INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

H4.SMR. 403/29

FIFTH COLLEGE ON MICROPROCESSORS: TECHNOLOGY AND APPLICATIONS IN PHYSICS

2 - 27 October 1989

Other Microprocessors
Part 2

D. CROSETTO
CERN, SPS Division, Geneva, Switzerland

SPARC Industry Overview

- A flexible, scalable RISC architecture
- Current Implementations are in CMOS (custom and gate array), ECL and GaAs
- A complete architecture
  - Integer
  - Floating Point
  - Memory Management
  - Cache
  - Multiprocessing
- Voted a candidate for the MIL-STD 32-bit instruction set architecture by the SAE task group chartered by the Department of Defense

SPARC Industry Overview

- The first and only open RISC architecture
- Created by Sun Microsystems in response to the declining performance enhancements of CISC microprocessors
- Open Industry standard
- Based on the RISC work of Dr. David Patterson at UC Berkeley
- Supported by 5 semiconductor licensees
  - Cypress Semiconductor, Texas Instruments, LSI Logic, Fujitsu, & BIT
- Over 80 design wins
- Over 200 application software packages already available
- Complete development support is in place

Workstations

- Single User, High Performance Workstation
- Sun 4-100 Series – (7 MIPS /-$30K)

  16 Mhz SPARC Processor/Floating Point Unit
  8 Mbytes Main Memory
  19 Inch 1152 x 900 Monochrome Monitor
  141 Mbyte Drive
  60 Mbyte 1/4" Tape Backup

- Server for a Project Team
- Sun 4-200 Series – (10 MIPS /-$45K)

  16 Mhz SPARC Processor/Floating Point Unit
  128 Kbytes Cache
  8 Mbytes Main Memory
  19 Inch 1152 x 900 Monochrome Monitor
  327 Mbyte Drive
  60 Mbyte 1/4" Tape Backup

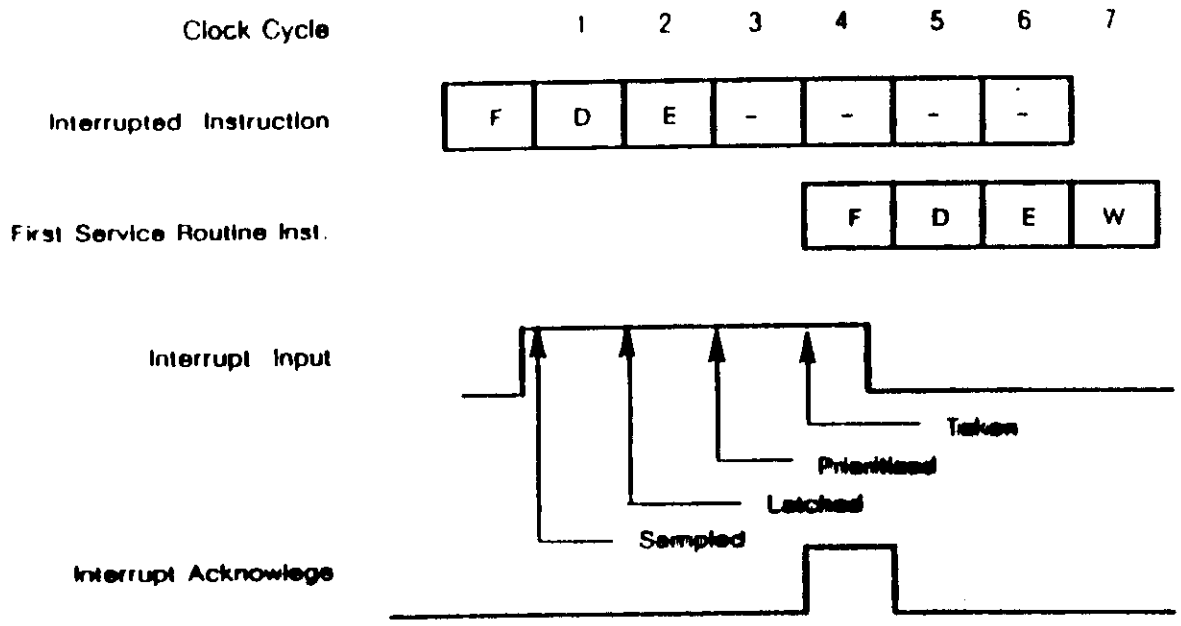
Competitive Analysis *

	Cypress 7C600	AMD 29000	Motorola 88000	MIPS R3000
Implementation	0.8µ CMOS	1.2µ CMOS	1.5µ CMOS	1.2µ CMOS
Clock Frequency	33–50MHz	16–30MHz	20MHz	16–25MHz
Scalable Cache Support	Yes	None	Yes	None
Cache Extensibility	Yes	None	Yes	None
Cache Size	64Kb–256Kb	None	32Kb–128Kb	None
Multiprocessing Support	Yes	None	Yes	None
Cache Coherency	Yes–Single Cycle Real–time	None	Yes–Multi Cycle Non Real–time	None
Semaphore Support	Yes	Yes	Yes	None
Floating Point	64–bit Double Precision	64–bit Double Precision	32–bit Single Precision	64–bit Double Precision
Register Windows	Yes	Partial (Stack Cache Only)	None	None
Dedicated Floating Point Registers	Yes	None	None	Yes

* Based on public info as of 12/88

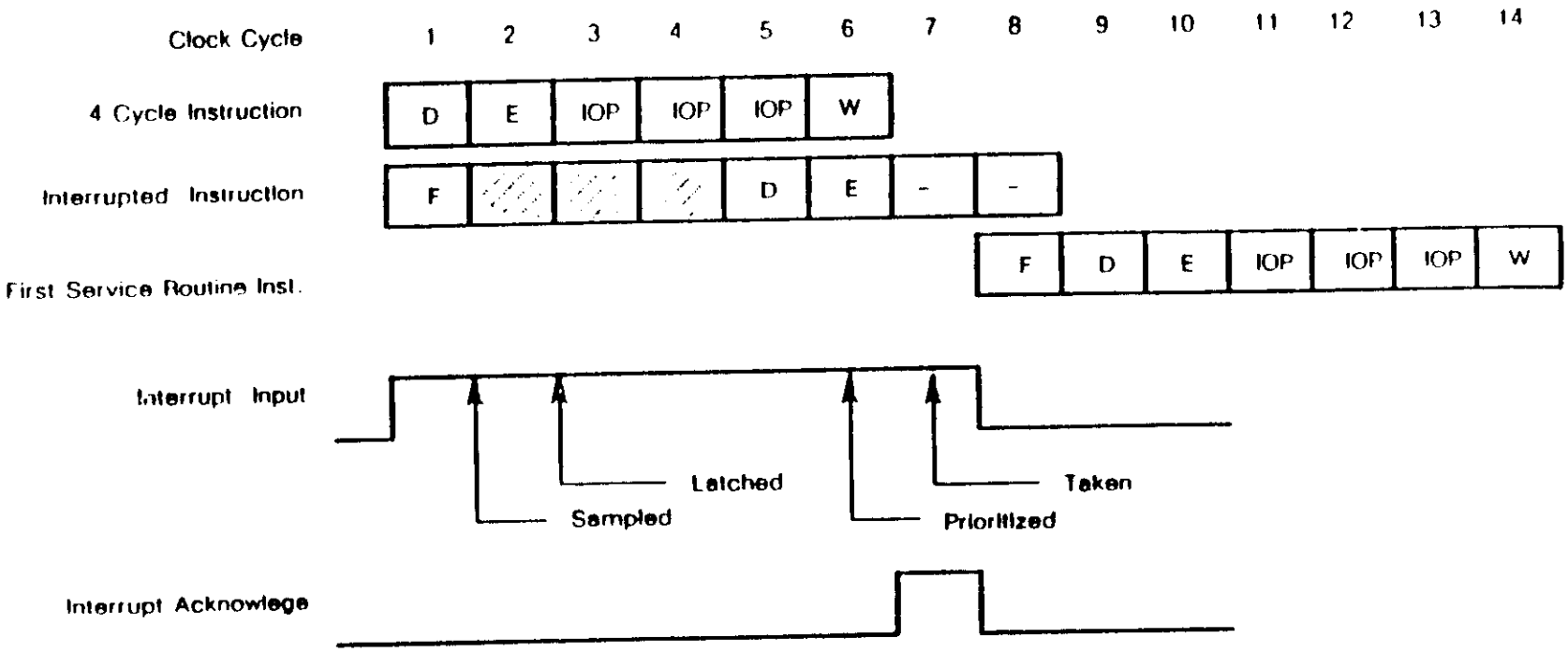
210 nSec

Best Case Interrupt Timing



Worst Case Interrupt Timing

420 nsec.



7C601 Context Switch

OPERATION	APPROX. CLOCKS	TIME @ 40MHZ
Enter Trap Handler from Interrupt	7–14	175 – 350 ns
Save Registers	1 – 262	25 ns – 6.6 us
Save FP Registers	0 – 78	0 – 2.0 us
Call Routine	2	50 ns
	10 – 356	250 ns – 8.9 us
Restore Registers	1 – 196	25 ns – 4.9 us
Restore FP Registers	0 – 58	0 – 1.5 us
	1 – 254	25 ns – 6.4 us

Real Time Operating System for SPARC

- Vx Works – Development
  - UNIX Based
  - Network Based Environment
    - TCP/IP Communications
    - Network File System (NFS)
      - Ethernet
      - PRONET
      - Shared Memory

Development Tools

Software:

- Compilers – C, Fortran, Pascal
  - Sun 3 – Cross Compilers
    - (-$1000 for C, Fortran and Pascal)
  - Sun 4 – Compilers
    - (-$1000 each for Fortran and Pascal, C comes with system)
- SPARCsim – Architectural Simulator
  - (-$5K for object/-$25K for source)
- RDBX – Remote Debugger and Monitor
  - C Based Monitor – Target Board/System
    - (-$500 for object/-$5K for source/-$30K for buyout)
  - Remote DBX Tool – Host Side
    - (-$ K for object/-$50K for source)

SPARC Application Software Base

- The largest shrinkwrap software base of any RISC microprocessor
- Thousands of times the lines of application code for all other RISC microprocessors combined
- World Class software covering all major applications of computing:
  - Office Automation
  - Financial Services
  - VAX/VMS Emulation
  - Artificial Intelligence
  - Electronic Design Automation
  - Manufacturing
  - Mechanical CAD
  - CASE
  - Project Management
  - MS-DOS Emulation
  - Electronic Publishing
  - Databases
  - Graphics
  - Biomedical
  - Imaging
  - Mathematics & Earth Resources
- Growing at an exponential rate away from the competition

