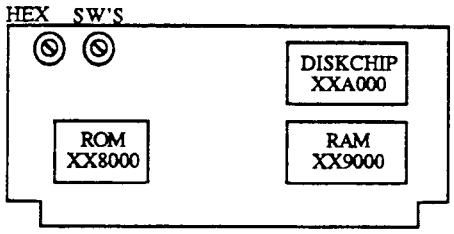
(A0) = DATA SPACE
TABLE(PC) = PROGRAM SPACE
TABLE(PC,D0.W) = PROGRAM SPACE

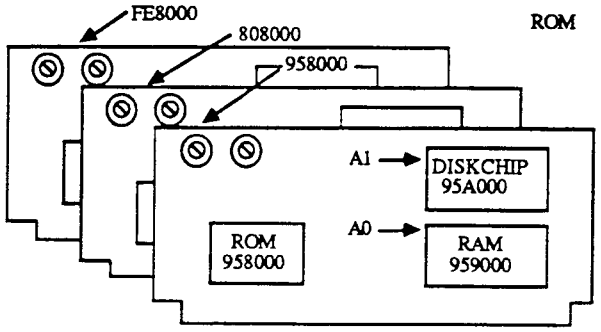
NOTE: program space

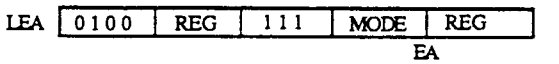
IF CS = ADDR • FC1 • $\overline{FC0}$
THEN NO WRITES ALLOWED

MT8-516-1

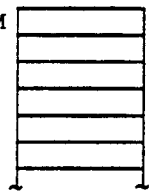
MC68000 USING LEA FOR POSITION INDEPENDENT SYSTEMS








```
                ORG   $9000
RAM             DS.B  $1000
                ORG   $A000
DISKCHIP        DS.B  $10
                ORG   $8000
ROM             LEA   RAM (PC), A0
                LEA   DISKCHIP (PC), A1
                 •
                 •
                 •
MOVE.B D4,(A1)
                 •
                 •
MOVE.B (A1),(A0)+
                 •
```



MT8-137-1

EXTENDED PRECISION INSTRUCTIONS

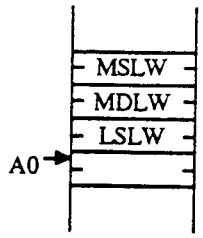
INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
ADDX	8,16,32	(DESTINATION) + (SOURCE) +X → DESTINATION	ADDX.s Dy, Dx ADDX.s -(Ay), -(Ax)	———
SUBX	8,16,32	(DESTINATION) - (SOURCE) -X → DESTINATION	SUBX.s Dy, Dx SUBX -(Ay), -(Ax)	———
NEGX	8,16,32	0 – (DESTINATION) – X → DESTINATION	NEGX.s <ea>	DATA ALTERABLE
ROXL ROXR	8,16,32	ROXL C X OPERAND	ROXd.s Dx, Dy — — — —	——— — — — —
	16	ROXR OPERAND C X	ROXd.s #<data>,Dy	———
			ROXd.W <ea>	MEMORY ALTERABLE

d = direction (L or R) s = size (B, W, or L)

MT8-703-1

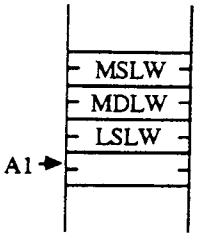
EXTENDED PRECISION ARITHMETIC

```
        Clear "X" bit in CCR
        Set "Z" bit in CCR
LP      ADDX.L -(A0),-(A1)
        Done?
          no - go to LP
          yes - return
```



X=0

X	X	0
MSLW	MDLW	LSLW
+ MSLW	+ MDLW	+ LSLW



MSLW - Most Significant LongWord
MDLW - Middle LongWord
LSLW - Least Significant LongWord

Should loop control instruction(s) affect X or Z?_____

MT8-208

BCD OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
ABCD	8	$(SOURCE)_{10} + (DESTINATION)_{10} + X \longrightarrow$ DESTINATION	ABCD.B Dy,Dx ABCD.B -(Ay), -(Ax)	——
NBCD	8	$0 - (DESTINATION)_{10} - X \longrightarrow$ DESTINATION	NBCD.B <ea>	DATA ALTERABLE
SBCD	8	$(DESTINATION)_{10} - (SOURCE)_{10} - X \longrightarrow$ DESTINATION	SBCD.B Dy, Dx SBCD.B -(Ay), -(Ax)	——

MT8-113-1

BIT MANIPULATION INSTRUCTIONS

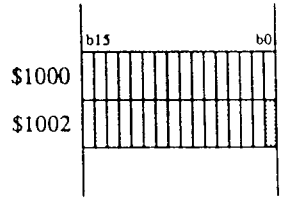
INSTRUCTION	*OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE EFFECTIVE ADDRESS MODES
BCHG	8,32	~ (bit number OF Destination) → Z ~ (bit number OF Destination) → bit number OF Destination	BCHG Dn, <ea> BCHG #<data>,<ea>	DATA ALTERABLE
BCLR	8,32	~ (bit number OF Destination) → Z 0 → bit number OF Destination	BCLR Dn, <ea> BCLR #<data>,<ea>	DATA ALTERABLE
BSET	8,32	~ (bit number OF Destination) → Z 1 → bit number OF Destination	BSET Dn, <ea> BSET #<data>,<ea>	DATA ALTERABLE
BTST	8,32	~ (bit number OF Destination) → Z	BTST Dn, <ea> BTST #<data>,<ea>	DATA (EXCLUDING IMMEDIATE)

*1. FOR MEMORY OPERATION, THE DATA SIZE IS BYTE.
2. FOR DATA REG. OPERATION, THE DATA SIZE IS LONG WORD.

MT8-117-1

MC68000 – BIT INSTRUCTION EXERCISE

Write a sequence of instructions that determine the value of the shaded bit. They should branch to label READY if the bit is clear.



