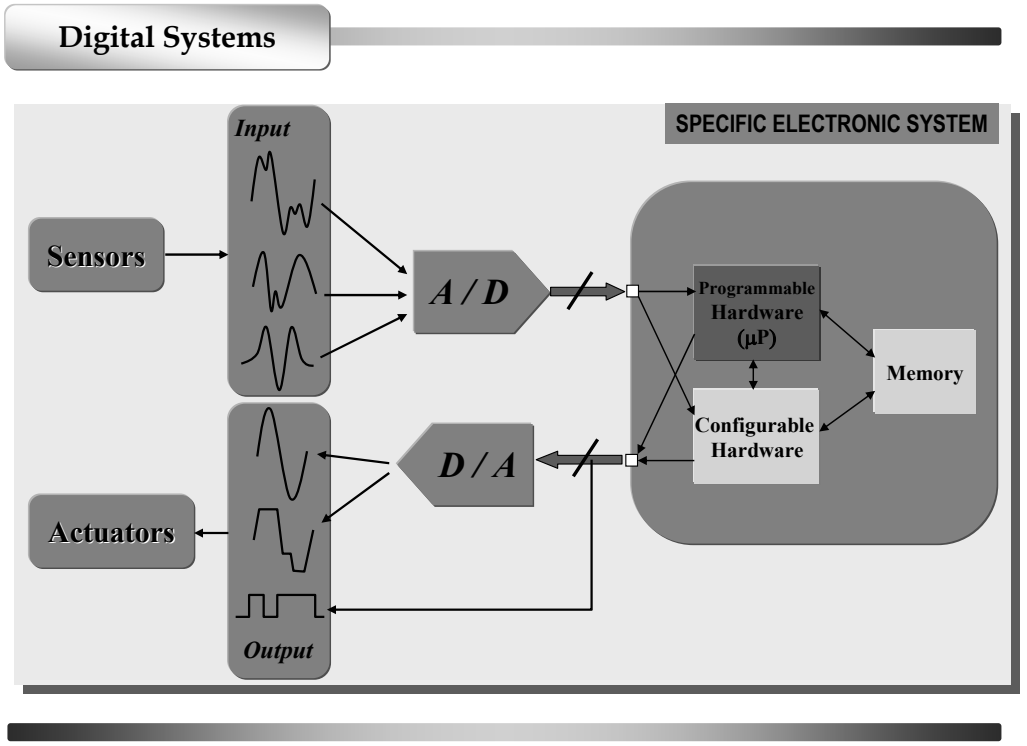
Chapter 1

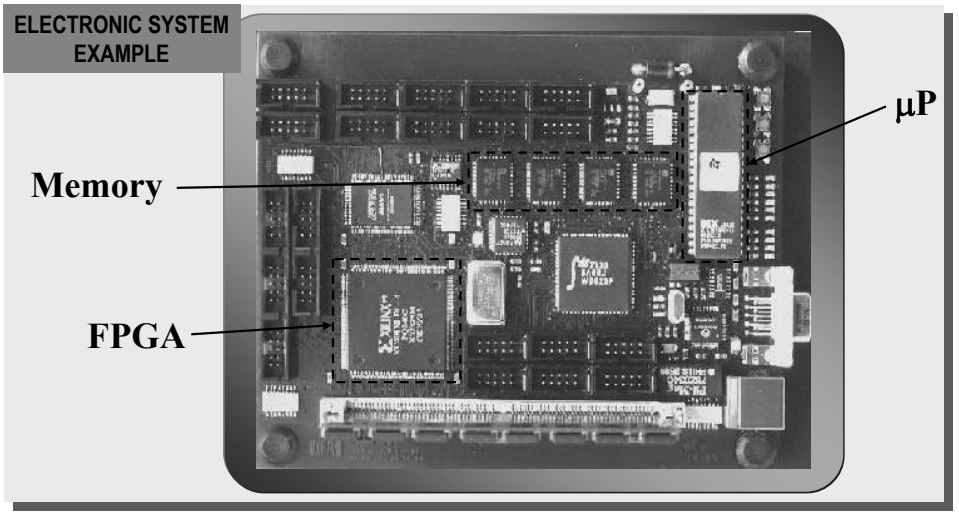
General Concepts

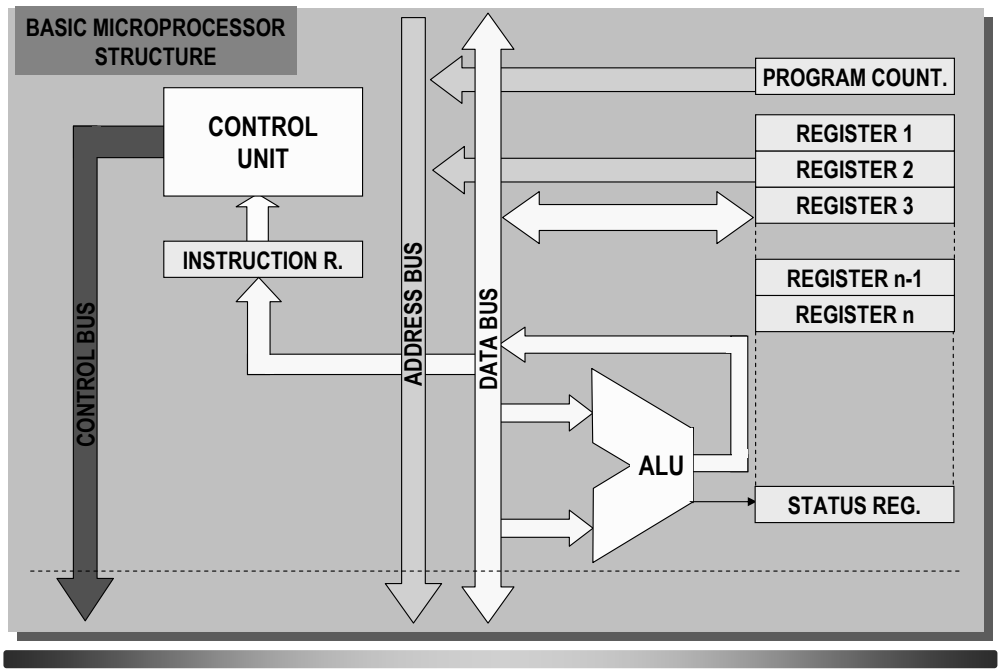


In an lectronic system we find different components :

1. Elements interfacing with physical world: Sensors and actuators
2. Elements transforming physical magnitudes (analog and continuous) into digital (discrete) : A/D and D/A converters
3. Elements that control the system and process variables: Microprocessors and configurable logic elements
4. Elements in charge of storing programs and variables: Memory

ELECTRONIC SYSTEM EXAMPLE





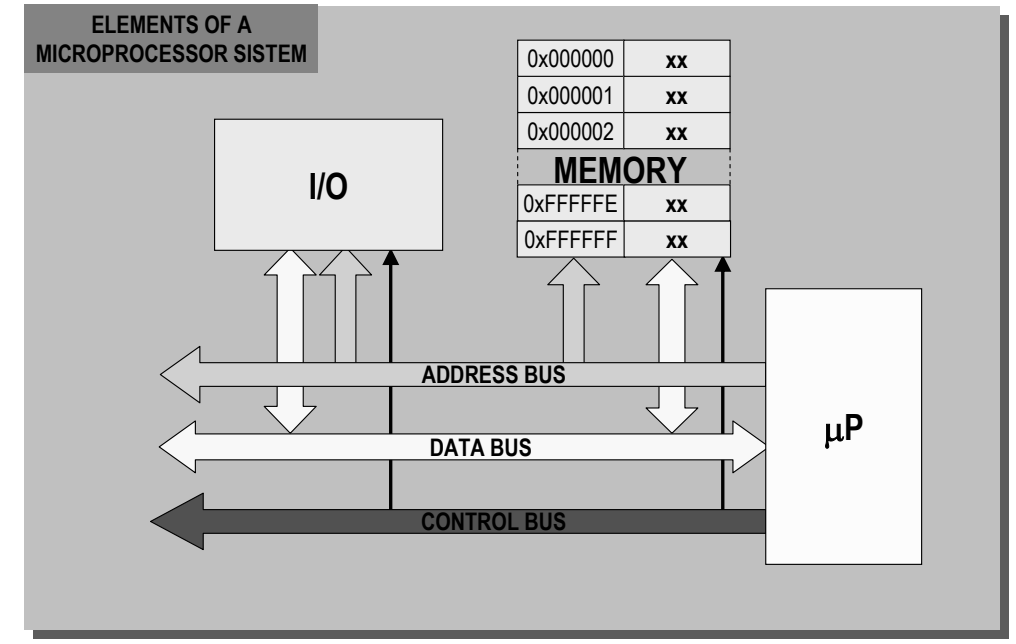
In a Microprocessor structure we have:

1. Subsystem implementing the state machine: Control unit
2. Local storage elements:

Dedicated registers (IR, PC i SR)

General purpose registers (R1...Rn)

3. Arithmetic-Logic unit: ALU
4. Interconnection elements: Busses



Microprocessador comunicates with external elements by means of three Sub-Busses

1. ADDRESS BUS.

Defines the address space accessible by the microprocessor. Every accessible element has to be within that space.

2. DATA BUS.

These lines are used to move data elements (instructions and operands) between the microprocessor and external elements.