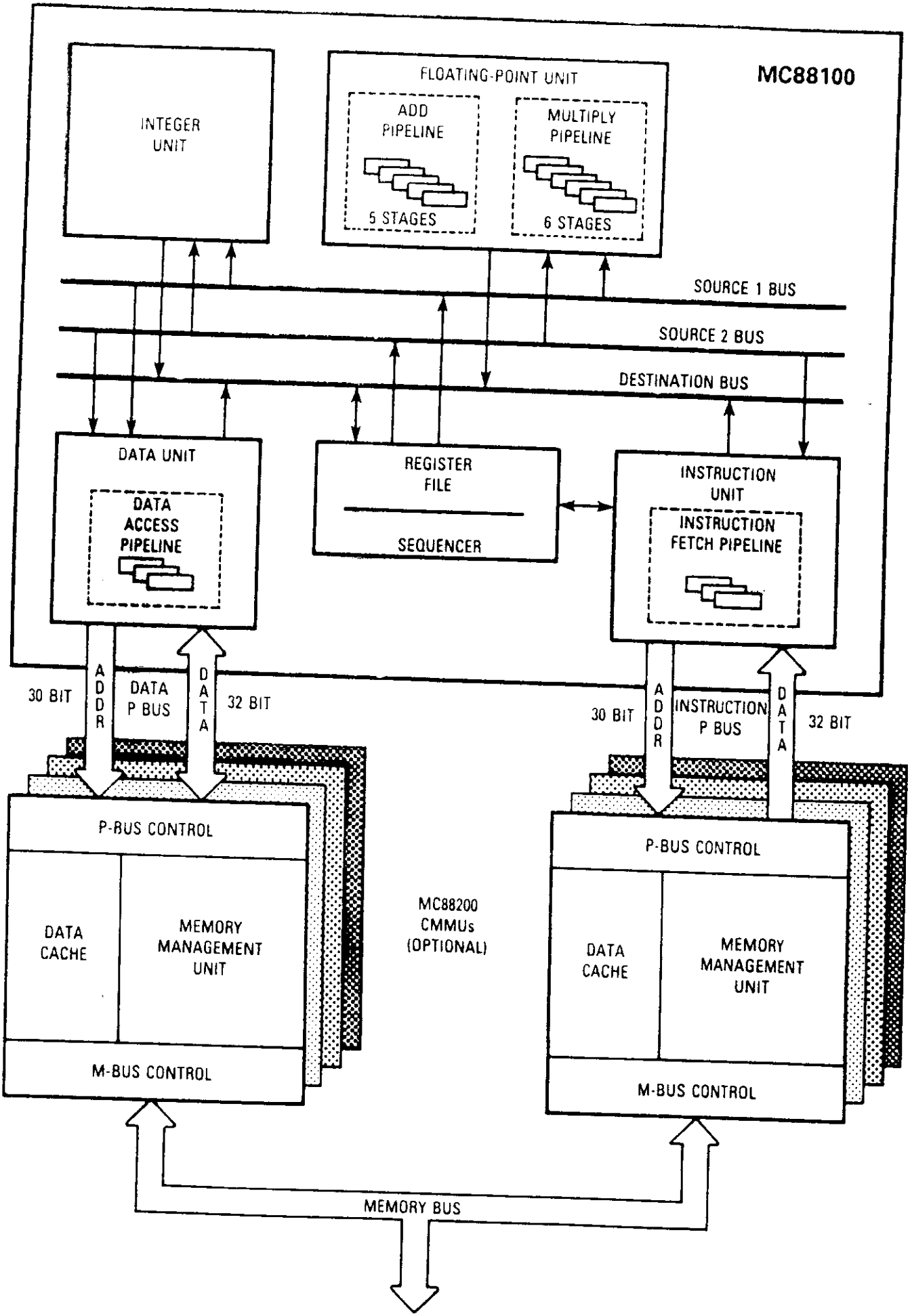


Figure 1-1. MC88100/MC88200 Block Diagram

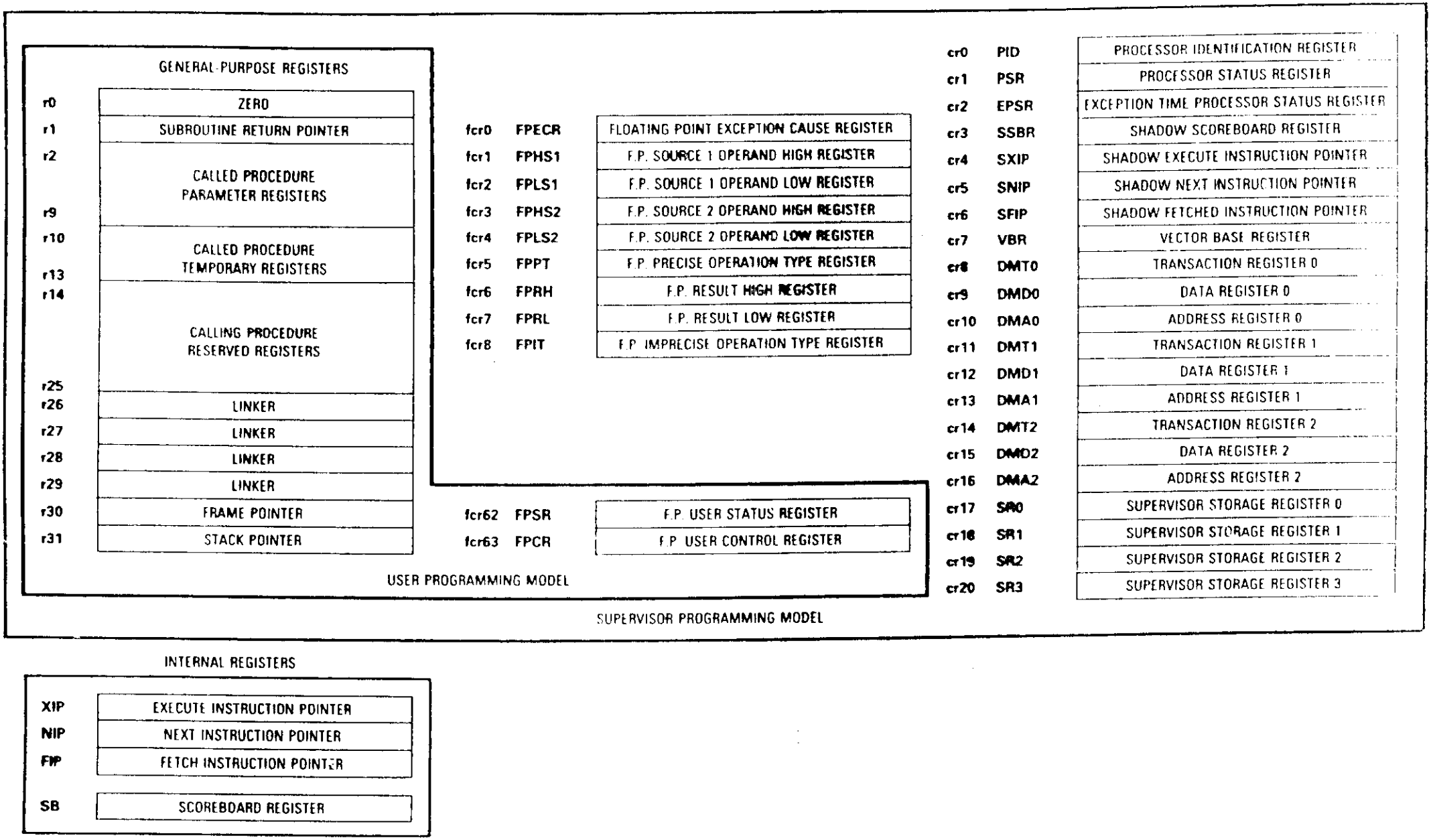


Figure 1-2. Programming Model

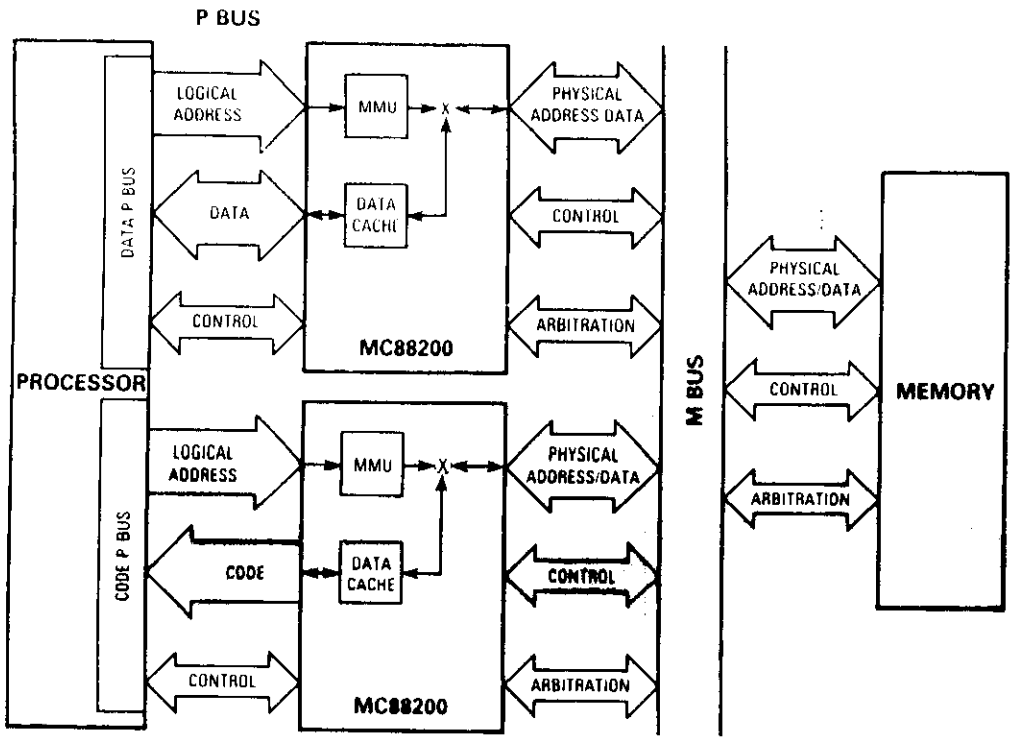


Figure 1-2. MC88200 System Configuration

memory systems that do not need address translation. One of two sets of protection and control attributes may be applied to the transaction, with selection dependent on the supervisory or user mode of the processor.

If there is no ATC entry that matches the processor's logical address, an ATC "miss" occurs, and the MC88200 searches translation tables in memory for a correct translation.

1.3.1 Address Translation Tables

When an ATC miss occurs, the MC88200 issues waits on the P bus and requests ownership of the M bus. When the M bus is available, the MC88200 completes the bus arbitration sequence, assumes M bus ownership, and performs the table search. If no exception conditions occur, the required translation descriptor is loaded into the PATC. The MC88200 then completes the memory access, translating the address through the PATC. When the memory access is complete, the MC88200 signals the processor that the

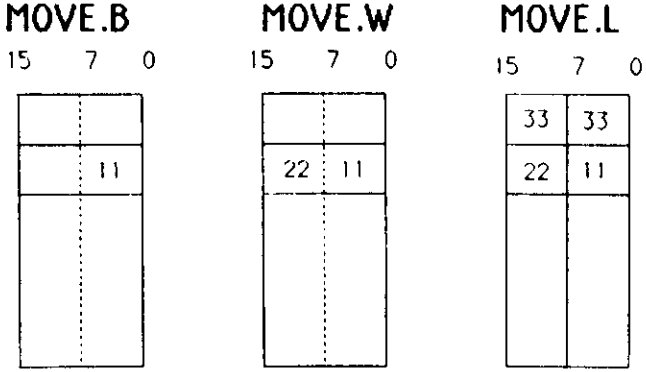
M68000 Operand sizes and storage of data in memory

Operand sizes:

- Byte (8 bits)
- Word (16 bits)
- Long word (32bits)

Almost all instructions allow definition of operand sizes:

M6809:	M68000:
STA MEM	MOVE.B D0,MEM
STD MEM	MOVE.W D0,MEM
	MOVE.L D0,MEM



The M68000

- The M68000 is a 16 bit microprocessor
  (externally 16 data bus pins)
  internally it has a 32 bit data path
- It has 24 address lines
  permitting an address range of 16 Mbytes
  directly accessible (without MMU)
- Data and address lines are non multiplexed
- It is completely incompatible to earlier
  (8 bit) microprocessors of Motorola
- 8 bit peripherals (e.g.PIA) may easily be
  interfaced to the M68000

The use of micros

8 Bit machines:

'Simple' controllers of machines (coffee machine etc.)

Depending on speed requirements:

Interface cards
Control system (e.g. temperature control
mechanical systems
gas flow control for detector...)

Data acquisition

Simple development system (Rosy/Flex)

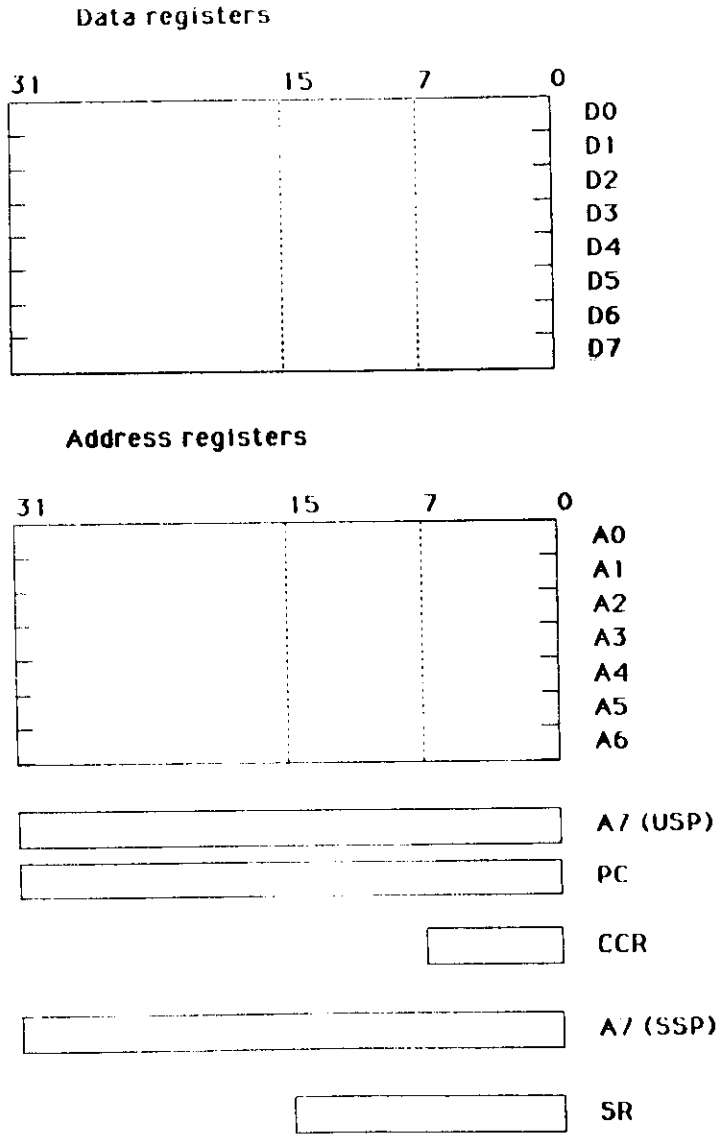
16 Bit machines:

More elaborate development systems (e.g Unix)
Fast controller cards
Data acquisition and control

32 bit machines:

Fast multi-user multi-tasking systems
Data evaluation
Number crunching (with coprocessor)
Graphics

M68000 programmer's model



Lecture summary

- Introduction
- Why 16 bit machines or why still 8 bit machines
- M68000 architecture (register layout)
- The 68000 instruction set
  - -> more addressing modes
  - -> features for operating system support
  - -> features for high level language support
- **Interfacing the M68000 to memory and external hardware**
  - -> asynchronous bus
  - -> interrupts
  - -> DMA and multiprocessing support
- M68020
  - -> cache memory
  - -> coprocessors
- Some aspects of the 80286
  - -> instruction set summary
  - -> on chip memory management
- M68030 Motorola
- 80386 Intel
- M68040 Motorola

Introduction

Where do we use micros?

- Controllers of machines
  - -> coffee machine
  - -> washing machine
  - -> camera
  - -> cars
  - -> ... many more

micro controllers

- at university, in the laboratory:
  - -> complex controllers
  - -> interface cards e.g. ADC,TDC subsystems
  - -> data acquisition
  - -> control systems

- Development systems
  - -> for program development e.g. programs used in controller cards as descibed above
  - -> for data evaluation
  - -> for administration
  - -> for text processing
  - -> for number crunching (physics simulation)

Would you use the same micro for all these applications?

Where is the limit between a micro computer and a mini computer (M68020, MicroVax)

Why do we go to 16/32 bit micros?

- It is more fun for the programmer!
- We need more computing power (a certain job has to be done in a fixed, very short time)
- The programs become so big, they do not fit into 64 kbytes of memory
- The instruction set is more adequate for high level languages or operating systems
- Faster response to external events

Goal of these lectures:

Show how to gain speed = computing power

Increasing the size of the data bus and increaing the clock frequency is **not** all

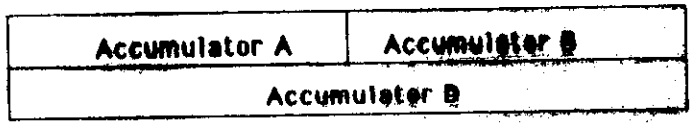
But . . .

Higher speed means

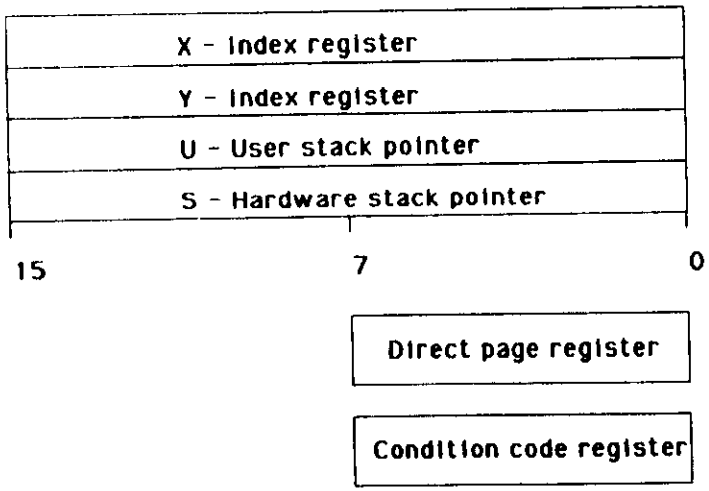
Higher complexity
Higher cost

M6809 programmer's model

Data registers:



Address registers:



Advantages of M68000 register layout

- More data registers:

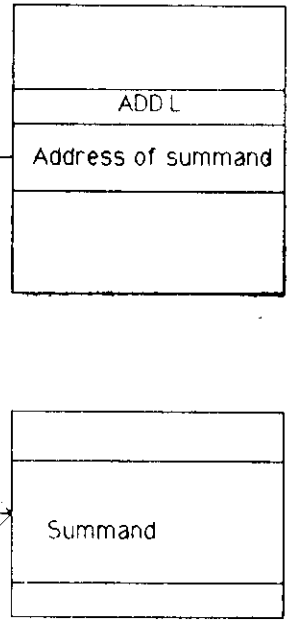
  -> less memory access
  -> direct access to 32 bits of data
  (internal 32 bit data and address bus)

- More address registers:

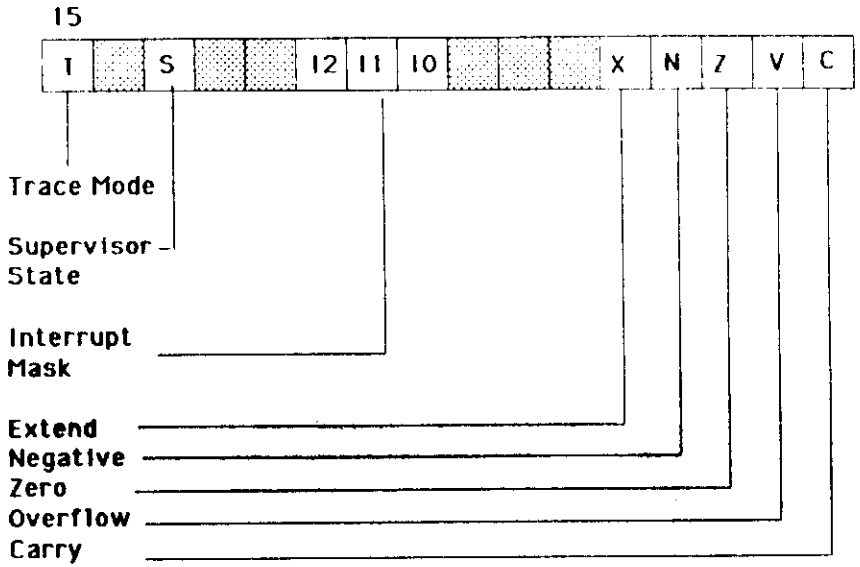
  -> more storage space for pointers
  -> faster table access

Example: (Add two values)	Nr of cycles
register to register add:	
ADD.L D1,D0	8+0
add with pointer in A0:	
ADD.L (A0)+,D0	8+8
add with absolute address:	
ADD.L Summand,D0	8+16

- Read Opcode
  (1 meory access)
- Read address
  (2 memory accesses)
- Read summand
  (2 memory accesses



The status register



A look at the M68000 instruction set

Most instructions in a program are data movements
(try to compare the number of instructions listed
below to any other type of instructions like arithmetic
logical etc.)