MT8-505-1

SPECIAL INSTRUCTIONS

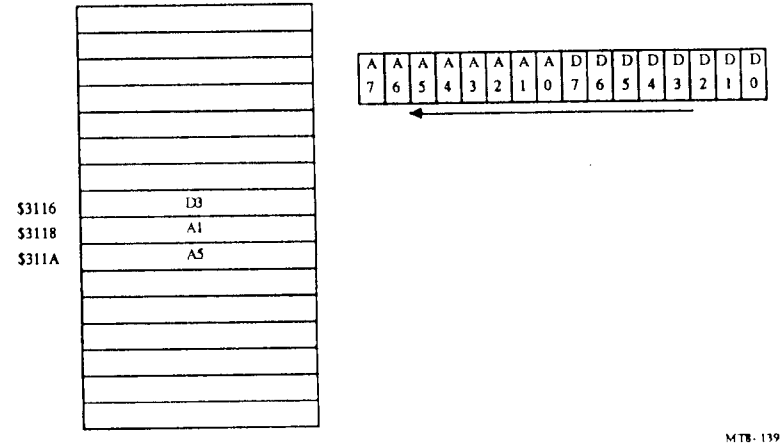
INSTRUCTION	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE ADD. CATEGORY
MOVEM	16,32	REGISTERS → DESTINATION (SOURCE) → REGISTERS	MOVEM REG LIST, <ea> MOVEM <ea>, REGISTER LIST	ALTERABLE CONTROL AND PREDECREMENT CONTROL AND POST INCREMENT
LINK	——	An → -(SP); SP → An; SP+d → SP	LINK An,# (disp.)	——
UNLK	——	An → SP; (SP) + → An	UNLK An	——
Scc	8	IF cc THEN 1's → DEST. ELSE 0's → DESTINATION	Scc <ea>	DATA ALTERABLE
* DBcc	16	IF cc THEN Dn-1 → Dn; IF Dn ≠ -1 THEN PC+d → PC	DBcc Dn.W, LABEL	——
CHK	16	IF Dn IS LESS THAN 0 OR GREATER THAN <ea> THEN EXCEPTION	CHK <ea>, Dn.W	DATA
MOVEP	16,32	(SOURCE) → DESTINATION	MOVEP Dx, d(Ay) MOVEP d(Ay), Dx	——

*DBF = DBRA

MT8-138-1

MC68000 – MOVEM (REGISTERS TO MEMORY)

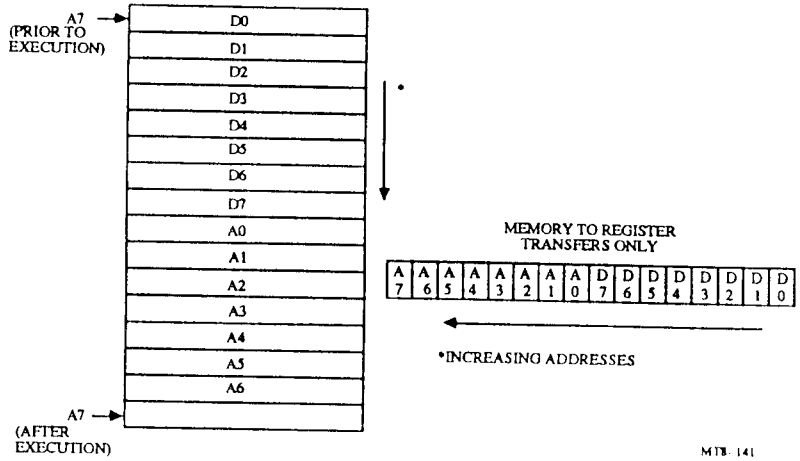
48B822083116 MOVEM.W A1/A5/D3,$3116



MTB-139

MC68000 – MOVEM (POST INCREMENT)

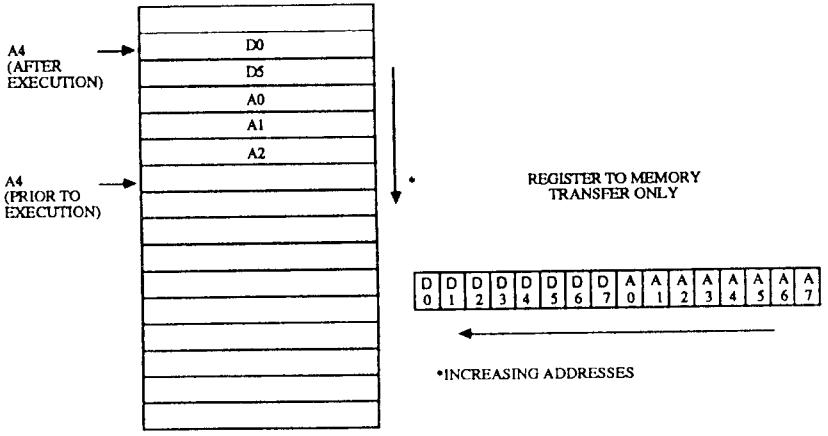
4C9F7FFF MOVEM.W (A7) + ,A0 – A6/D0-D7



MTB-141

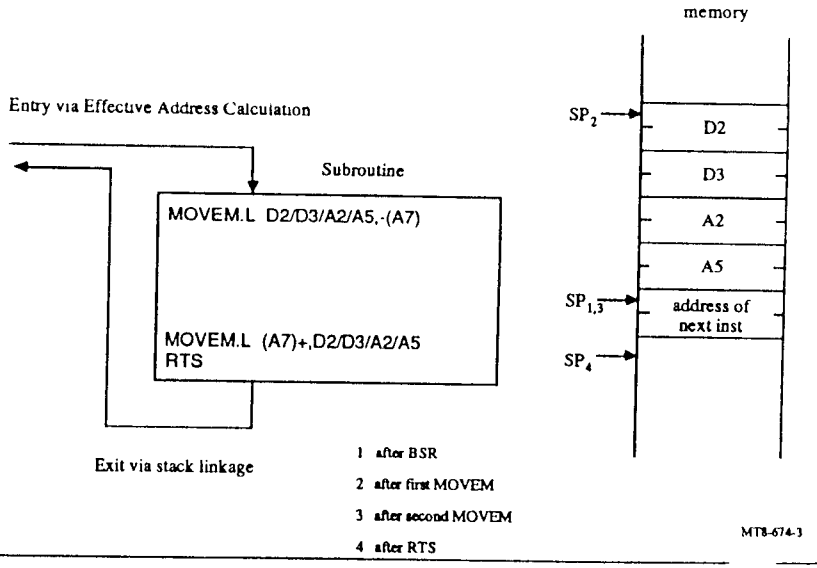
MC68000 – MOVEM (PREDECREMENT)