3. CONTROL BUS.

Contains all the timing, synchronization and control signals.

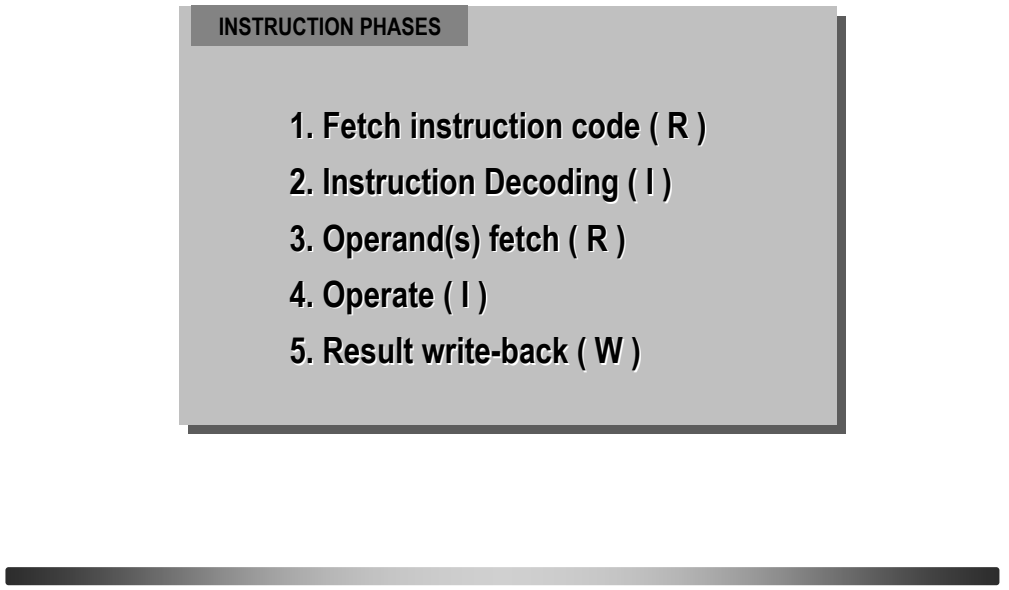
Two kind of external elements:

1. MEMORY

Contains all the information (program and data) relative to the application(s) running in the system.

2. I/O

Implements the interface between the microprocessor and the physical world (periferals)



Instruction phases:

1. Fetch instruction code

PC > ADDRESS BUS.

DATA BUS > IR

One or more READ machine cycles

2. Instruction decoding

CONTROL UNIT decyphers instruction code.

Internal OPERATION

3. Read operand(s)

PC or Rx > ADDRESS BUS

DATA BUS > Rx

One or more READ machine cycles

4. Operate

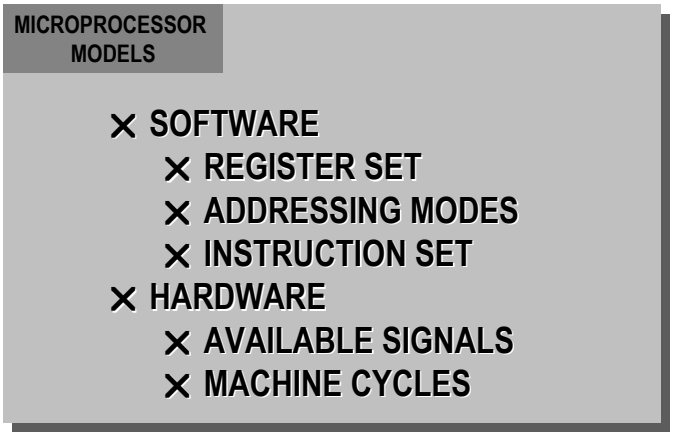
iNTERNAL OPERATION (ALU)

5. Result(s) Write-back

PC or Rx > ADDRESS BUS

Rx > DATA BUS

One or more WRITE machine cycles



To use a microprocessor in a given applicartion we don't need to know its detailed. A symplified MODEL will suffice:

For application programming: SOFTWARE MODEL

REGISTER Set: Where can we (temporary) store our objects?

INSTRUCTION Set: What can we do with different objects?