M6809		M68000		
		MOVE	SOURCE,DESTINATION	
LDA	STA	MOVE	MEM,D0	MOVE D0,MEM
LDB	STB	MOVE	MEM,D1	MOVE D1,MEM
LDX	STX	MOVE.L	MEM,A0	MOVE.L A0,MEM
LDY	STY	MOVE.L	MEM,A1	MOVE.L A1,MEM
LDU	STU	MOVE.L	MEM,A7	MOVE.L A7,MEM
LDS	STS	same as MOVE.L	MEM,SP	MOVE.L SP,MEM
LEAX				
LEAY				
LEAX	OFFSET,PCR	LEA	OFFSET(PC),A0	
PSHS	PULS	MOVE.L	D0,-(SP)	MOVE.L (SP)+,D0
PSHU	PULU			

Many data movement instructions are replaced
by the MOVE instruction

More on instructions

All the instructions you know from the M6809 are there:
arithmetic instructions:
ADD, SUB, LSR, LSL, ROL, ROR, MULU, MULS, DIVU, DIVS.....
as well as the branch instructions:
BRA, BR_{CC} where CC may be EQ, NE, CC, CS....

The branch instruction allow offset size specification:

BRA.S	OFFSET	8 bit offset
BRA	OFFSET	16 bit offset

Some Goodies

DB_{CC}	If (conditions false) then Dn-1 -> Dn if Dn <> -1 then PC+d -> PC else PC+2 -> PC (fall through to next instruction)
DBRA	
DBEQ	
DBCC...	

Syntax: DBRA D0,Loop

Example:

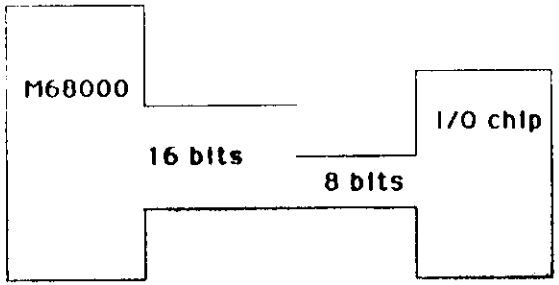
```
*  Program waits for a peripheral to respond
*  within a given time. Response is signalled
*  by a flag bit in the I/O chips status register

    some stuff here


            MOVE.L    #IOChip_Address,A0   A0 points to the
*                                          I/O chip
            MOVE.W    #Time_Out,D0         Timeout count
Timeout_Loop
            BTST.B    #IOFlag(A0),D0       See if peripheral
*                                          responded
            DBNE      D0,TimeOut_Loop      wait if timeout
*                                          not expired
```

Talking to 8 bit peripherals:

MOVE.L D0,Peripheral does not work on 8 bit peripherals



MOVEP.L D0,IOREG(A0)

Sends 4 bytes of data to the peripheral incrementing the address by 2 after each memory cycle.

The students' favorite subject: Addressing modes

M6800	M68000
Inherent: CLRA	Address/Data register direct CLR.W D0
Immediate LDA #$50	Immediate MOVE.W #$50,D0
Extended, Direct LDA $50	Absolute short/long address MOVE.W $8700.W,D0 address is sign extended thus picks up data at $FF8700 MOVE.W $8700.L,D0
Indexed LDA ,X LDA OFFSET,X	Address register indirect MOVE.W (A0),D0 Address register indirect with displacement MOVE.W OFFSET(A0),D0

Accumulator offset from register
LDA B,X
LDY D,X
	Address register indirect with index
	MOVE.W OFFSET(A0,D1),D0

More on addressing modes

Autoincrement/decrement from register
```
LDD   ,X++
```

Addressregister inderect with postincrement
```
MOVE.W   (A0)+,D0
```
(increments A0 by 2 increments A0 by 4 if longword transfer)

Extended indirect
```
LDA   [$F000]
```

PC relative
```
LDA  OFFSET,PC
```

Program counter with displacement
```
MOVE.W OFFSET(PC),D0
```
Program counter with index
```
MOVE.W OFFSET(PC,D1.L),D0
```

Instructions specially suited for implementations of high level languages

Imagine a PASCAL procedure:

```
procedure example(var
                      first_par:    integer;
                      second_par : real);

var

   local_variable_1   :  integer;
   local_variable_2   :  real;

begin

(*   some code   *)

end;
```

The variable local_variable_1&2 are completely local to the procedure (they cannot be seen in the main program). The best place to store these variables is therefore the stack.

The M68000 instructions LINK and UNLK provide the ideal tool for easy allocation of space for local variables on the stack

LINK

PERATION: An → -(SP) ; SP → An ; SP+d → SP

SSEMBLER SYNTAX: LINK An, #<displacement>

TTRIBUTES: UNSIZED

ESCRIPTION: THE CURRENT CONTENT OF THE SPECIFIED ADDRESS REGISTER IS PUSHED ONTO THE STACK. AFTER THE PUSH, THE ADDRESS REGISTER IS LOADED FROM THE UPDATED STACK POINTER. FINALLY, THE [redacted] IS SPECIFIED TO ALLOCATE STACK [redacted]

[redacted]

LINK AND UNLK CAN BE USED TO MAINTAIN A LINKED LIST OF LOCAL DATA AND PARAMETER AREAS ON THE STACK FOR [redacted] SUBROUTINE CALLS

INSTRUCTIONS SPECIFICALLY SUITED FOR IMPLEMENTATIONS OF HIGH LEVEL LANGUAGES

LINK AND UNLK

SYNTAX: LINK An, #DISPLACEMENT

EXAMPLE: LINK A6, # LOCAL VARIABLES

FUNCTION: 1) PUSH CURRENT CONTENT OF ADDRESS REGISTER ONTO THE STACK

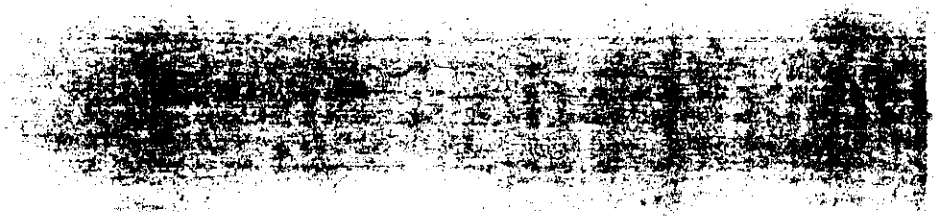
1) AFTER THE PUSH LOAD THE ADDRESS REGISTER [redacted] STACK POINTER

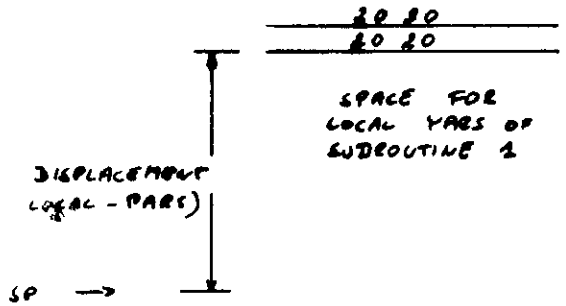
2) [redacted] TO ALLOCATE STACK SPACE BY SUBTRACTING [redacted] FROM THE STACK POINTER

→ 10000

A6 : [redacted]

LINK IS EXECUTED AT ENTRY INTO THE SUBROUTINE TO ALLOCATE THE STACK SPACE NEEDED OF LOCAL VARIABLES

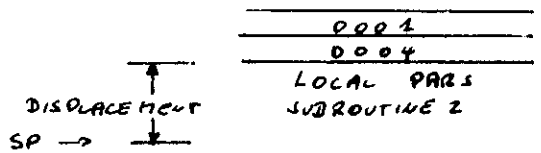




R6 : 10004

RED : AFTER FIRST SUBROUTINE CALL

R6 : 10004 + DISPLACMT (RED) + 4



BLUE : AFTER SECOND SUBROUTINE CALL

INSTRUCTIONS SPECIFICALLY SUITED FOR

IMPLEMENTATIONS OF HIGH LEVEL LANGUAGES

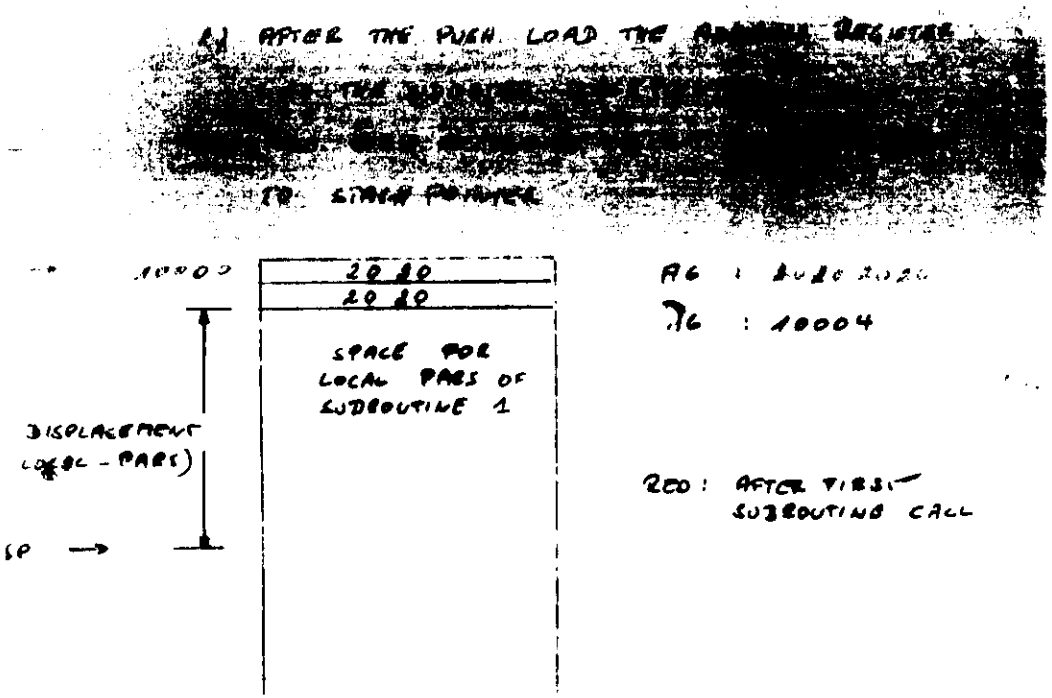
LINK AND UNLINK

SYNTAX : LINK An, #DISPLACEMENT

EXAMPLE : LINK A6, # LOCAL PARAMETERS

FUNCTION : a) PUSH CURRENT CONTENT OF ADDRESS REGISTER ONTO THE STACK

b) AFTER THE PUSH, LOAD THE ADDRESS REGISTER WITH THE ADDRESS OF THE STACK POINTER



A6 : 2020 2020

A6 : 10004

RES : AFTER FIRST SUBROUTINE CALL

LINK IS EXECUTED AT ENTRY INTO THE SUBROUTINE TO ALLOCATE THE STACK SPACE NEEDED OF LOCAL VARIABLES

INSTRUCTIONS SPECIFICALLY SUITED FOR

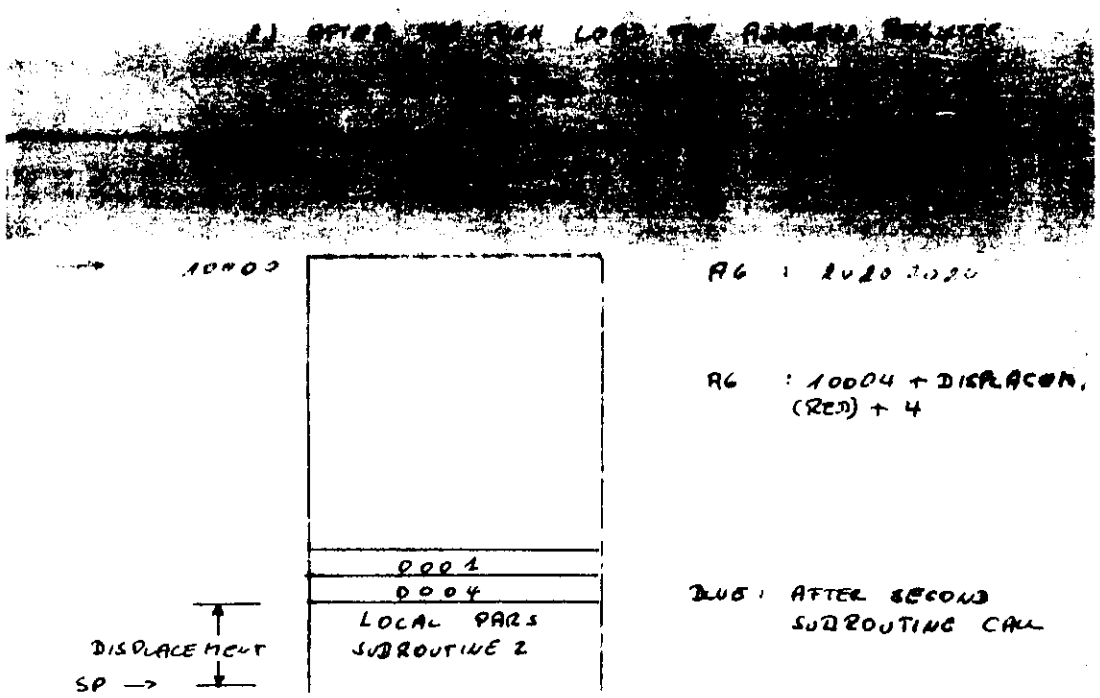
IMPLEMENTATIONS OF HIGH LEVEL LANGUAGES

LINK AND UNLINK

SYNTAX : LINK An, #DISPLACEMENT

EXAMPLE : LINK A6, # LOCAL PARAMETERS

FUNCTION : a) PUSH CURRENT CONTENT OF ADDRESS REGISTER ONTO THE STACK



A6 : 2020 2020

A6 : 10004 + DISPLACEM. (RES) + 4

RES : AFTER SECOND SUBROUTINE CALL

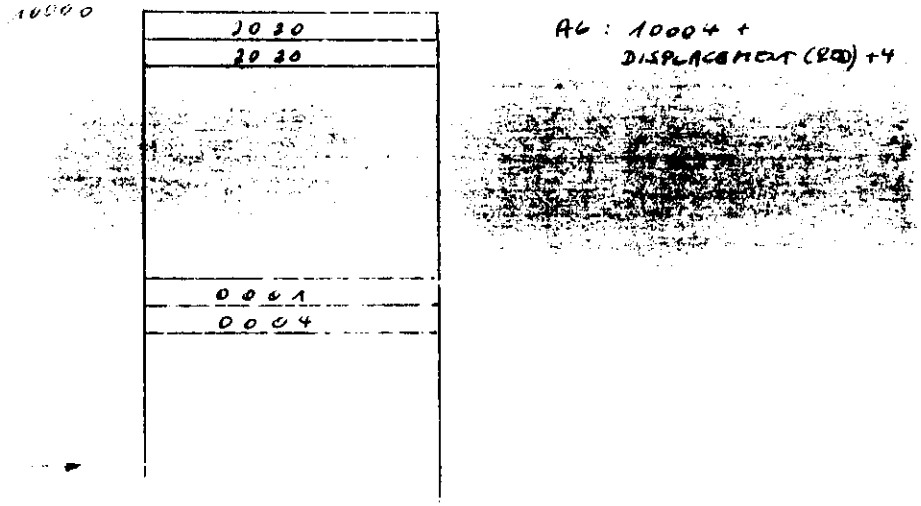
LINK IS EXECUTED AT ENTRY INTO THE SUBROUTINE TO ALLOCATE THE STACK SPACE NEEDED OF LOCAL VARIABLES