48A484E0 MOVEM.W D0/D5/A0 – A2, – (A4)



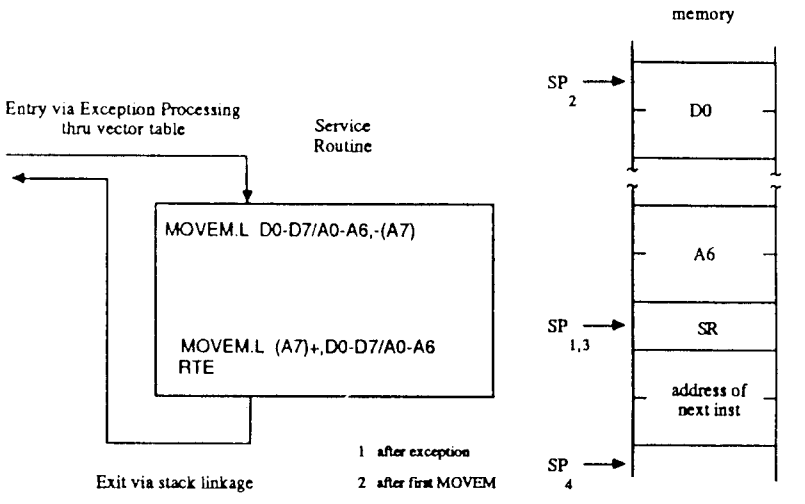
MTB-140

SUBROUTINE ENTRY & EXIT



MTB-674-3

EXCEPTION SERVICE ROUTINE ENTRY & EXIT



MT8-675-1

PARAMETER PASSING / LOCAL WORK SPACE

location	description	advantages	disadvantages
registers	calling routine loads registers/ subroutine uses registers	fast access dynamic reentrant	MPU registers used up limited number
stack	calling routine pushes parameters onto stack/ subroutine allocates storage on stack	MPU registers not used up dynamic reentrant	slower access
mailbox	pre-defined memory area(s)	MPU registers not used up dynamic	slower access non-reentrant
*in line	parameters in instruction stream following the call. subroutine must compute location of parameters	RAM/MPU register resources not used position independent	slower access static return address must be adjusted

* parameter passing only

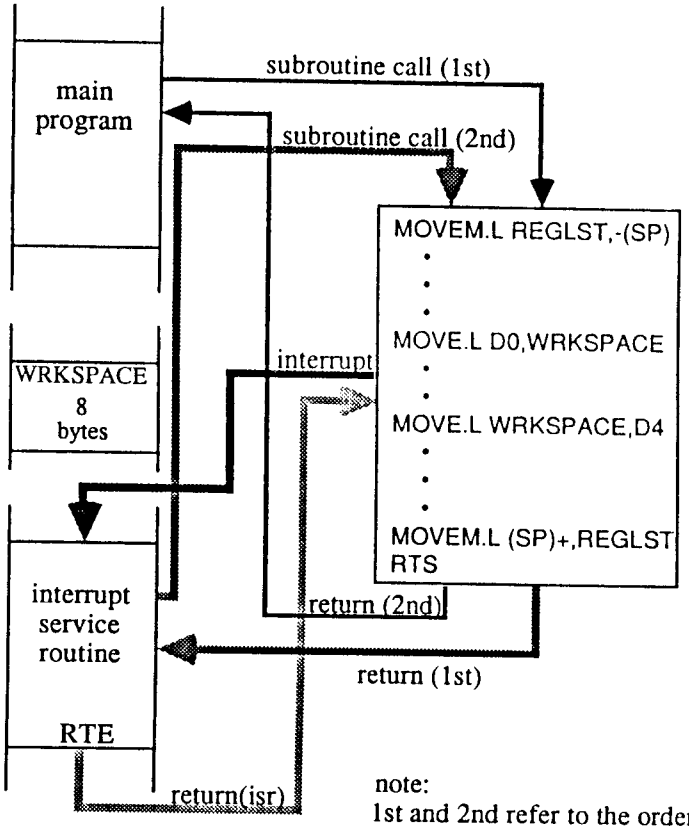
. Parameters may be passed by value or by address.

MT8-515-2

MC68000 – THE PROBLEM OF REENTRANCY

Reentrant - a program which may be exited by means of an interrupt or subroutine call and be re-entered and operate properly.

Example: not reentrant (using fixed address)



note:
1st and 2nd refer to the order in which events occur

MT8-507-3

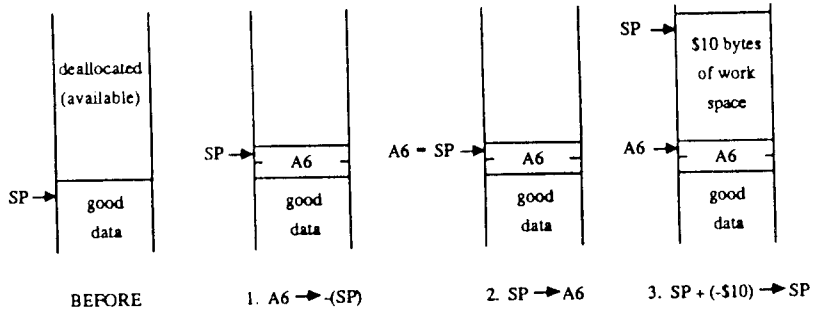
MC68000 – LINK INSTRUCTION

Example:

LINK A6,#-$10

Operation:

1. A6 → -(SP)
2. SP → A6
3. SP + (-$10) → SP



MT8-513-3

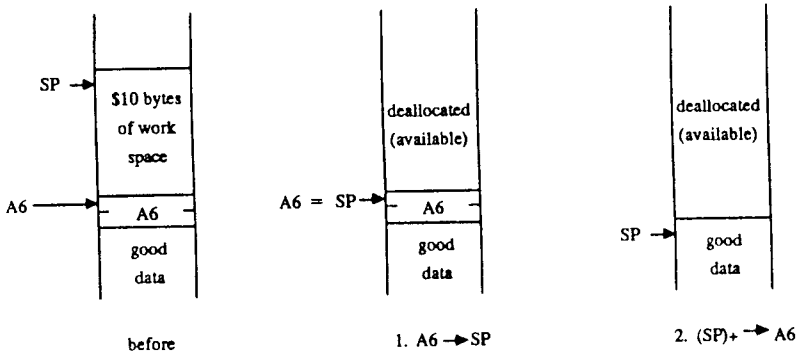
MC68000 – UNLK INSTRUCTION

Example:

UNLK A6

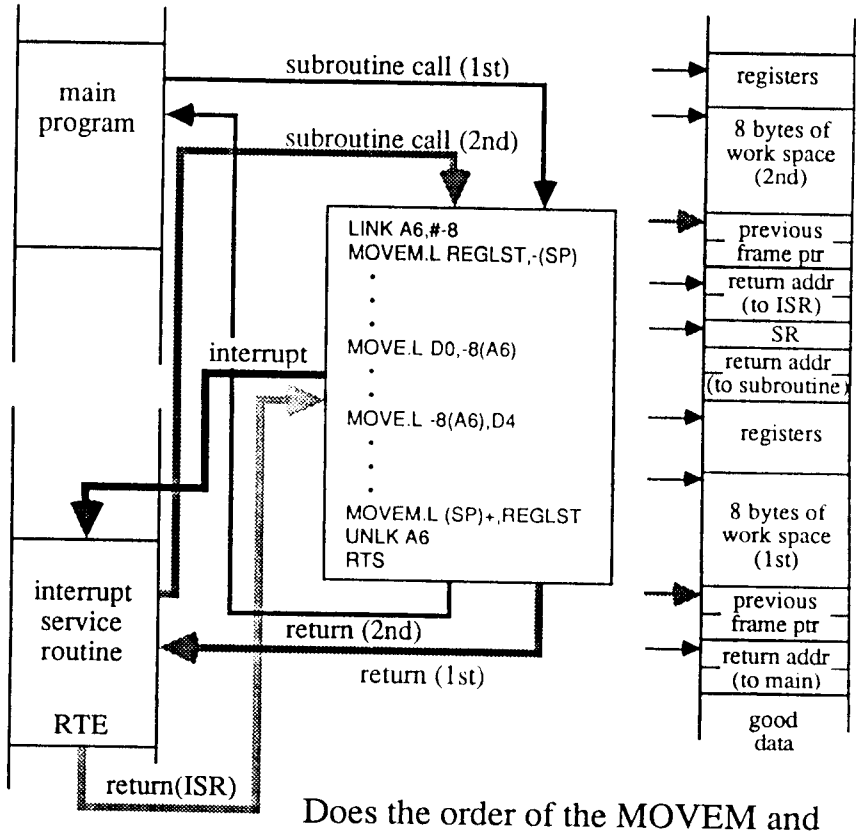
Operation:

1. A6 → SP
2. (SP)+ → A6



MT8-512-2

MC68000 – REENTRANT PROGRAM

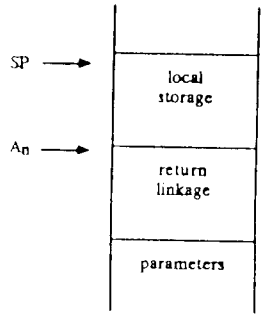


note: 1st and 2nd refer to the order in which events occur.

Does the order of the MOVEM and LINK instructions in the subroutine make any difference?

MT8-508-2

SUBROUTINE STACK USAGE



MOVEM reg list,-(A7)
LINK An,#-size

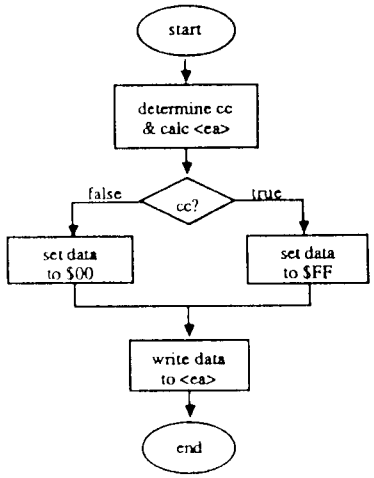
JSR
or
BSR

PEA
or
MOVE <ea>,-(A7)

MT8-676-1

Scc

Scc	8	IF cc THEN 1's → DESTINATION ELSE 0's → DESTINATION	Scc <ea>	DATA ALTERABLE



MT8-672

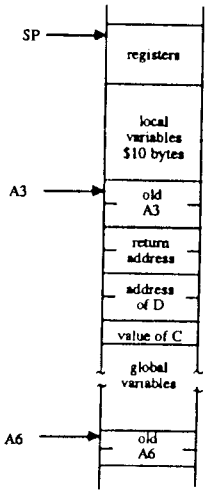
MC68000 – USE OF FRAME POINTER LOCAL AND GLOBAL VARIABLES

Main

```
*Allocate global variables
   LINK  A6 #-$20
      .
      .
      .
*Prepare for call
*VARC is passed by value
*VARD has its address
*    passed
   MOVE.W  #VARC,-(SP)
   PEA  VARD
   JSR  ROUTINE
      .
      .
```

Subroutine

```
ROUTINE:
*Allocate local variables
   LINK  A3,#-$10
*Preserve callers registers
   MOVEM.L  regist,-(SP)
      .
      .
*Access local variable
   MOVE.B  -2(A3),D0
      .
      .
*Access global variable
   MOVE.W  -8(A6),D4
      .
      .
*Access value of VARC
   MOVE.W  12(A3),D0
*Access variable VARD
   MOVEA.L  8(A3),A0
   MOVE.B  D0,(A0)
      .
      .
   MOVEM.L  (SP)+,regist
   UNLK  A3
   RTS
```



MT8-517-2

USE OF Scc

Compiler Statement:

```
IF A>3 AND A≥B
   THEN
      .
      .
   ELSE
      .
      .
```

Assembler Code:

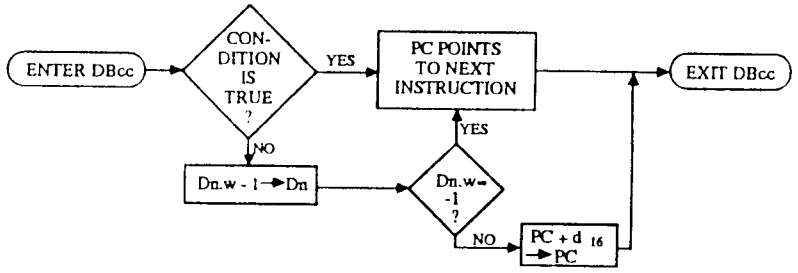
```
      MOVE.W   <ea of A>,D0
      CMP.W    #3,D0
      SGT      D1
      CMP.W    <ea of B>,D0
      SGE      D2
      AND.B    D1,D2
      BEQ      ELSE
THEN
  .
  .
ELSE
  .
  .
```