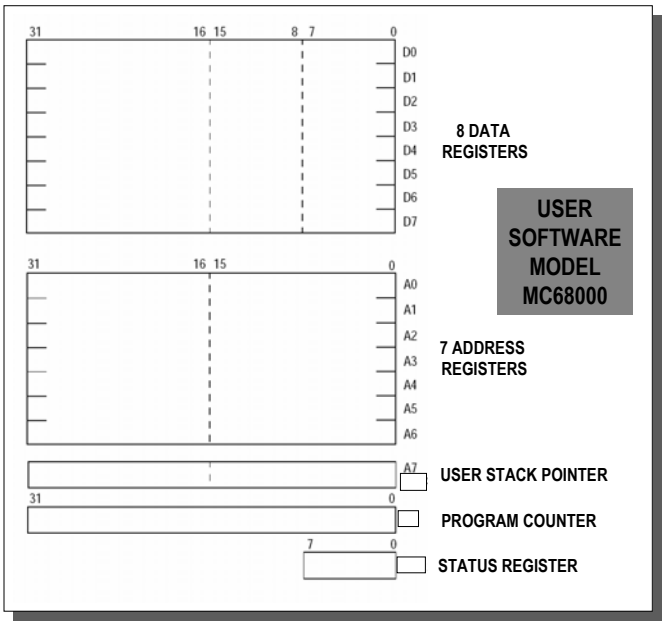
ADDRESSING Modes: How can we have access to different objects?

To build the system: HARDWARE MODEL

SIGNALS available: What signals can we connect?

MACHINE CYCLES: What is the temporal relationship between different signals?

All that information is made available by manufacturer through operation manuals.



MC68000 (Motorola, Philips...)

8 ADDRESS REGISTERS (pointers)

8 DATA REGISTERS (general purpose)

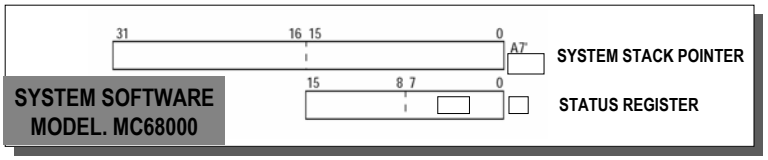
1 PC (Instruction address)

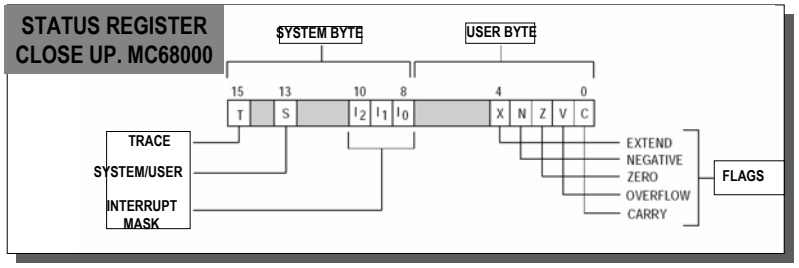
1 SR (Operation FLAGS)

Of the 8 address registers, A7 és the STACK POINTER of user data structures, and some instructions make an IMPLICIT reference to it (PEA, LINK, UNLK...)

Any <ea> (EFFECTIVE ADRESS) using a REGISTER has to use an ADDRESS REGISTER

The SR (Status register) contains the FLAGS corresponding to the last OPERATION in the ALU. However, they can be used or modified at any time.





In SUPERVISOR MODE, apart from user registers we have:

A second STACK POINTER

A CONTROL REGISTER

μP ADDRESSING MODES

1. IMPLICIT
   Operand implicit in I.C.
2. IMMEDIATE
   Operand included in I.C.
3. ABSOLUTE
   Address included in I.C.
4. DIRECT
   Operand in a Register
5. INDIRECT
   Address in a Register
6. RELATIVE
   Address as the sum of:
   - A register and an offset
   - Two registrers and an offset

ADDRESSING MODES. MC68000

Mode	Generation	Syntax
Register Direct Addressing		
Data Register Direct	EA=Dn	Dn
Address Register Direct	EA=An	An
Absolute Data Addressing		
Absolute Short	EA = (Next Word)	(xxx).W
Absolute Long	EA = (Next Two Words)	(xxx).L
Program Counter Relative Addressing		
Relative with Offset	EA = (PC)+d_{16}	(d_{16},PC)
Relative with Index and Offset	EA = (PC)+d_8	(d_8,PC,Xn)
Register Indirect Addressing		
Register Indirect	EA = (An)	(An)
Postincrement Register Indirect	EA = (An), An ← An+N	(An)+
Predecrement Register Indirect	An ◆ An–N, EA=(An)	-(An)
Register Indirect with Offset	EA = (An)+d_{16}	(d_{16},An)
Indexed Register Indirect with Offset	EA = (An)+(Xn)+d_8	(d_8,An,Xn)
Immediate Data Addressing		
Immediate	DATA = Next Word(s)	#<data>
Quick Immediate	Inherent Data	
Implied Addressing		
Implied Register	EA = SR, USP, SSP, PC,	SR,USP,SSP,PC,

EA	=	Effective Address	PC	=	Program Counter	SR	= Status Register
Dn	=	Data Register	d_8	=	8-Bit Offset (Displacement)	USP	= User Stack Pointer
An	=	Address Register	d_{16}	=	16-Bit Offset (Displacement)	SSP	= Supervisor Stack Pointer
( )	=	Contents of	Xn	=	Address or Data Register used as Index Register		