THE UNLK INSTRUCTION :

SYNTAX : UNLK A_n

EXAMPLE : UNLK A6

FUNCTION : THE STACK POINTER IS LOADED FROM THE SPECIFIED REGISTER. THE ADDRESS REGISTER IS THEN LOADED WITH THE LONGWORD PULLED OFF THE STACK

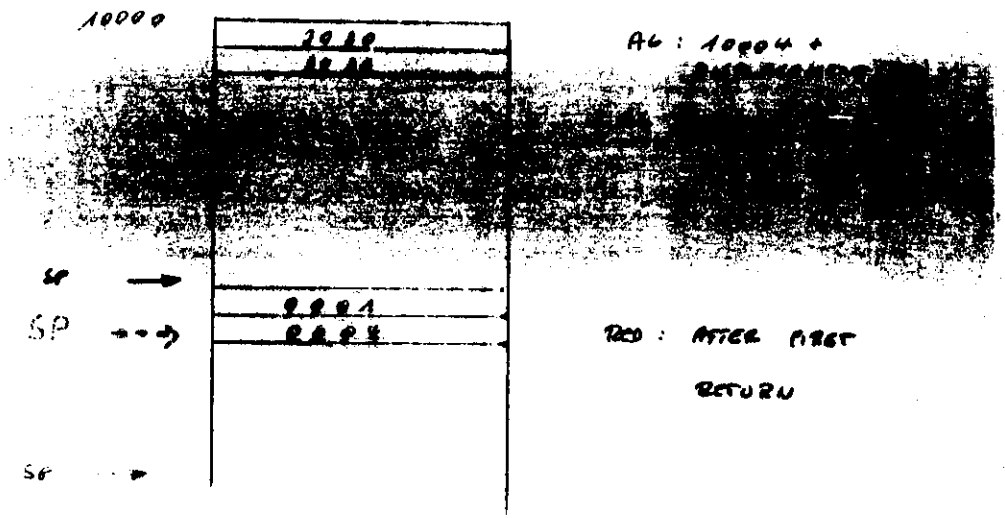


THE UNLK INSTRUCTION :

SYNTAX : UNLK A_n

EXAMPLE : UNLK A6

FUNCTION : THE STACK POINTER IS LOADED FROM THE SPECIFIED REGISTER. THE ADDRESS REGISTER IS THEN LOADED WITH THE LONGWORD PULLED OFF THE STACK

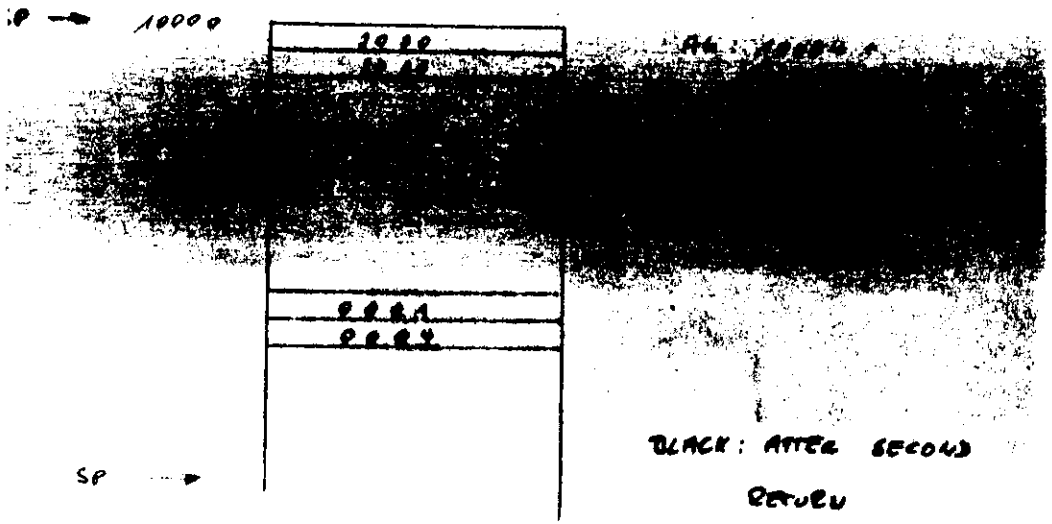


THE UNLINK INSTRUCTION :

SYNTAX : UNLK A_n

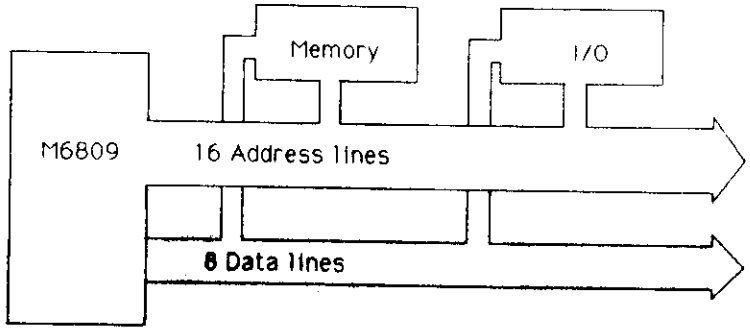
EXAMPLE : UNLK A6

FUNCTION : THE STACK POINTER IS LOADED FROM THE SPECIFIED REGISTER. THE ADDRESS REGISTER IS THEN LOADED WITH THE LONGWORD PULLED OFF THE STACK

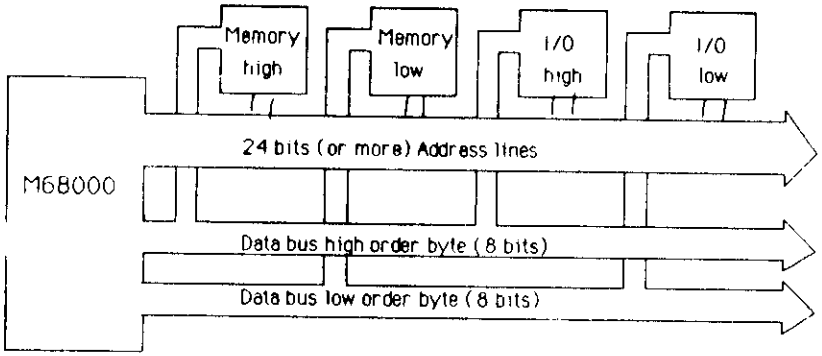


REMARK : THE STATE OF THE REGISTERS AFTER THE SECOND RETURN IS THE SAME AS BEFORE THE FIRST CALL

M6809



M68000

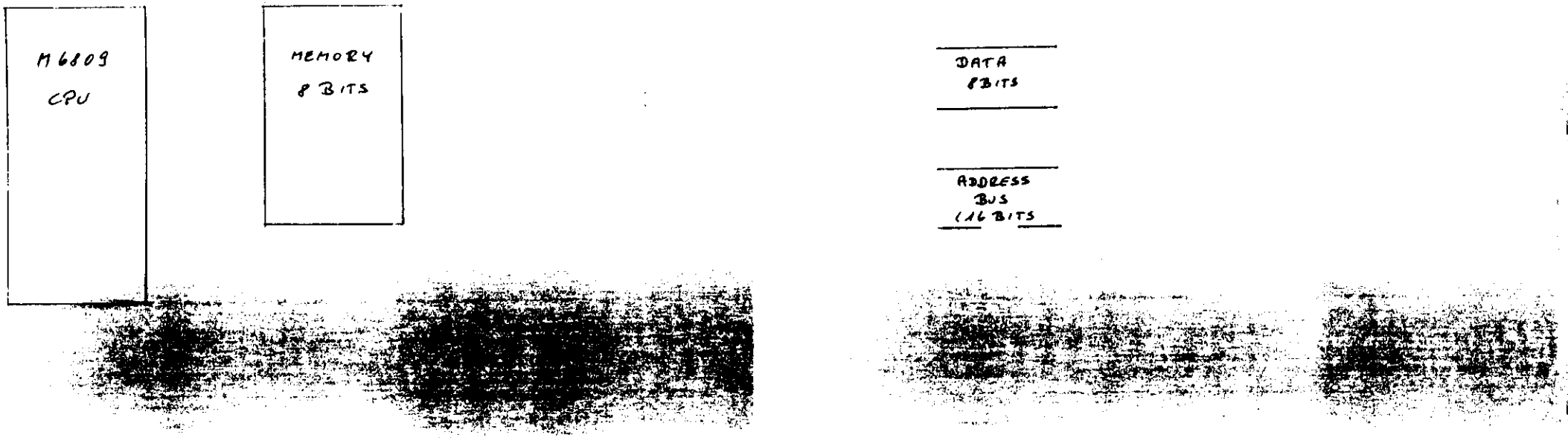


Conclusion:

16 Bit machine: Higher speed, bigger address range, more complexity, higher cost

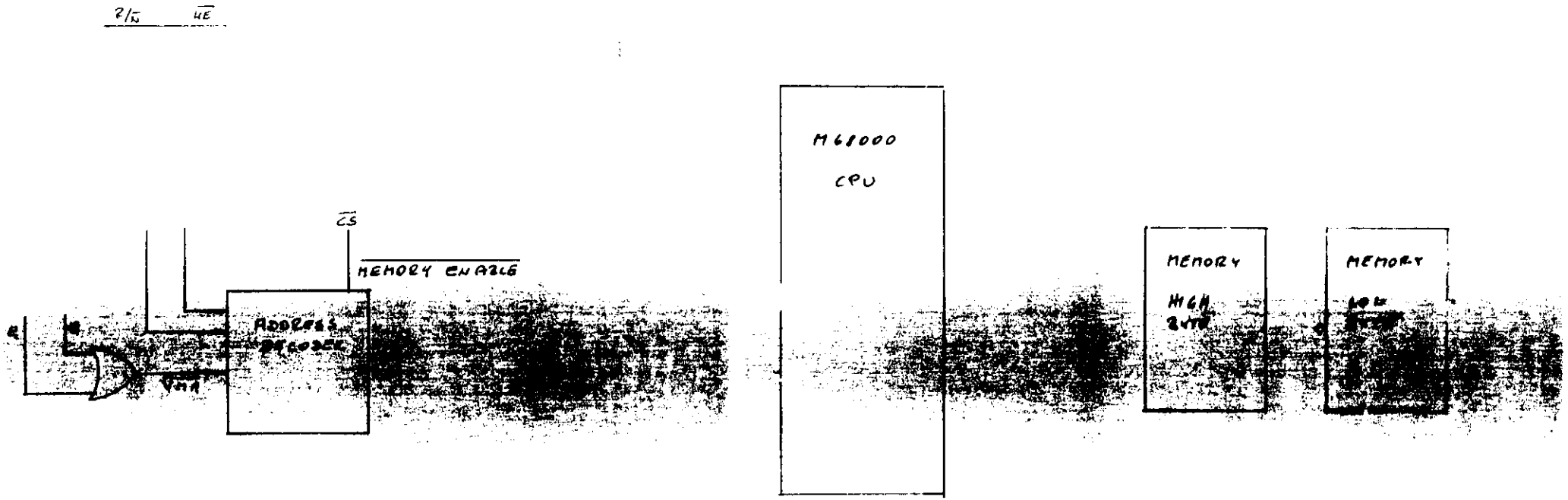
8 Bit machine: Lower speed, limited address range (but suffucient for most applications) less complex and thus less difficult to

INTERFACING THE 6809 TO MEMORY



INTERFACING THE M68000
TO EXTERNAL MEMORY

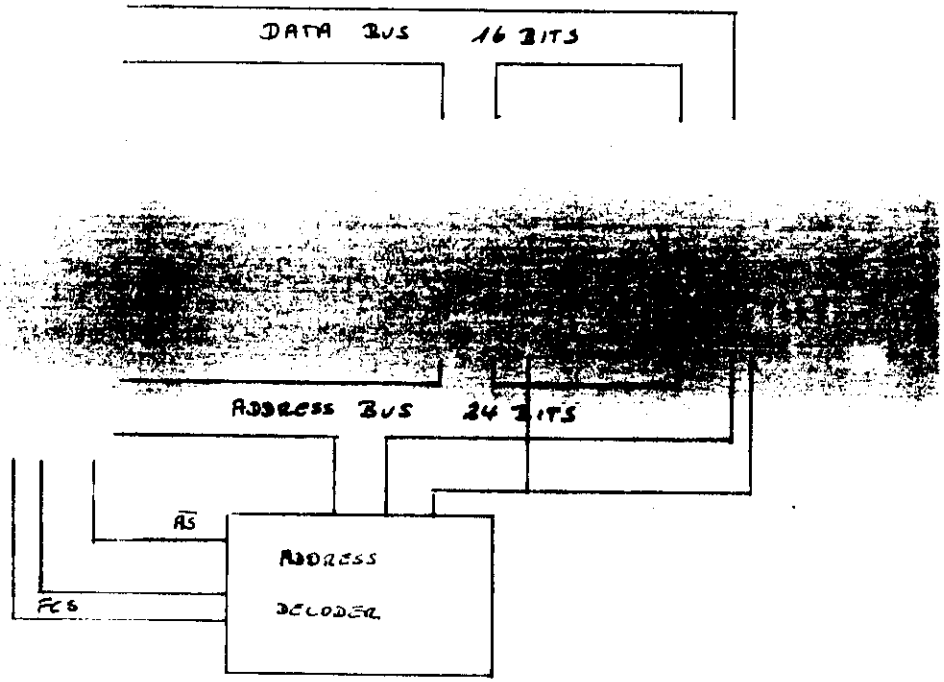
(SYNCHRONOUS BUS)



WHAT CAN BE DONE IF THE MEMORY IS TOO SLOW?

- BUY FASTER (MORE EXPENSIVE) MEMORY
- CHANGE SYSTEM CLOCK FREQUENCY
  (DEGRADES SYSTEM PERFORMANCE)

IN A WORKING SYSTEM REPLACEMENT OF SLOW MEMORY BY HIGH SPEED MEMORY DOES <u>NOT</u> IMPROVE SYSTEM PERFORMANCE !