Note: A and B are assumed to be 16-bit signed integers

MT8-673-1

MC68000 – DBcc INSTRUCTION

DBcc	16	IF $\overline{cc}$ THEN Dn-1 → Dn; IF Dn ≠ -1 THEN PC+d → PC	DBcc Dn.W, LABEL	—



* DBF=DBRA

MT8-143-1

MC68000 – DBcc EXAMPLES

```
AGAIN  .
       .
       .
       .
       .
       .
       .
DBEQ D0,AGAIN
NEXT   .
       .
       .
```

	D0.W	SR	D0.W	SR	D0.W	SR	
REGISTER CONTENTS PRIOR TO EXECUTION	0001	A600	0000	A604	0000	A600	
RESULTS	D0.W=0; PC=AGAIN		D0.W=0000; PC=NEXT		D0.W=$FFFF; PC=NEXT		

MT8-144

MC68000 – LOOPING CONSTRUCTS (DBcc)

Conditional Loops

```
REPEAT
  -
  -   } test
  -
  COUNT = COUNT-1
UNTIL TEST=0 OR COUNT=-1
```

e.g. DBEQ

```
* INITIALIZE COUNTER
LOOP  -
      -
      -
      DBEQ Dn,LOOP
```

Counted Loops

```
FOR X = 1 TO N DO
  -
  -
  -
  -
ENDF
```

e.g. DBF

```
      MOVEQ.L #N-1,Dn
LOOP  -
      -
      -
      -
      DBF Dn,LOOP
```

MT8-520-2

MC68000 – STRING COMPARE EXAMPLE

```
*  STRING COMPARE
*
*TWO STRINGS ARE COMPARED FOR
*SAMENESS. A0 POINTS TO THE BEGIN-
*NING OF THE FIRST STRING AND
*A1, TO THE BEGINNING OF THE SECOND.
*THE NUMBER OF LONG WORDS TO BE COMPARED
*IS IN D1.
*
00002000 003C0004 ENTRY ORI     #$04,CCR    *FORCE Z=1
00002004 6002           BRA.S   SKIP        *IS NUMBER IN D1=0?
00002006 B388     AGAIN CMPM.L  (A0)+,(A1)+ *COMPARE STR1 WITH STR2
00002008 56C9FFFC SKIP  DBNE    D1,AGAIN    *WHILE Z=1 & D1 ≠ -1 DO AGAIN
0000200C 4E71           NOP                 *NEXT INSTRUCTION
```

MT8-145-3

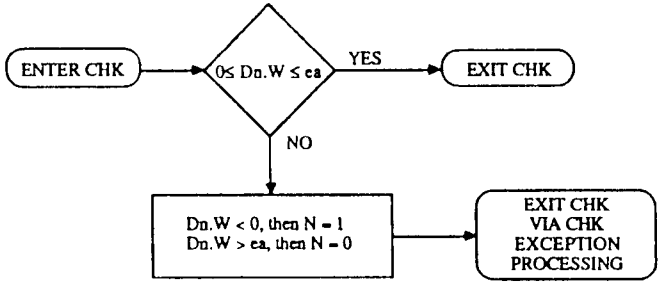
MC68000 – PALINDROME CHECK

```
*     DETECTING A 16-BIT PALINDROME
*
* A 16-BIT WORD POINTED TO BY A0
* IS CHECKED FOR SYMMETRY,I.E.,
* THE WORD READS THE SAME LEFT
* TO RIGHT AS IT READS RIGHT TO
* LEFT. IF IT IS DETERMINED TO BE
* A PALINDROME,THE BYTE TO WHICH
* THE STACK POINTER IS POINTING
* AFTER RTS WILL BE ALL 1'S.
*
00001000  48E7E000
00001004  7407      PALCHK MOVEM.L  D0/D1/D2,-(SP)
00001006  3010             MOVEQ.L  #7,D2          * 7 --> COUNTER
00001008  E248             MOVE.W   (A0),D0        *TEST WORD -->  D0.W
0000100A  E351      AGAIN  LSR.W    #1,D0          *LSB,D0.W --> X,CCR
0000100C  51CAFFFA         ROXL.W   #1,D1          *X,CCR --> LSB,D1.W
00001010  B200             DBF      D2,AGAIN       *DO THIS LOOP 8 TIMES
00001012  57EF0010         CMP.B    D0,D1          *TST;WRD=MIRROR IMAGE OF LSB
00001016  4CDF0007         SEQ      16(SP)         *THN $FF --> 16(SP); ELS  00 --> 16(SP)
0000101A  4E75             MOVEM.L  (SP) +,D0/D1/D2
                           RTS
```

MTB-146-2

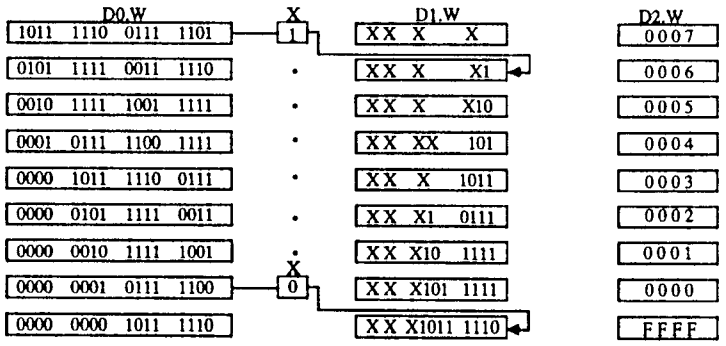
MC68000 – CHK INSTRUCTION

CHK	16	IF Dn IS LESS THAN 0 OR GREATER THAN <ea> THEN EXCEPTION	CHK <ea>, Dn	DATA



MTB-148

MC68000 – STEPS IN THE PALINDROME CHECK



MTB-147

CHK INSTRUCTION EXAMPLE

source code

```
DIMENSION ARRAY [LENGTH]
.
.
.
A = ARRAY [i]
```

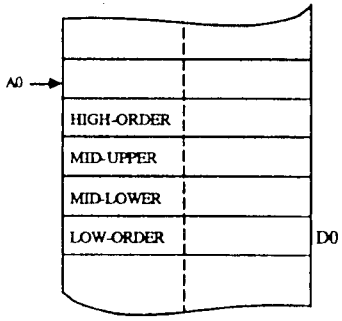
compiler output

```
MOVE.W   #i,D2
CHK      #LENGTH,D2
MOVE.B   0(A0,D2.W),D4
```