See MC68000 PROGRAMMER'S REFERENCE MANUAL:
2.2 EFFECTIVE ADRESSING MODES

DATA MOVEMENT INSTRUCTIONS. MC68000

Instruction	Operand Syntax	Operand Size	Operation
EXG	Rn, Rn	32	Rn ← → Rn
FMOVE	FPm,FPn <ea>,FPn FPm,<ea> <ea>,FPcr FPcr,<ea>	X B, W, L, S, D, X, P B, W, L, S, D, X, P 32 32	Source → Destination
FSMOVE, FDMOVE	FPm,FPn <ea>,FPn	X B, W, L, S, D, X	Source → Destination; round destination to single or double precision.
FMOVEM	<ea>,<list>[1] <ea>,Dn <list>[1],<ea> Dn,<ea>	32, X X 32, X X	Listed Registers → Destination Source → Listed Registers
LEA	<ea>,An	32	<ea> → An
LINK	An,#<d>	16, 32	SP – 4 → SP; An → (SP); SP → An, SP + D → SP
MOVE MOVE16 MOVEA	<ea>,<ea> <ea>,<ea> <ea>,An	8, 16, 32 16 bytes 16, 32 → 32	Source → Destination Aligned 16-Byte Block → Destination
MOVEM	list,<ea> <ea>,list	16, 32 16, 32 → 32	Listed Registers → Destination Source → Listed Registers
MOVEP	Dn, (d_{16},An) (d_{16},An),Dn	16, 32	Dn 31–24 → (An + d_n); Dn 23–16 → (An + d_n + 2); Dn 15–8 → (An + d_n + 4); Dn 7–0 → (An + d_n + 6) (An + d_n) → Dn 31–24; (An + d_n + 2) → Dn 23–16; (An + d_n + 4) → Dn 15–8; (An + d_n + 6) → Dn 7–0
MOVEQ	#<data>,Dn	8 → 32	Immediate Data → Destination
PEA	<ea>	32	SP – 4 → SP; <ea> → (SP)
UNLK	An	32	An → SP; (SP) → An; SP + 4 → SP

See MC68000 PROGRAMMER'S REFERENCE MANUAL:

INTEGER ARITHMETIC INSTRUCTIONS. MC68000

Instruction	Operand Syntax	Operand Size	Operation
ADD ADDA	Dn,<ea> <ea>,Dn <ea>,An	8, 16, 32 8, 16, 32 16, 32	Source + Destination → Destination
ADDI ADDQ	#<data>,<ea> #<data>,<ea>	8, 16, 32 8, 16, 32	Immediate Data + Destination → Destination
ADDX	Dn,Dn –(An), –(An)	8, 16, 32 8, 16, 32	Source + Destination + X → Destination
CLR	<ea>	8, 16, 32	0 → Destination
CMP CMPA	<ea>,Dn <ea>,An	8, 16, 32 16, 32	Destination – Source
CMPI	#<data>,<ea>	8, 16, 32	Destination – Immediate Data
CMPM	(An)+,(An)+	8, 16, 32	Destination – Source
CMP2	<ea>,Rn	8, 16, 32	Lower Bound → Rn → Upper Bound
DIVS/DIVU DIVSL/DIVUL	<ea>,Dn <ea>,Dr–Dq <ea>,Dq <ea>,Dr–Dq	32 ÷16 → 16,16 64 ÷32 → 32,32 32 ÷32 → 32 32 ÷32 → 32,32	Destination ÷Source → Destination (Signed or Unsigned Quotient, Remainder)
EXT EXTB	Dn Dn Dn	8 → 16 16 → 32 8 → 32	Sign-Extended Destination → Destination
MULS/MULU	<ea>,Dn <ea>,Dl <ea>,Dh–Dl	16 x 16 → 32 32 x 32 → 32 32 x 32 → 64	Source x Destination → Destination (Signed or Unsigned)
NEG	<ea>	8, 16, 32	0 – Destination → Destination
NEGX	<ea>	8, 16, 32	0 – Destination – X → Destination
SUB SUBA	<ea>,Dn Dn,<ea> <ea>,An	8, 16, 32 8, 16, 32 16, 32	Destination = Source → Destination
SUBI SUBQ	#<data>,<ea> #<data>,<ea>	8, 16, 32 8, 16, 32	Destination – Immediate Data → Destination
SUBX	Dn,Dn –(An), –(An)	8, 16, 32 8, 16, 32	Destination – Source – X → Destination

See MC68000 PROGRAMMER'S REFERENCE MANUAL:

LOGIC, BIT AND BCD INSTRUCTIONS. MC68000

Instruction	Operand Syntax	Operand Size	Operation
AND	<ea>,Dn Dn,<ea>	8, 16, 32 8, 16, 32	Source $\wedge$ Destination $\rightarrow$ Destination
ANDI	#<data>,<ea>	8, 16, 32	Immediate Data $\wedge$ Destination $\rightarrow$ Destination
EOR	Dn,<ea>	8, 16, 32	Source $\oplus$ Destination $\rightarrow$ Destination
EORI	#<data>,<ea>	8, 16, 32	Immediate Data $\oplus$ Destination $\rightarrow$ Destination
NOT	<ea>	8, 16, 32	~ Destination $\rightarrow$ Destination
OR	<ea>,Dn Dn,<ea>	8, 16, 32	Source $\vee$ Destination $\rightarrow$ Destination
ORI	#<data>,<ea>	8, 16, 32	Immediate Data $\vee$ Destination $\rightarrow$ Destination

Instruction	Operand Syntax	Operand Size	Operation
BCHG	Dn,<ea> #<data>,<ea>	8, 32 8, 32	~ (<Bit Number> of Destination) $\rightarrow$ Z $\rightarrow$ Bit of Destination
BCLR	Dn,<ea> #<data>,<ea>	8, 32 8, 32	~ (<Bit Number> of Destination) $\rightarrow$ Z; 0 $\rightarrow$ Bit of Destination
BSET	Dn,<ea> #<data>,<ea>	8, 32 8, 32	~ (<Bit Number> of Destination) $\rightarrow$ Z; 1 $\rightarrow$ Bit of Destination
BTST	Dn,<ea> #<data>,<ea>	8, 32 8, 32	~ (<Bit Number> of Destination) $\rightarrow$ Z

Instruction	Operand Syntax	Operand Size	Operation
ABCD	Dn,Dn –(An), –(An)	8 8	$Source_{10} + Destination_{10} + X \rightarrow$ Destination
NBCD	<ea>	8	$0 - Destination_{10} - X \rightarrow$ Destination
PACK	–(An), –(An) #<data> Dn,Dn,#<data>	16 $\rightarrow$ 8 16 $\rightarrow$ 8	Unpackaged Source + Immediate Data $\rightarrow$ Packed Destination
SBCD	Dn,Dn –(An), –(An)	8 8	$Destination_{10} - Source_{10} - X \rightarrow$ Destination
UNPK	–(An),–(An) #<data> Dn,Dn,#<data>	8 $\rightarrow$ 16 8 $\rightarrow$ 16	Packed Source $\rightarrow$ Unpacked Source Unpacked Source + Immediate Data $\rightarrow$ Unpacked Destination

See MC68000 PROGRAMMER'S REFERENCE MANUAL:

PROGRAM CONTROL INSTRUCTIONS. MC68000

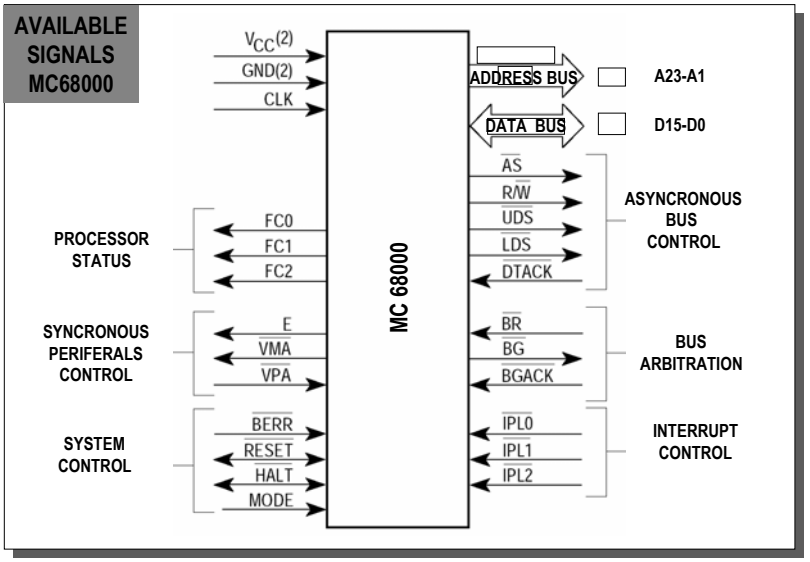
Instruction	Operand Syntax	Operand Size	Operation
Integer and Floating-Point Conditional			
Bcc, FBcc	<label>	8, 16, 32	If Condition True, Then PC + d_n → PC
DBcc, FDBcc	Dn,<label>	16	If Condition False, Then Dn – 1 → Dn If Dn → –1, Then PC + d_n → PC
Scc, FScc	<ea>	8	If Condition True, Then 1's → Destination; Else 0's → Destination
Unconditional			
BRA	<label>	8, 16, 32	PC + d_n → PC
BSR	<label>	8, 16, 32	SP – 4 → SP; PC → (SP); PC + d_n → PC
JMP	<ea>	none	Destination → PC
JSR	<ea>	none	SP – 4 → SP; PC → (SP); Destination → PC
NOP	none	none	PC + 2 → PC (Integer Pipeline Synchronized)
FNOP	none	none	PC + 4 → PC (FPU Pipeline Synchronized)
Returns			
RTD	#<data>	16	(SP) → PC; SP + 4 + d_n → SP
RTR	none	none	(SP) → CCR; SP + 2 → SP; (SP) →PC; SP + 4 → SP
RTS	none	none	(SP) → PC; SP + 4 → SP