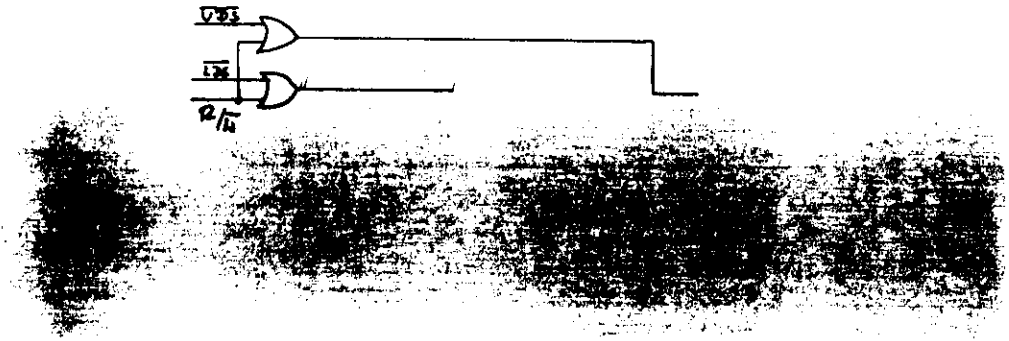


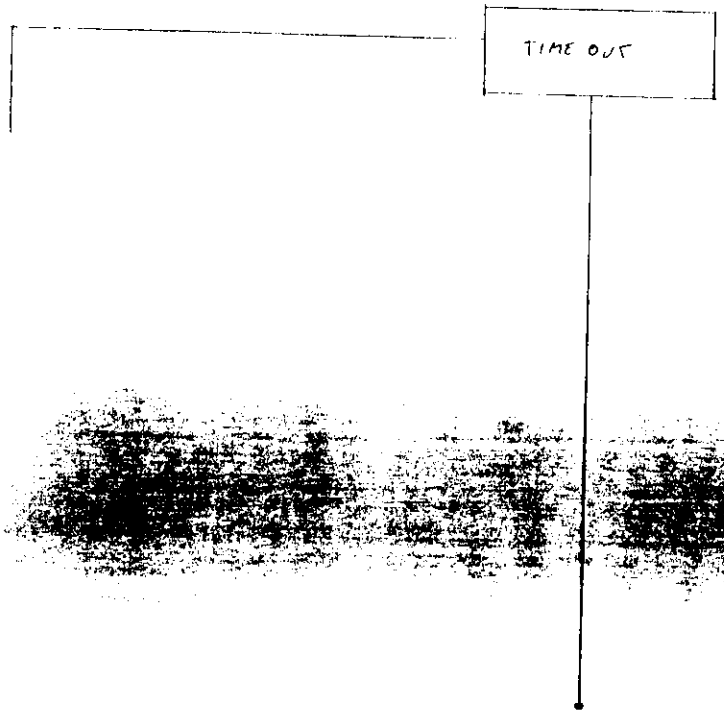
R/W

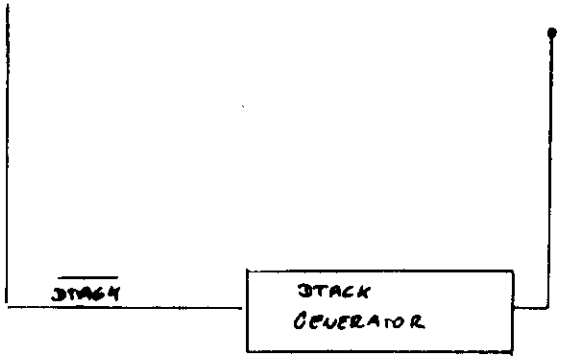
MOVE.W D0, MEM OK !

BUT ...

MOVE.B D0, MEM DESTROYS VALID DATA !





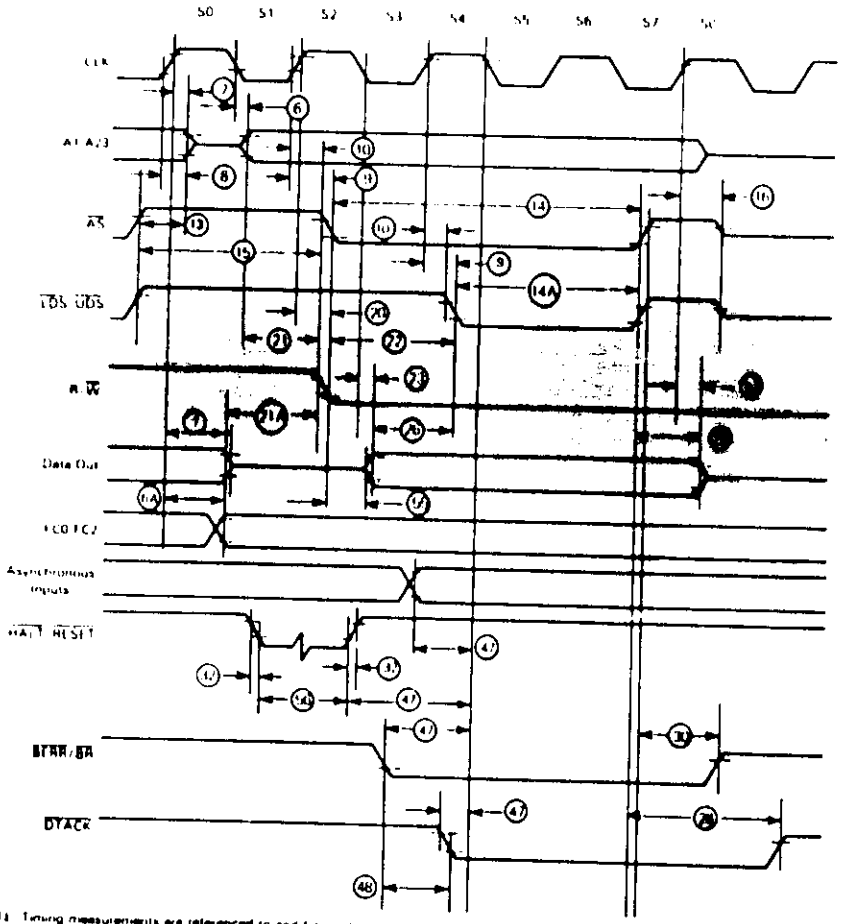


WHAT HAPPENS IF THE MEMORY ADDRESS IS WRONG ?

MOVE.W D0, INVALID

NO $\overline{\text{DTACK}}$ IS GENERATED => MACHINE HANGS

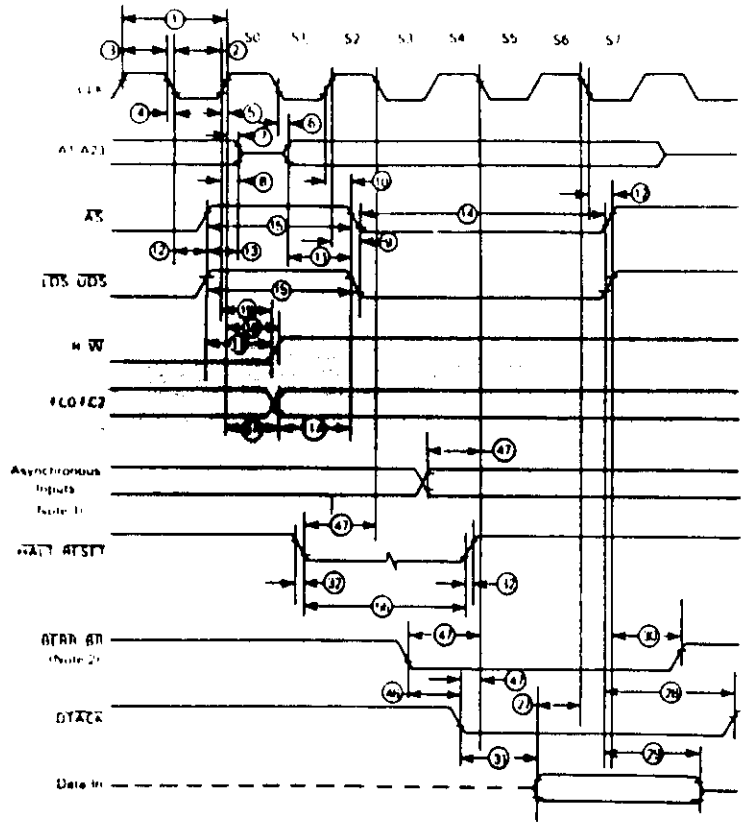
FIGURE 6 – WRITE CYCLE TIMING



NOTE: 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.

MC68000L4•MC68000L6•MC68000L8•MC68000L10

FIGURE 5 – READ CYCLE TIMING



NOTES

1. Setup time for the asynchronous inputs BGACK, IPL0, IPL1, 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 the 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.

AC ELECTRICAL SPECIFICATIONS V_{CC} = 5.0 Vdc ±5%, V_{SS} = 0 Vdc, T_A = 0°C to 70°C, See Figures 5 and 6

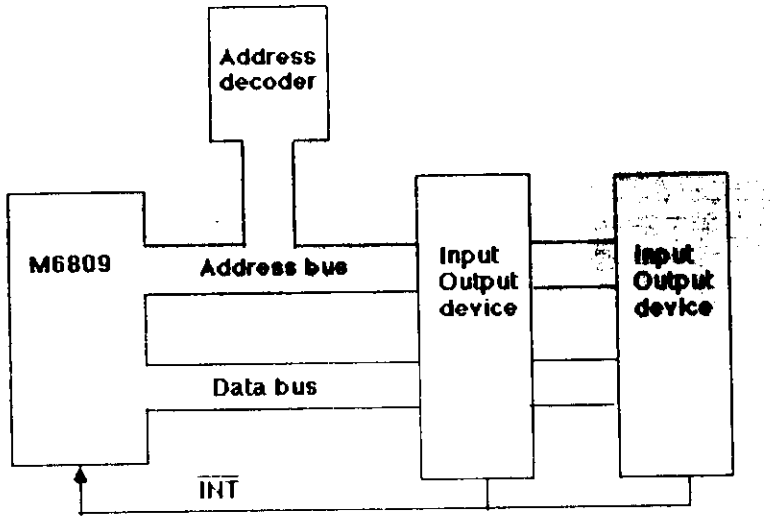
Number	Characteristic	Symbol	4 MHz MC68000L4		6 MHz MC68000L6		8 MHz MC68000L8		10 MHz MC68000L10		Unit
			Min	Max	Min	Max	Min	Max	Min	Max	
1	Clock Period	t_{cyc}	250	500	167	500	125	500	100	500	ns
2	Clock Width Low	t_{CL}	115	250	75	250	55	250	45	250	ns
3	Clock Width High	t_{CH}	115	250	75	250	55	250	45	250	ns
4	Clock Fall Time	t_{Cf}		10		10	–	10		10	ns
5	Clock Rise Time	t_{Cr}		10		10	–	10		10	ns
6	Clock Low to Address	t_{CLAV}		90		80	–	70		60	ns
6A	Clock High to FC Valid	t_{CHFCV}		90		80	–	70		60	ns
7	Clock High to Address Data High Impedance (Maximum)	t_{CHAZx}		120		100		80		70	ns
8	Clock High to Address FC Invalid (Minimum)	t_{CHAZn}	0		0		0		0		ns
9	Clock High to AS, DS Low (Maximum)	t_{CHSLx}		80		70		60		50	ns
10	Clock High to AS, DS Low (Minimum)	t_{CHSLn}	0	–	0	–	0	–	0	–	ns
11	Address to AS, DS (Read) Low, AS Write	t_{AVSL}	50		30	–	30		20		ns
11A	FC Valid to AS, DS (Read) Low, AS Write	t_{FCVSL}	80		70	–	60		50		ns
12	Clock Low to AS, DS High	t_{CLSH}		90	–	80	–	70		60	ns
13	AS, DS High to Address FC Invalid	t_{SHAZ}	60	–	40		30		20		ns
14	AS, DS Width Low (Read) AS Write	t_{SL}	530	–	337	–	240		195		ns
14A	DS Width Low (Write)	–	280	–	170	–	115	–	90		ns
15	AS, DS Width High	t_{SH}	280		180	–	150		100		ns
16	Clock High to AS, DS High Impedance	t_{CHSZ}		120	–	100	–	80		70	ns
17	AS, DS High to R/W High	t_{SHRH}	60	–	50		40	–	20	–	ns
18	Clock High to R/W High (Maximum)	t_{CHRHx}	–	80	–	80	–	70		60	ns
19	Clock High to R/W High (Minimum)	t_{CHRHn}	0	–	0		0	–	0		ns
20	Clock High to R/W Low	t_{CHRL}		90	–	80		70		60	ns
21	Address Valid to R/W Low	t_{AVRL}	45	–	25		20		0		ns
21A	FC Valid to R/W Low	t_{FCVRL}	80	–	70	–	60		50		ns
22	R/W Low to DS Low (Write)	t_{RLSL}	200	–	140		80		50		ns
23	Clock Low to Data Out Valid	t_{CLDO}		80	–	80		70		50	ns
24	DS High to Data Out Invalid	t_{SHDO}	60		40		30		20		ns
24A	Data Out Valid to DS Low (Write)	t_{DOSL}	50		30		30		20		ns
25	Data In to Clock Low (Setup Time)	t_{DICL}	30		20		15		10		ns
26	AS, DS High to DTACK High	t_{SHDAH}	0	240	0	180	0	120	0	80	ns
27	DS High to Data Invalid (Hold Time)	t_{SHDI}	0		0	–	0		0		ns
28	AS, DS High to BERR High	t_{SHBEH}	0		0		0		0		ns
29	DTACK Low to Data In (Setup Time)	t_{DALDI}		150		120		90		60	ns
30	HALT and RESET Input Transition Time	t_{RHr}	0	200	0	200	0	200	0	200	ns
31	Clock High to BG Low	t_{CHGL}	–	90	–	80	–	70	–	60	ns
32	Clock High to BG High	t_{CHGH}		90	–	80	–	70	–	60	ns
33	BR Low to BG Low	t_{BRLGL}	1.5	3.0	1.5	3.0	1.5	3.0	1.5	3.0	Clk Per
34	BR High to BG High	t_{BRHGH}	1.5	3.0	1.5	3.0	1.5	3.0	1.5	3.0	Clk Per
37	BGACK Low to BG High	t_{GALGH}	1.5	3.0	1.5	3.0	1.5	3.0	1.5	3.0	Clk Per
38	BG Low to Bus High Impedance (With AS High)	t_{GLZ}	–	120	–	100	–	80	–	70	ns
39	BG Width High	t_{GH}	1.5	–	1.5	–	1.5	–	1.5	–	Clk Per
40	BGACK Width	t_{BGI}	1.5		1.5		1.5	–	1.5	–	Clk Per
47A	Asynchronous Input Setup Time	t_{ASI}	30		25	–	20	–	20	–	ns
48	BERR Low to DTACK Low (Note 3)	t_{BELDAL}	60	–	60	–	60	–	60	–	ns
53	Data Hold from Clock High	t_{CHDO}	0	–	0	–	0	–	0	–	ns
55	R/W to Data Bus Impedance Change	t_{RLDO}	55		30	–	30	–	20	–	ns
56	Halt/RESET Pulse Width (Note 4)	t_{HRPW}	10	–	10	–	10	–	10	–	Clk Per

NOTES

1. For a loading capacitance of less than or equal to 500 picofarads, subtract 5 nanoseconds from the values given in these columns.
2. Actual value depends on clock period.
3. If #47 is satisfied for both DTACK and BERR, #48 may be 0 ns.
4. After V_{CC} has been applied for 100 ms.
5. For "-0" #14 and #14A, make sets #14 and #14A are one clock period less than the given number.
6. If the asynchronous setup time (#47) requirements are satisfied, the DTACK low to data setup time (#31) requirement can be ignored. The data must only satisfy the data in to clock low setup time (#27) for the following cycle.