MTB-503-1

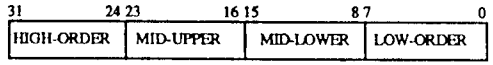
MC68000 – MOVEP INSTRUCTION

	OPERAND SIZE	OPERATION	NOTATION	ALLOWABLE ADDRESS CATEGORY
MOVEP	16,32	(SOURCE) → DESTINATION	MOVEP Dx,d(Ay) MOVEP d(Ay),Dx	—



MOVEP.L D0,2(A0)

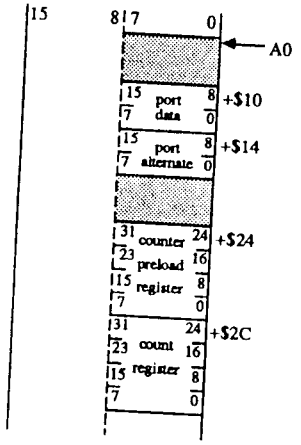
INCREASING MEMORY ADDRESSES

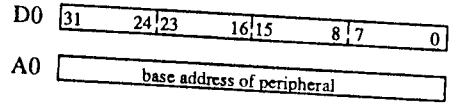


NOTE: only address register indirect with displacement addressing allowed

MT8-159-1

MC68000 – MOVEP EXAMPLE



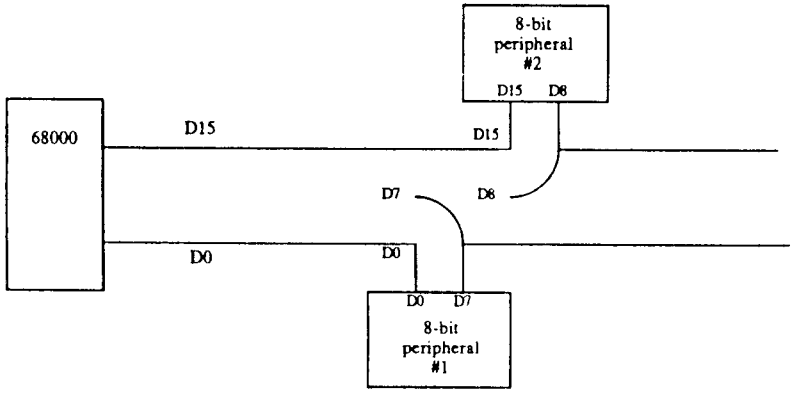


MOVEP.W D0,$10(A0)
MOVEP.W $14(A0),D0
MOVEP.L D0,$24(A0)

Write the instruction that would copy the 32-bit count register into data register 0.

MT8-525-2

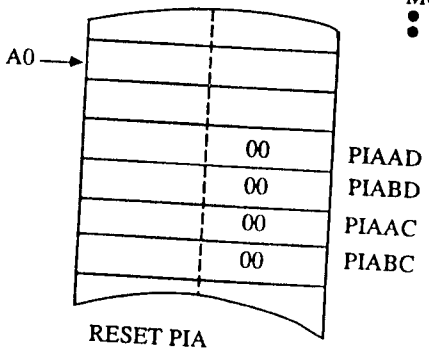
ATTACHING 8-BIT PERIPHERALS

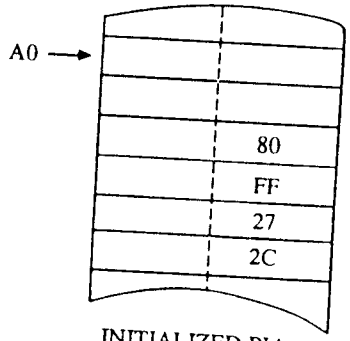


MT8-678-1

INITIALIZING PIA USING MOVEP EXAMPLE

```
•
•
RESET
INIT  LEA BIOADR,A0
•
      MOVE.L #$80FF272C,D0
      MOVEP.L D0,$5(A0)
•
•
```





MT8-165-2

EFFECTIVE ADDRESSING MODES AND CATEGORIES

EFFECTIVE ADDRESS MODES	MODE	REGISTER	ADDRESSING CATEGORIES			
			DATA	MEMORY	CONTROL	ALTERABLE
Dn	000	REGISTER NUMBER	X			X
An	001	REGISTER NUMBER				X
(An)	010	REGISTER NUMBER	X	X	X	X
(An) +	011	REGISTER NUMBER	X	X		X
-(An)	100	REGISTER NUMBER	X	X		X
d_{16} (An)	101	REGISTER NUMBER	X	X	X	X
d_8 (An,Rx)	110	REGISTER NUMBER	X	X	X	X
xxx.W	111	000	X	X	X	X
xxx.L	111	001	X	X	X	X
d_{16} (PC)	111	010	X	X	X	
d_8 (PC,Rx)	111	011	X	X	X	
#xxx	111	100	X	X		

MT8-188-1

MC68000 – EA MODE CATEGORY DESCRIPTION

DATA — If an effective address mode may be used to refer to data operands, it is considered a data addressing effective address mode.

MEMORY — If an effective address mode may be used to refer to memory operands, it is considered a memory addressing effective address mode.

ALTERABLE — If an effective address mode may be used to refer to alterable (writable) operands, it is considered an alterable addressing effective address mode.

CONTROL — If an effective address mode may be used to refer to memory operands without an associated size, it is considered a control addressing effective address mode.