Letters cc in the integer instruction mnemonics Bcc, DBcc, and Scc specify testing one of the following conditions:

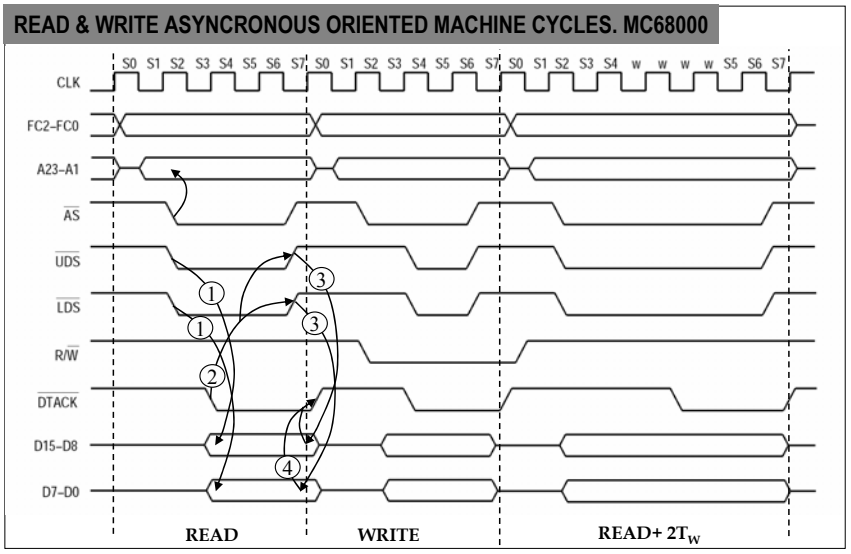
CC—Carry clear	GE—Greater than or equal
LS—Lower or same	PL—Plus
CS—Carry set	GT—Greater than
LT—Less than	T—Always true*
EQ—Equal	HI—Higher
MI—Minus	VC—Overflow clear
F—Never true*	LE—Less than or equal
NE—Not equal	VS—Overflow set

*Not applicable to the Bcc instructions.

See MC68000 PROGRAMMER'S REFERENCE MANUAL:

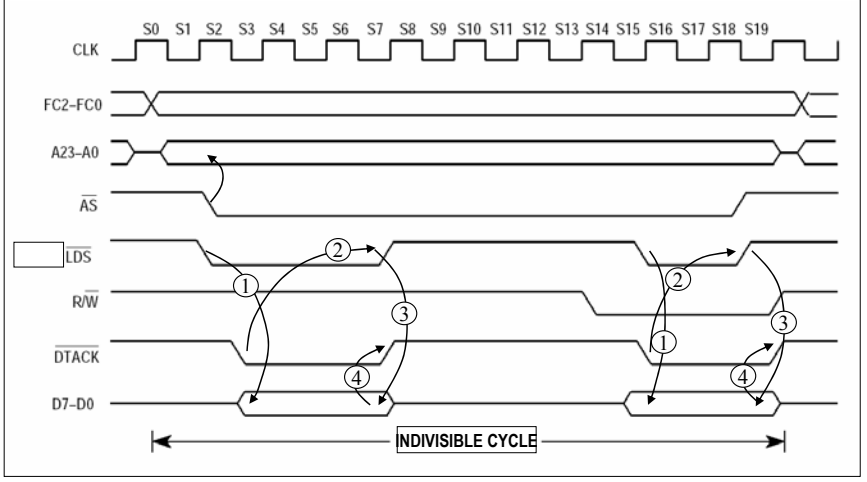


See MC68000 8-/16-/32 BIT MICROPROCESSOR USER'S MANUAL

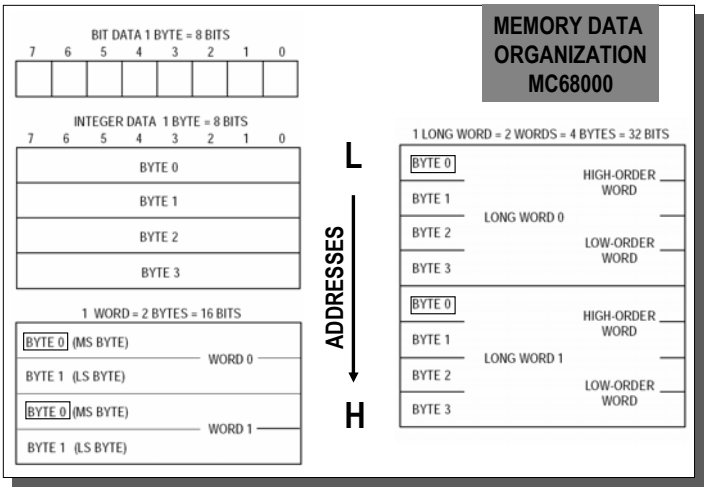


See MC68000 8-/16-/32 BIT MICROPROCESSOR USER'S MANUAL

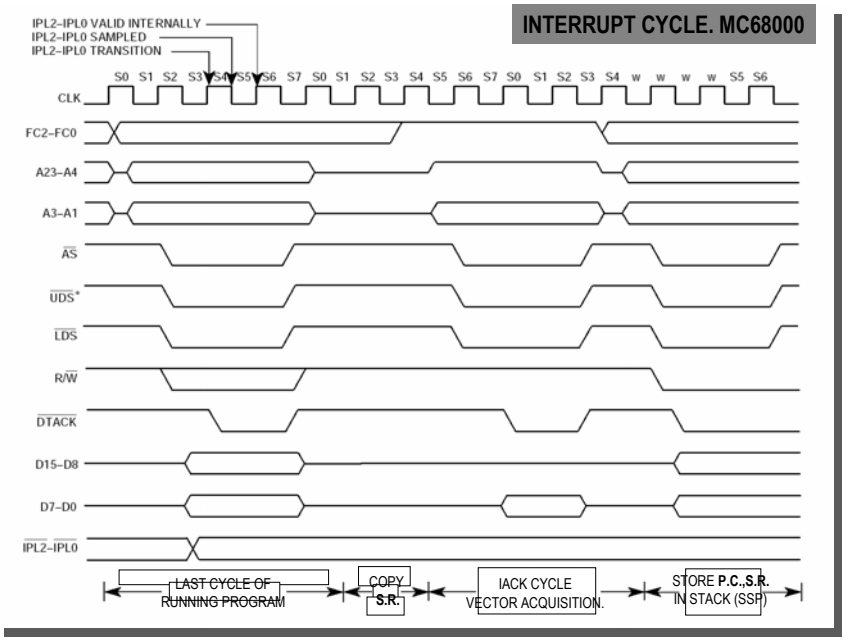
RMW MACHINE CYCLE. MC68000



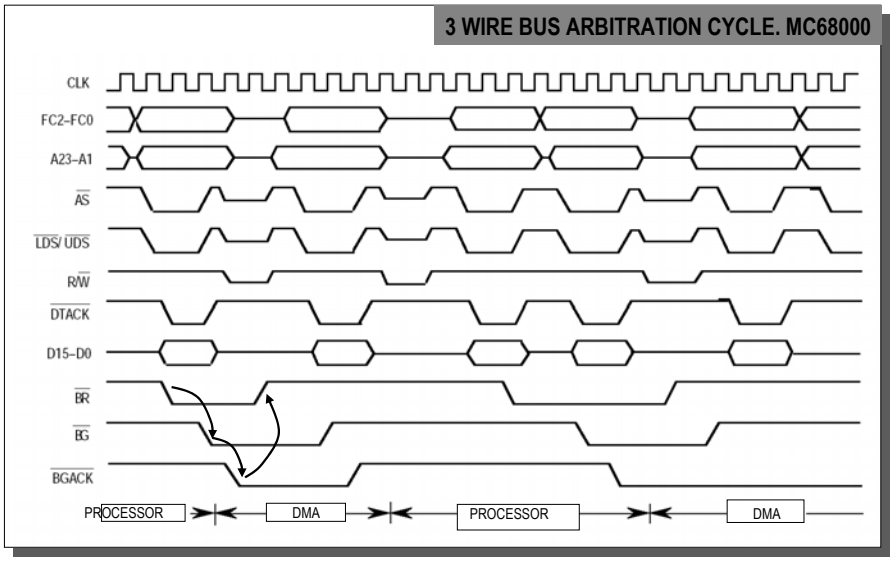
See MC68000 8-/16-/32 BIT MICROPROCESSOR USER'S MANUAL



See MC68000 8-/16-/32 BIT MICROPROCESSOR USER'S MANUAL



See MC68000 8-/16-/32 BIT MICROPROCESSOR USER'S MANUAL



See MC68000 8-/16-/32 BIT MICROPROCESSOR USER'S MANUAL