Interrupts

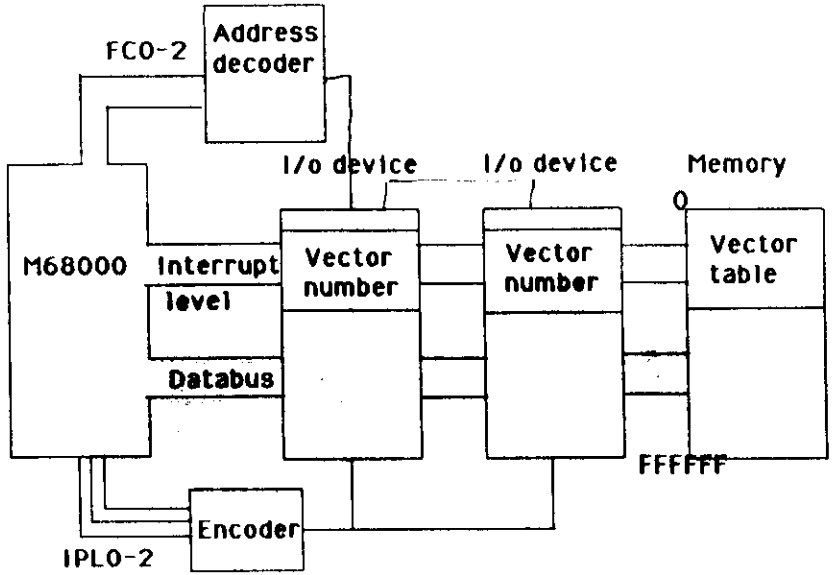
M6809



When an interrupt arrives:

- CPU saves machine state after end of current instruction
- CPU picks up start address of interrupt service routine at fixed memory address
- CPU exeutes routine at this address
- 3 levels of priority: FIRQ, IRQ, NMI
- Polling is necessary if more than 1 device is used on same interrupt line

M68000 Vectored interrupts



M68000 Interrupt sequence

CPU | I/O device

Request interrupt

↓

Grant interrupt

1.) Compare to current priority level and complete instruction
2.) Place interrupt level on A1-A3
3.) Set R/W to read
4.) Set FC0-2 to IACK
5.) Assert AS
6.) Assert LDS

↓

Provide vector number

1.) Place vector number on D0-D7
2.) Assert DTACK

↓

Acquire vector number

1.) Latch vector number
2.) Negate LDS
3.) negate AS

↓

Release

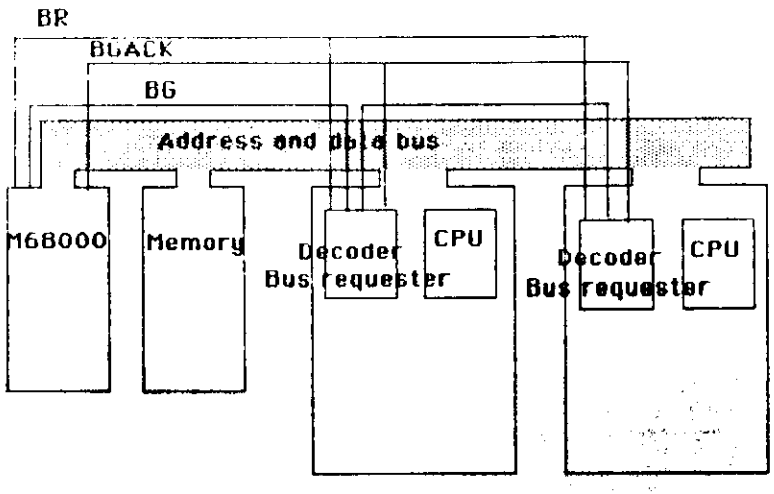
1.) Negate DTACK

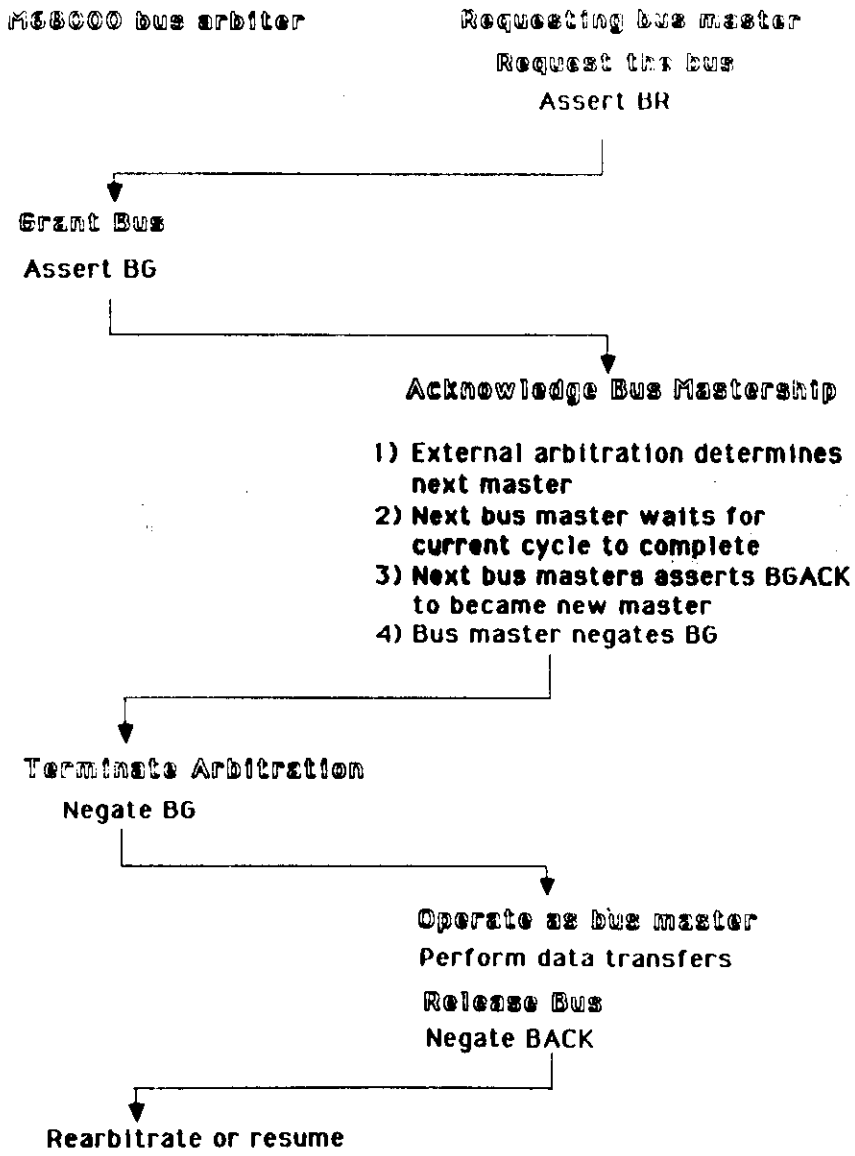
M68000 function code bits

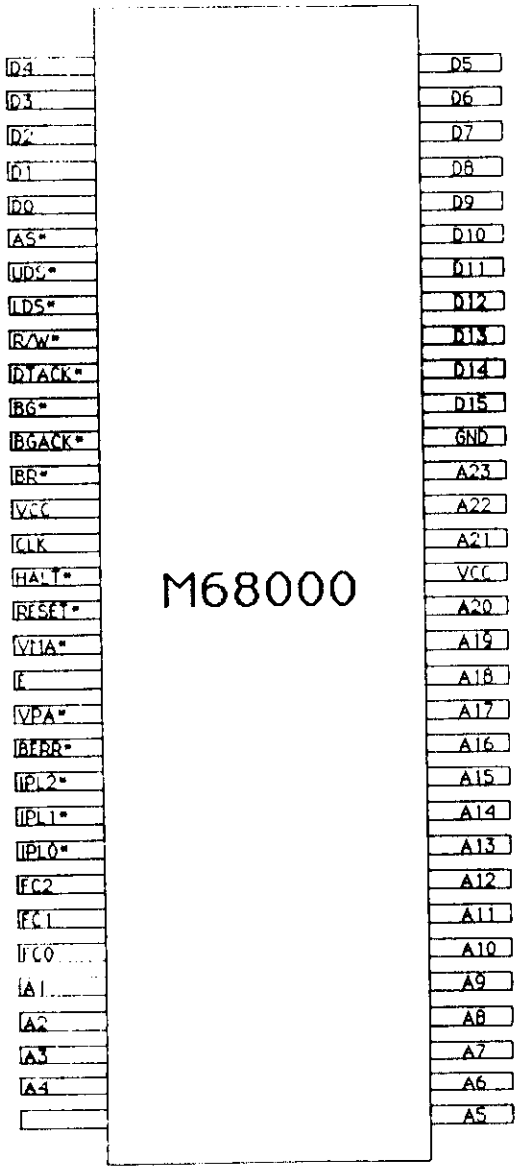
FC2	FC1	FC0	Cycle type
0	0	0	Undefined (reserved)
0	0	1	User data space
0	1	0	User program space
0	1	1	Undefined (reserved)
1	0	0	Undefined reserved
1	0	1	Supervisor data space
1	1	0	Supervisor program space
1	1	1	CPU space

The function code bits may be used to implement (in hardware) memory protection features. Important: Make access to system resources impossible for user programs. To access system resources they must call the operating system (running in supervisor mode).

Interfacing more than one master on the M68000 bus







Pin Number	Function
A1	BGACK*
A2	A1
A3	A30
A4	A29
A5	A28
A6	A23
A7	A22
A8	A19
A9	V_{CC}
A10	GND
A11	A14
A12	A11
A13	A8
B1	GND
B2	BG
B3	BR
B4	A31
B5	A27
B6	A24
B7	A20
B8	A18
B9	GND
B10	A15
B11	A13
B12	A10
B13	A6
C1	RESET
C2	CLOCK
C3	GND
C4	A0
C5	A29
C6	A25
C7	A21
C8	A17
C9	A16
C10	A12
C11	A9
C12	A7
C13	A5

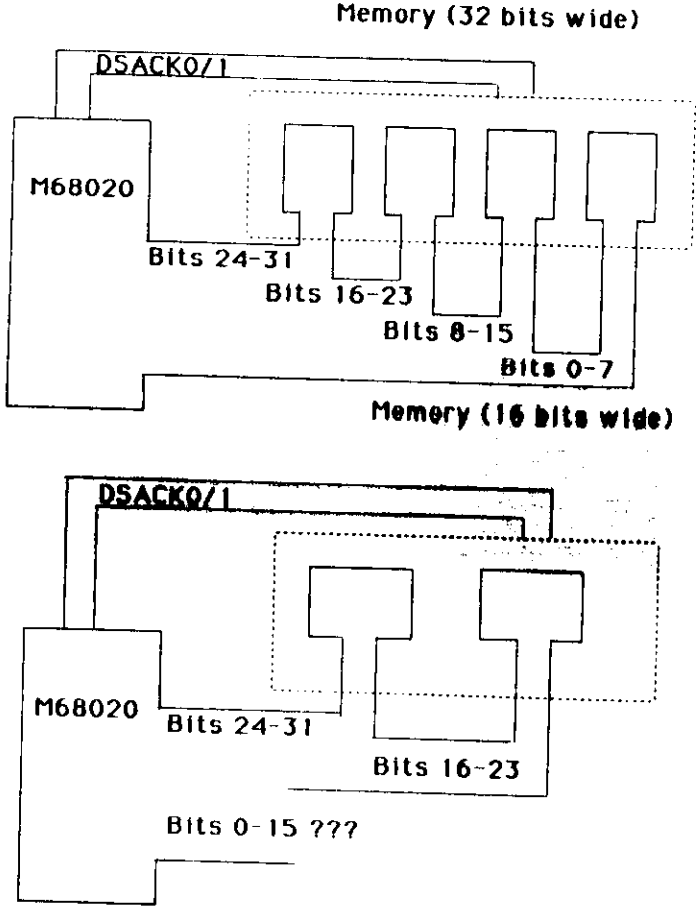
Pin Number	Function
D1	V_{CC}
D2	V_{CC}
D3	V_{CC}
D4 D11	
D12	A4
D13	A3
E1	E
E2	VMA
E3	V_{CC}
E12	A2
E13	LDS
F1	SIZ0
F2	FC2
F3	FC1
F12	GND
F13	BERR
G1	ECS
G2	SIZ1
G3	OBEN
G11	V_{CC}
G12	GND
G13	V_{CC}
H1	CDIS
H2	AVEC
H3	DSACK0
H12	IPL2
H13	GND
J1	DSACK1
J2	BERR
J3	GND
J12	IPL0
J13	IPL1

Pin Number	Function
K1	GND
K2	HALT
K3	GND
K12	D1
K13	D0
L1	AS
L2	R/W
L3	D30
L4	D27
L5	D23
L6	D19
L7	GND
L8	D15
L9	D11
L10	D7
L11	GND
L12	D3
L13	D2
M1	DS
M2	D29
M3	D26
M4	D24
M5	D21
M6	D18
M7	D16
M8	V_{CC}
M9	D13
M10	D10
M11	D6
M12	D5
M13	D4
N1	D31
N2	D28
N3	D25
N4	D22
N5	D20
N6	D17
N7	GND
N8	V_{CC}
N9	D14
N10	D12
N11	D9
N12	D8
N13	V_{CC}

The VCC and GND pins are separated into three groups to provide individual power supply connections for the address bus buffers, data bus buffers, and all other output buffers and internal logic.