MT8-189

MC68000 INSTRUCTION SET

DATA

CLR	MOVE
EXG	SWAP
EXT	

SHIFTS & ROTATES

ASL	LSR
ASR	ROL
LSL	ROR

CONTROL

BRA	RTE
BSR	RTR
JMP	RTS
JSR	TRAP
NOP	

ARITHMETIC

ADD	MUL
CMP	NEG
DIV	SUB
TST	

MULTIPLE PRECISION

ADDX	ROXR
NEGX	SUBX
ROXL	

CONDITIONAL

Bcc	Scc
DBcc	TRAPV

LOGICAL

AND	NOT
EOR	OR

BIT MANIPULATION

BCHG	BSET
BCLR	BTST

BINARY CODED DECIMAL

ABCD	SBCD
NBCD	

SPECIAL

ILLEGAL	LEA	LINK
CHK	PEA	UNLK
TAS	RESET	STOP
	MOVEP	MOVEM

MT8-527-1

ADDRESSING MODES THAT SIGN EXTEND THE ADDRESS OR THE DATA

1. ABSOLUTE SHORT
   WORD ADDRESS EXTENDED TO LONG WORD ADDRESS
2. ADDRESS REGISTER DIRECT (AS A DESTINATION)
   WORD DATA EXTENDED TO LONG WORD DATA
3. ADDRESS REGISTER INDIRECT WITH DISPLACEMENT
   WORD DISPLACEMENT EXTENDED TO LONG WORD DISPLACEMENT
4. ADDRESS REGISTER INDIRECT WITH INDEX
   A. WORD INDEX EXTENDED TO LONG WORD INDEX
   B. BYTE DISPLACEMENT EXTENDED TO LONG WORD DISPLACEMENT
5. PROGRAM COUNTER WITH DISPLACEMENT
   SAME AS 3 ABOVE
6. PROGRAM COUNTER WITH INDEX
   SAME AS 4 ABOVE

MT8-192

MC68000 – PRIVILEGE STATES

STATE	S BIT	OPERATIONAL RESTRICTIONS	
USER	0	PRIVILEGED INSTRUCTIONS WHICH ARE NOT ALLOWED:	
		INSTRUCTION	OPERATION
		RESET	RESET EXTERNAL DEVICES
		RTE	RETURN FROM EXCEPTION
		STOP	STOP PROGRAM EXECUTION
		ORI TO SR	LOGICAL OR TO STATUS REGISTER
		MOVE USP	MOVE USER STACK POINTER
		ANDI TO SR	LOGICAL AND TO STATUS REGISTER
		EORI TO SR	LOGICAL EOR TO STATUS REGISTER
		MOVE EA TO SR	LOAD NEW STATUS REGISTER
SUPERVISOR	1	NO RESTRICTIONS – ALL INSTRUCTIONS MAY BE EXECUTED	

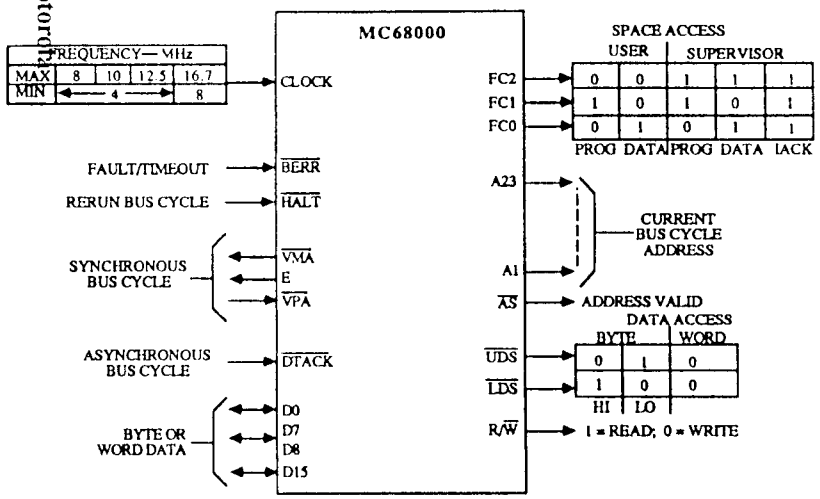
MT8-084-1

INSTRUCTIONS THAT SIGN EXTEND THE DATA

1. ADDA.W (ADDQ.W TO An ≡ ADDQ.L)
   SOURCE DATA SIGN EXTENDED TO LONG WORD
2. CMPA.W
   SOURCE DATA SIGN EXTENDED TO LONG WORD
3. EXT.W OR EXT.L
   SIGN EXTENDS BYTE TO WORD OR WORD TO LONG WORD
4. SUBA.W (SUBQ.W TO An ≡ SUBQ.L)
   SOURCE DATA SIGN EXTENDED TO LONG WORD
5. MOVEA.W
   DESTINATION IS AN ADDRESS REGISTER
6. MOVEM.W MEMORY TO REGISTERS
   SOURCE DATA SIGN EXTENDED TO LONG WORD
   DESTINATION IS ANY REGISTER
7. MOVEQ.L IMMEDIATE BYTE SIGN EXTENDED TO LONG WORD
   DESTINATION IS A DATA REGISTER

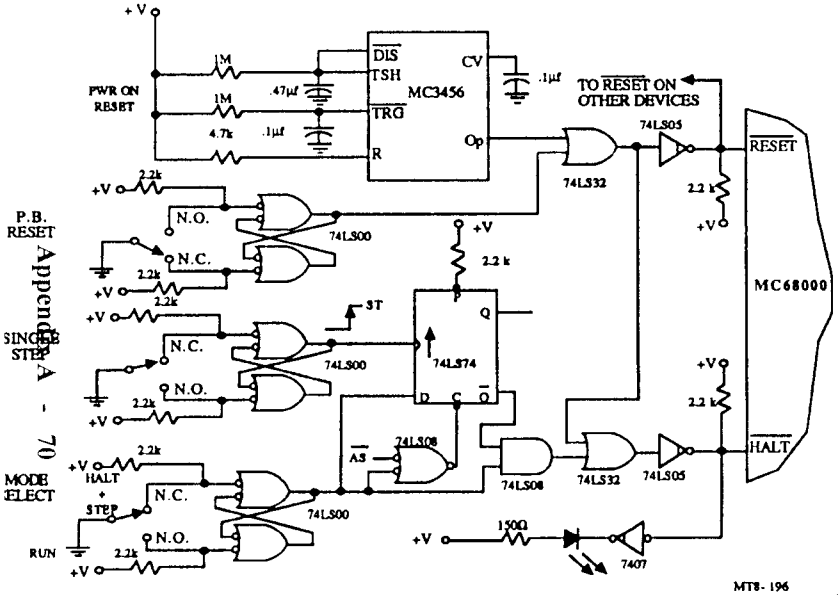
MT8-193

MC68000 – BUS CYCLE REVIEW



MT8-194-2

RESET, HALT, SINGLE STEP OPERATION



MT8-196

INSTRUCTION PREFETCH

DEFINITION: The execution of an instruction begins when the microroutine for that instruction is entered.