Group	V_{CC}	GND
Address Bus	A9 D3	A10 B9 C3 F12
Data Bus	M8 N8 N13	L7 L11 N7 K3
Logic	D1 D2 E3 G11 G13	G12 H13 J3 K1

M68020 dynamic bus sizing



This works because memory signals not only end of memory cycle (DTACK) but also tranfer size

DSACK0	DSACK1	
1	1	Wait
1	0	8 bit transferred
0	1	16 bit transferred
0	0	32 bits transferred

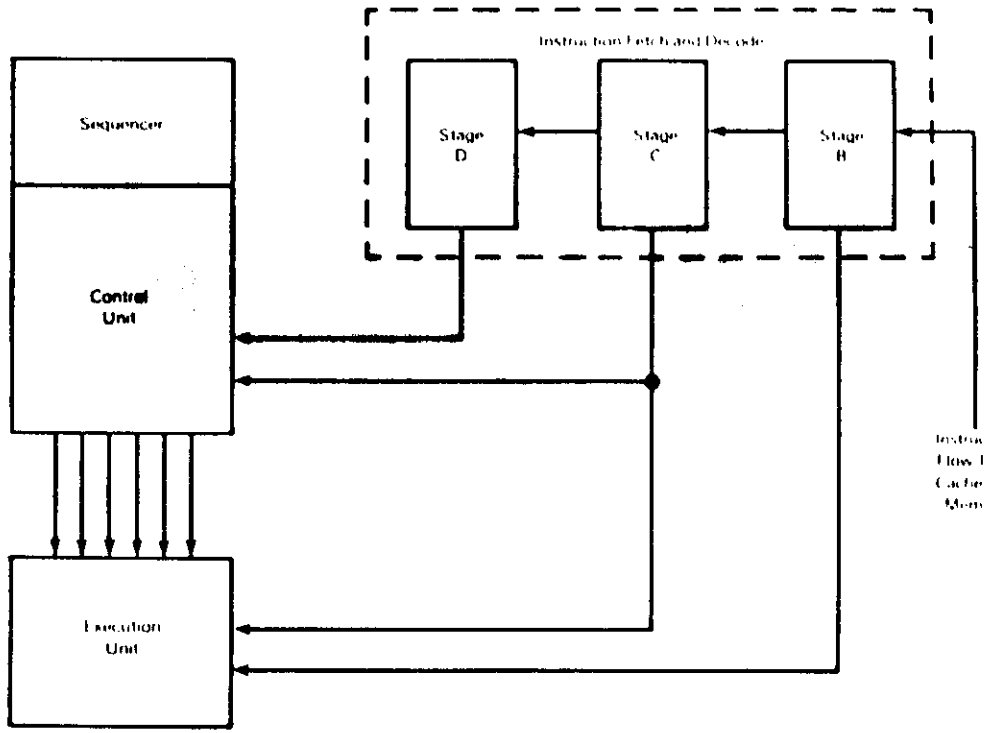
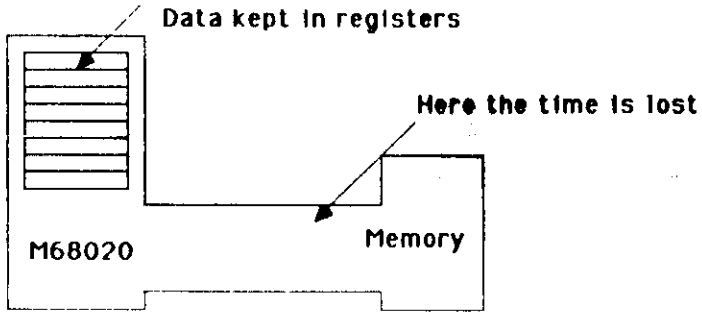


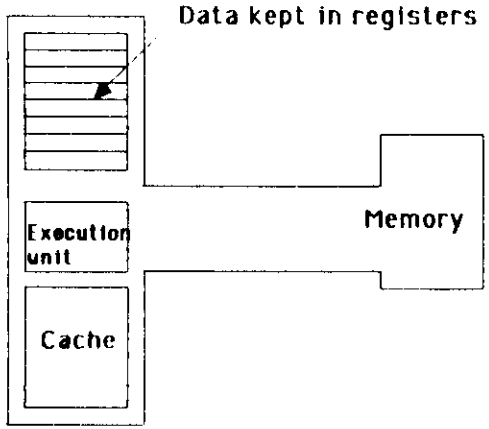
Figure 1-5. MC68020 Pipeline

The M68020 instruction cache

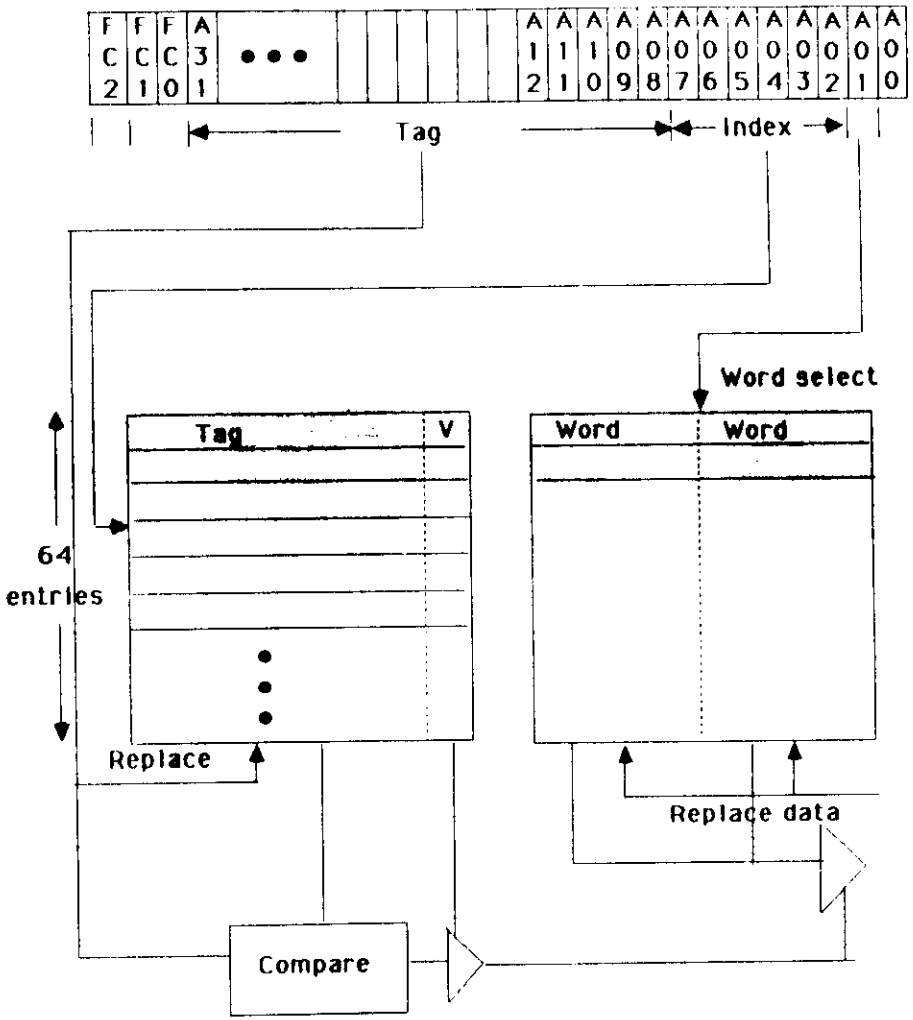
- Most time is lost in memory cycles because the chip has to send signals off and acquire signals external to the chip.
- Data may be kept in registers. Instruction codes may not



- Most time in a program is spend in small loops



How the cache works

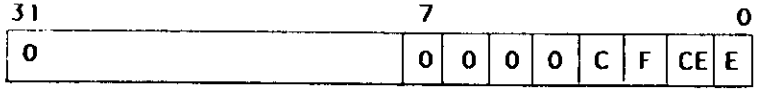


- When at the index generated by A2-A7 a tag word identical to A8-A31 is found, the FC2 bit coincides with the FC2 bit in ths cache tag field and the V bit is set to true then a cache hit occurs

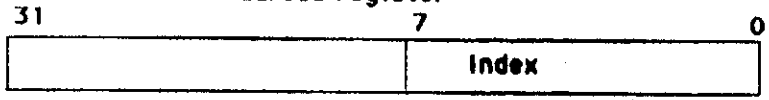
Some special instructions fo cache control

- Cache control ie executed bye writing data into the cache control register

Cache control register



Cache address register

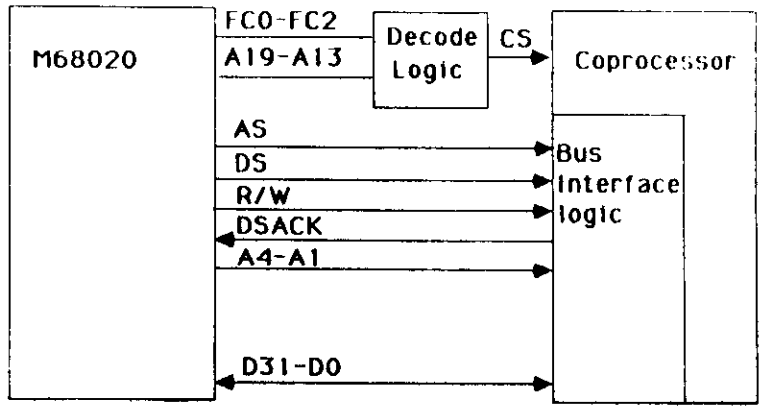


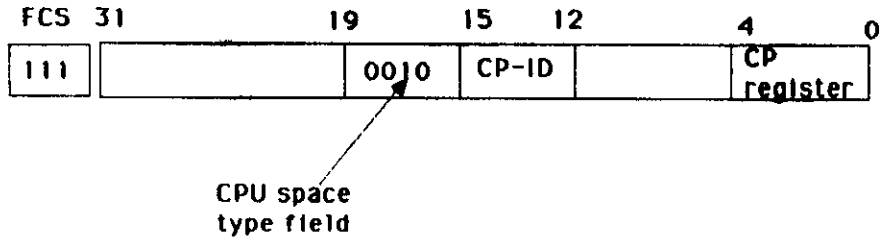
Access to these registers is given be the special instruction: MOVEC Rn,Rc
MOVEC Rc,Rn

The Rc then defines CAAR (Cache address register)
or the CACR (Cache control register)

- E Enable cache
- F Freeze cache
- CE Clear entry
- C Clear cache

The M68020 coprocessor interface





- In CPU space at $20000-$2001F registers of first coprocessor
- Up to 8 coprocessors possible
- CP-IDs
  - 000 MC68851 PMMU
  - 001 MC68881 Floating point coprocessor

The M68020 coprocessor interface

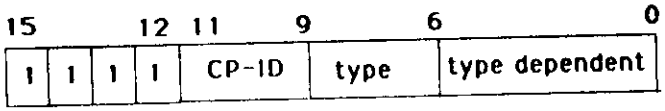
- Specialized coprocessors may still augment the M68020 processing power

  Typical examples: Floating point coprocessor
  Memory management unit (MMU)
  Graphics coprocessor

  at Cern: Fastbus coprocessor

- a coprocessor ins **not** an I/O chip !!!
- **a coprocessor has a special interface**
- **a coprocessor extends the M68020 instruction set**

Coprocessor instructions:



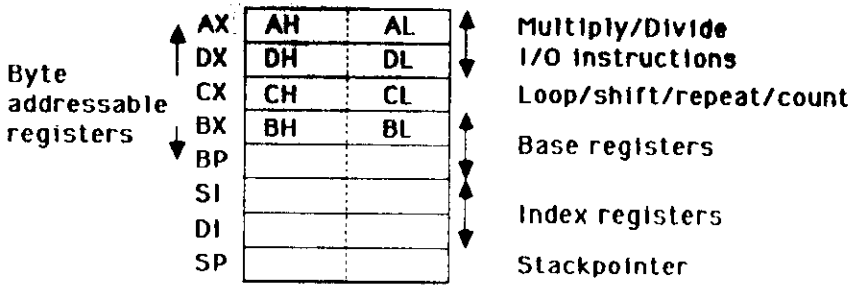
- $F in the high order 4 bits of an M68020 instruction mean: instructions to be executed by a coprocessor

- If no coprocessor with identifier CP-ID is available the M68020 (as the M68000) generates a trap (SWI). Like this coprocessor instructions can be emulated in software.

Some aspects of the Intel 80286

- Intel's high end 16 bit microprocessor
- 16 data lines, 24 address lines multiplexed
- segmented memory
- optimized capabilities for multitasking systems

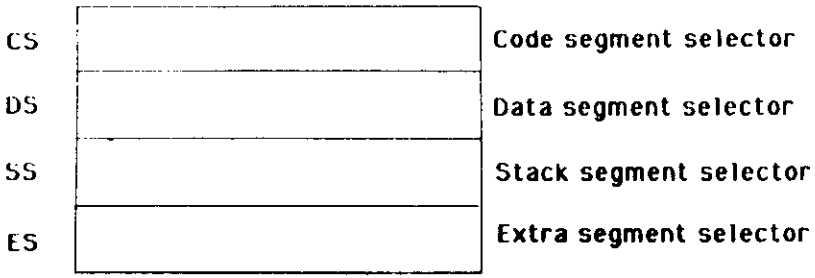
The programmer's model



Conclusion:

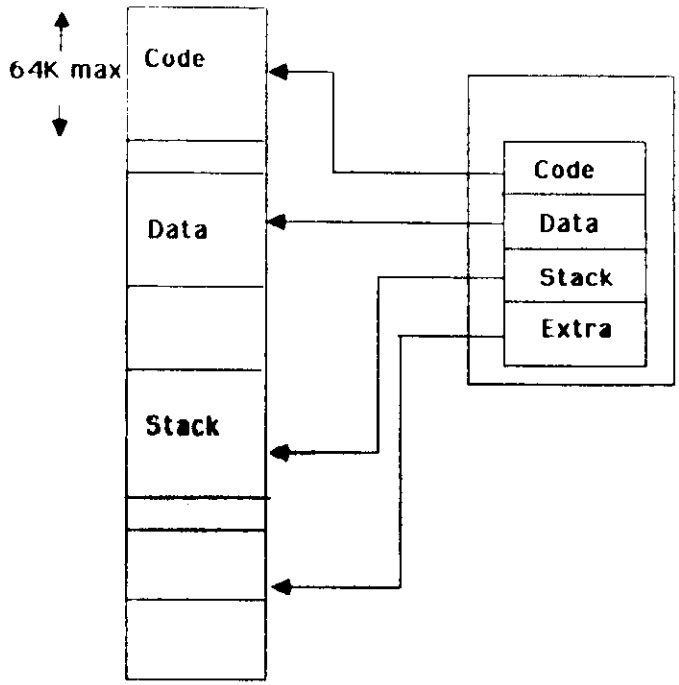
Less registers than the M68000 and register use is defined. Registers are 16 (not 32) bits wide.

- In addition to the general purpose registers we find 4 segment registers:



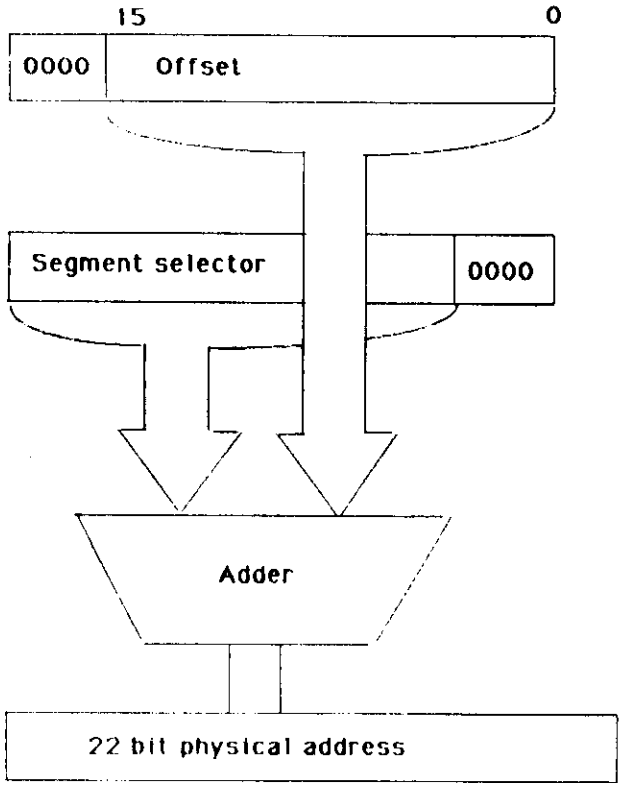
The segment registers are use automatically as follows:

Instructions	Code (CS)	Automatic with prefetch
Stack:	Stack (SS)	All stack push and pops
Local Data	Data (DS)	All data references exept stack and strings
External Data	Extra (ES)	Alternate data segment and sting operations



- In real address mode the 80286 offers an address range of 1Mbyte (20 address lines).
  However to exeed 64kbytes the corresponding segment register has to be loaded.
  (This correponds to a memory resident overlay)