1. When the execution of an instruction begins, the operation word and the word following have already been fetched. The operation word is in the instruction decoder.

2. In the case of multiword instructions, as each additional word of the instruction is used internally, a fetch is made to the instruction stream to replace it.

3. The last fetch for an instruction from the instruction stream is made when the operation word is discarded and decoding is started on the next instruction.

4. If the instruction is a single-word instruction causing a branch, the second word is not used. But because this word is fetched by the preceding instruction, it is impossible to avoid this superfluous fetch.

5. In the case of an interrupt or trace exception, both words are not used.

6. The program counter usually points to the last word fetched from the instruction stream.

INSTRUCTION PREFETCH EXAMPLE:

```
        ORG  0      RESET VECTOR
        DC.L        INISSP
        DC.L        RESTART

        ORG         INTVECTOR
        DC.L        INTHANDLER

        ORG
RESTART:
        NOP
        BRA.S       LABEL
        ADD.W       D0,D1
LABEL:
        SUB.W       disp(A0),A1
        CMP.W       D2,D3
        SGE.B       D7
          :
          :
INTHANDLER:
        MOVE.W      LADR1,LADR2
        NOP
        SWAP.W      D4
```

NOTE: The order of operations described within each microroutine is not exact, but is intended for illustrative purposes only.

MICROROUTINE	<EA>	OPERAND
RESET	$0	SSP HIGH
	$2	SSP LOW
	$4	PC HIGH
	$6	PC LOW
	(PC)	NOP
	+(PC)	BRA.S
NOP	+(PC)	ADD
BRA.S	(PC+d)	SUB
	+(PC)	displ
<interrupt occurs>	+(PC)	CMP
SUB d(A0),A1	d(A0)	<src>
	+(PC)	SGE
INTERRUPT	-(SSP)	PC LOW
	IACK	VECTOR#
	-(SSP)	PC HIGH
	-(SSP)	SR
	(VR)	PC HIGH
	+(VR)	PC LOW
	(PC)	MOVE
	+(PC)	XXX HIGH
MOVE XXX.L,YYY.L	+(PC)	XXX LOW
	+(PC)	YYY HIGH
	XXX	<src>
	+(PC)	YYY LOW
	+(PC)	NOP
	YYY	<dest>
	+(PC)	SWAP
NOP	+(PC)	OPWORD

MT8-699-1

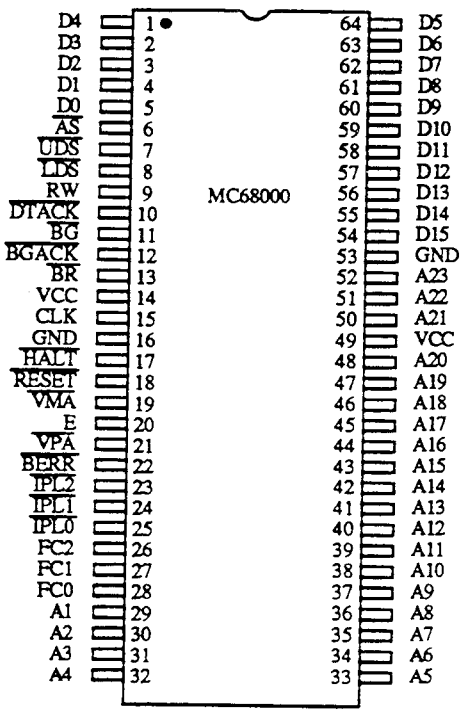
MC68000 – SIGNAL SUMMARY

SIGNAL NAME	MNEM.	INPUT/OUTPUT	ACTIVE STATE	HI - Z	
				ON $\overline{HALT}$	ON $\overline{BGACK}$
ADDRESS BUS	A1-A23	OUTPUT	HIGH	YES	YES
DATA BUS	D0-D15	INPUT/OUTPUT	HIGH	YES	YES
ADDRESS STROBE	$\overline{AS}$	OUTPUT	LOW	NO	YES
READ/WRITE	R/$\overline{W}$	OUTPUT	READ-HIGH WRITE-LOW	NO	YES
UPPER & LOWER DATA STROBES	$\overline{UDS}$,$\overline{LDS}$	OUTPUT	LOW	NO	YES
DATA TRANSFER ACKNOWLEDGE	$\overline{DTACK}$	INPUT	LOW	NO	NO
BUS REQUEST	$\overline{BR}$	INPUT	LOW	NO	NO
BUS GRANT	$\overline{BG}$	OUTPUT	LOW	NO	NO
BUS GRANT ACKNOWLEDGE	$\overline{BGACK}$	INPUT	LOW	NO	NO
INTERRUPT PRIORITY LEVEL	$\overline{IPL0}$,$\overline{IPL1}$ $\overline{IPL2}$	INPUT	LOW	NO	NO
BUS ERROR	$\overline{BERR}$	INPUT	LOW	NO	NO
RESET	$\overline{RESET}$	INPUT/OUTPUT	LOW	NO[1]	NO[1]
HALT	$\overline{HALT}$	INPUT/OUTPUT	LOW	NO[1]	NO[1]
ENABLE	E	OUTPUT	HIGH	NO	NO
VALID MEMORY ADDRESS	$\overline{VMA}$	OUTPUT	LOW	NO	YES
VALID PERIPHERAL ADDRESS	$\overline{VPA}$	INPUT	LOW	NO	NO
FUNCTION CODE OUTPUT	FC0,FC1, FC2	OUTPUT	HIGH	NO	YES
CLOCK	CLK	INPUT	HIGH	NO	NO
POWER INPUT (2)	V_{CC}	INPUT	—	—	—
GROUND (2)	GND	INPUT	—	—	—

NOTES:
1. OPEN DRAIN

MT8- 198

MC68000 – FOOTPRINT



MT8- 199