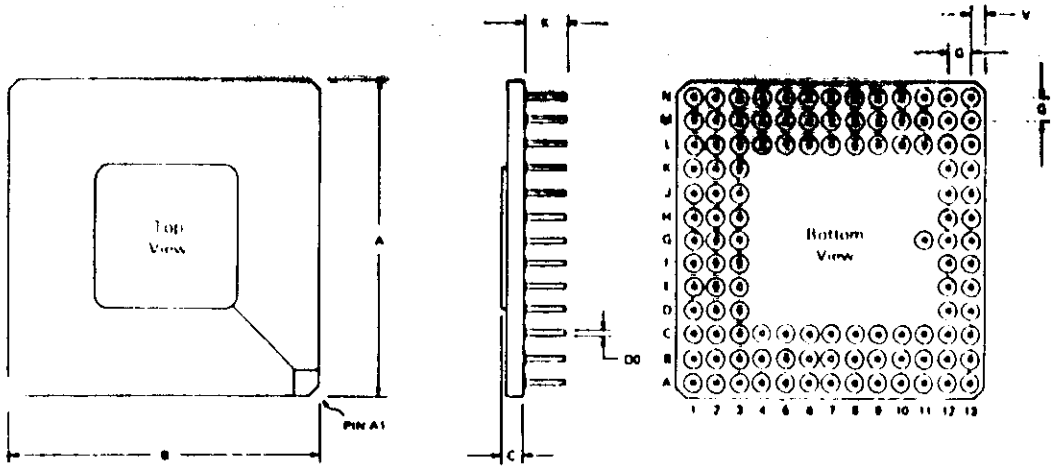
Address generation in real address mode



11.2 PACKAGE DIMENSIONS AND PIN ASSIGNMENT

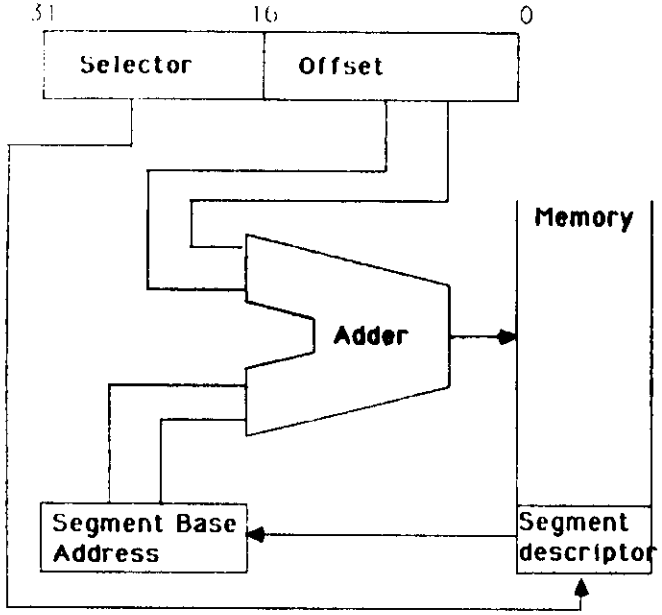
MC68020
RC Suffix Package
Preliminary
Mechanical
Detail

DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	34.18	34.90	1.345	1.375
B	34.18	34.80	1.345	1.375
C	2.87	3.17	.100	.150
DD	.46	.51	.017	.019
G	2.54 BSC		.100 BSC	
K	4.37	4.87	.170	.190
V	1.74	2.28	.065	.095



The 80286 protected virtual mode

- Used in multitasking systems to protect a tasks address space from access by other tasks



The descriptor is loaded from memory into a descriptor cache when the corresponding segment register is changed

Reserved	
Access rights	Base 23-16
Base 15-0	
Limit 15-0	

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Order this technical summary by BRE508/D

CROSETTO D.

MC68030

Technical Summary

Second Generation 32-Bit Enhanced Microprocessor

The MC68030 is the industry's second generation 32-bit enhanced microprocessor. The MC68030 is a virtual memory microprocessor based on an MC68020 core with additional enhanced performance features. Increased internal parallelism is provided by multiple internal data buses and address buses and a versatile bus controller that supports two-clock cycle bus accesses and one-clock cycle burst accesses in order to maximize performance with paged mode, nibble mode, and static column DRAM technology. A 256-byte on-chip instruction cache in addition to a 256-byte data cache improves data flow to the execution unit and further boosts performance. On-chip paged memory management reduces the minimum physical bus cycle time to two clocks, and provides zero translation time to any bus cycle. The paged memory management structure can be enabled/disabled by software for applications not requiring the memory management feature. The rich instruction set and addressing modes of the MC68020 have been maintained allowing a clear migration path for M68000 systems.

The main features of the MC68030 are:

- Object Code Compatible with the MC68020 and Earlier M68000 Microprocessors
- Complete 32-Bit Non-Multiplexed Address and Data Buses
- Sixteen 32-Bit General Purpose Data and Address Registers
- Two 32-Bit Supervisor Stack Pointers and 10 Special Purpose Control Registers
- 256-Byte Instruction Cache and 256-Byte Data Cache that can be Accessed Simultaneously
- Paged Memory Management Unit that Translates Addresses in Parallel with Instruction Execution
- Two Transparent Segments Allow Untranslated Blocks to be Defined for Systems that Transfer Large Blocks of Data to Predefined Addresses, e.g., Graphics Applications
- Pipelined Architecture with Increased Parallelism Allows Accesses from Internal Caches to Occur in Parallel with Bus Transfers and Multiple Instructions to be Executing Concurrently
- Enhanced Bus Controller Supports Asynchronous Bus Cycles, Synchronous Bus Cycles that can Operate in Two Clocks, and Burst Data Transfers that can Operate in One Clock, all with Physical Addresses
- Dynamic Bus Sizing Supports 8-/16-/32-Bit Memories and Peripherals
- Complete Support for Coprocessors with the M68000 Coprocessor Interface
- 4-Gigabyte Direct Addressing Range
- Implemented in Motorola's HCMOS Technology that Allows CMOS and HMOS (High Density NMOS) Gates to be Combined for Maximum Speed, Low Power, and Small Die Size
- Selection of Processors Speeds: 16.67 and 20 MHz

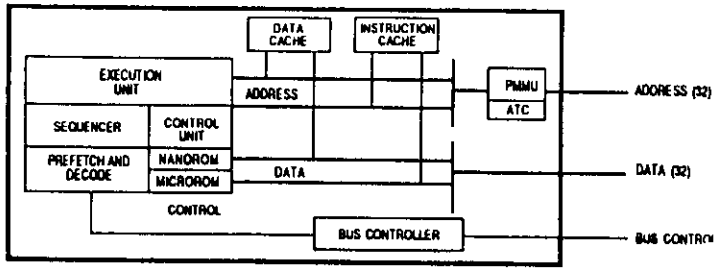


Figure 1. MC68030 Block Diagram

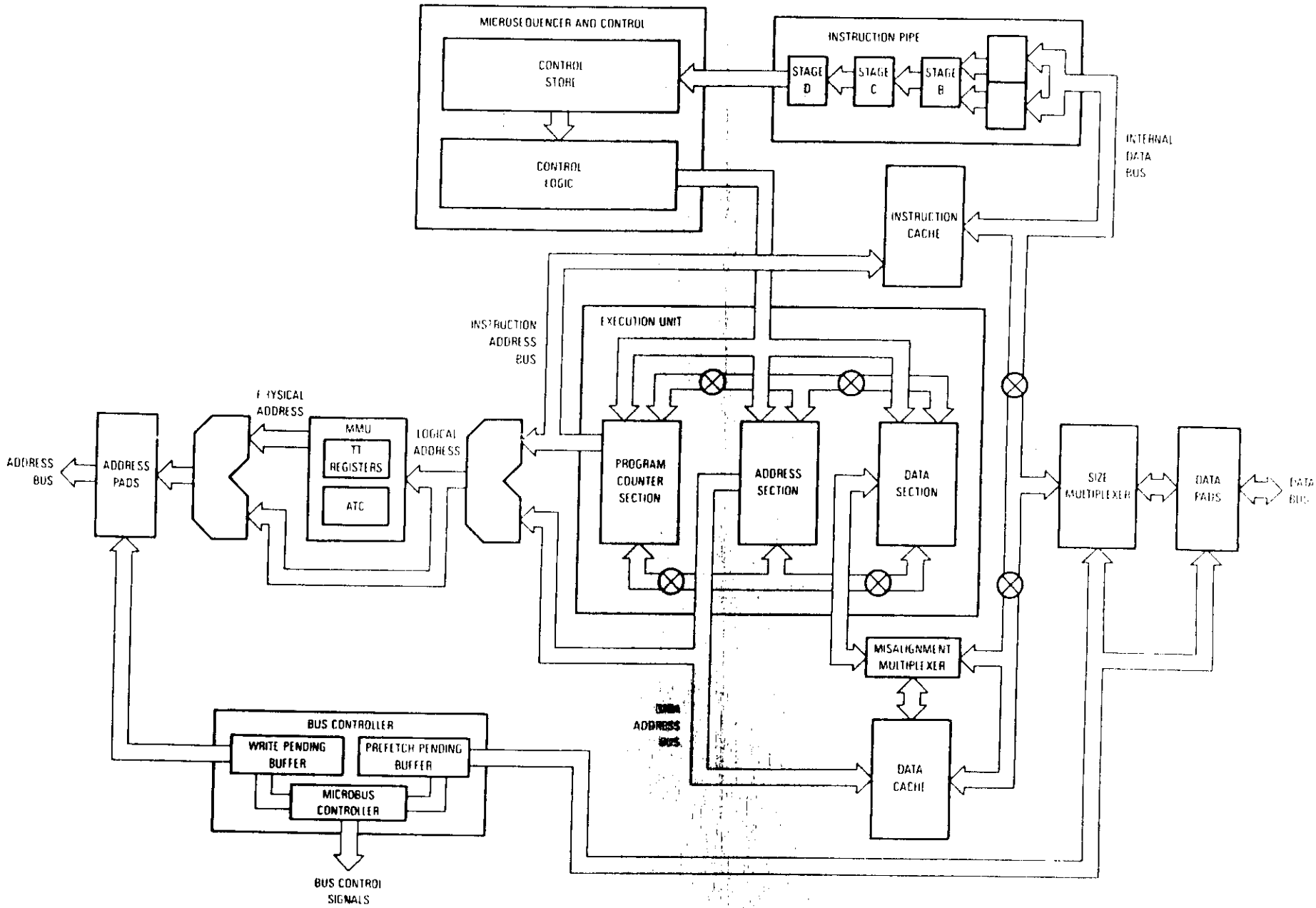


Figure 1-1. MC68030 Block Diagram

80386
HIGH PERFORMANCE MICROPROCESSOR WITH INTEGRATED MEMORY MANAGEMENT

- **Flexible 32-Bit Microprocessor**
  - **— 8, 16, 32-Bit Data Types**
  - **— 8 General Purpose 32-Bit Registers**
- **Very Large Address Space**
  - **— 4 Gigabyte Physical**
  - **— 64 Terabyte Virtual**
  - **— 4 Gigabyte Maximum Segment Size**
- **Integrated Memory Management Unit**
  - **— Virtual Memory Support**
  - **— Optional On-Chip Paging**
  - **— 4 Levels of Protection**
  - **— Fully Compatible with 80286**
- **Object Code Compatible with All 8086 Family Microprocessors**
- **Virtual 8086 Mode Allows Running of 8086 Software in a Protected and Paged System**
- **Hardware Debugging Support**
- **Optimized for System Performance**
  - **— Pipelined Instruction Execution**
  - **— On-Chip Address Translation Caches**
  - **— 12.5 and 16 MHz Clock**
  - **— 32 Megabytes/Sec Bus Bandwidth**
- **High Speed Numerics Support via 80287 and 80387 Coprocessors**
- **Complete System Development Support**
  - **— Software: C, PL/M, Assembler System Generation Tools**
  - **— Debuggers: PSCOPE, ICE™-386**
- **High Speed CHMOS III Technology**
- **132 Pin Grid Array Package**

(See Packaging Specification, Order #231369)

The 80386 is an advanced 32-bit microprocessor designed for applications needing very high performanc and optimized for multitasking operating systems. The 32-bit registers and data paths support 32-bit addresse and data types. The processor addresses up to four gigabytes of physical memory and 64 terabytes (2**46) c virtual memory. The integrated memory management and protection architecture includes address translatio registers, advanced multitasking hardware and a protection mechanism to support operating systems. I addition, the 80386 allows the simultaneous running of multiple operating systems.

Instruction pipelining, on-chip address translation, a a high bus bandwidth ensure short average instructio execution times and high system throughput. The 80386 processor is capable of execution at sustained rate of between 3 and 4 million instructions per second.

The 80386 offers new testability and debugging features. Testability features include a self-test and direc access to the page translation cache. Four new breakpoint registers allow conditional or unconditional breal point traps on code execution or data accesses, for powerful debugging of even ROM-based systems.

Object-code compatibility with all iAPX 86 family members (8086, 8088, 80186, 80188, 80286) means th 80386 offers immediate access to the world's largest microprocesor software base.

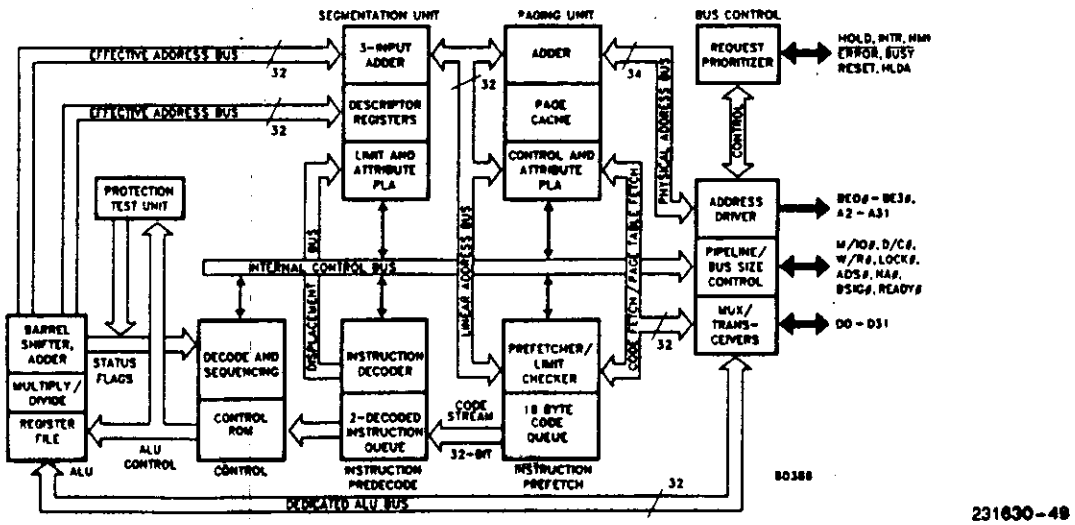


Figure 1-1. 80386 Pipelined 32-Bit Microarchitecture

Unix™ is a Trademark of AT&T Bell Labs.
MS-DOS is a Trademark of MicroSoft Corporation.

October 198

Order Number: 231630-00

Order this document
by MC68040/D

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

CROSETTO D.

MC68040

Product Preview

Third-Generation, 32-Bit Very High Performance Microprocessor

The MC68040 is Motorola's third generation of M68000-compatible, high-performance, 32-bit microprocessors. The MC68040 is a virtual memory microprocessor employing multiple, concurrent execution units and highly integrated architecture to provide very high performance in a monolithic HCMOS device. On a single chip, the MC68040 integrates an MC68030-compatible integer processing unit (IPU), an IEEE-compatible (ANSI/IEEE Standard 754), floating-point unit, fully independent 8K-byte instruction and data caches, and a paged memory management unit (MMU). A high degree of instruction execution parallelism is achieved through the use of multiple independent execution pipelines, multiple internal buses, and full internal Harvard architecture, including separate physical address space caches for both instruction-stream and data-stream accesses. With the inclusion of bus monitoring hardware, the MC68040 directly supports multiprocessing applications.

The MC68040 is user object-code compatible with previous members of the M68000 Family and is specifically optimized to reduce the execution time of compiler-generated code. The MC68040 is implemented in Motorola's latest HCMOS technology, providing an ideal balance between speed, power, and physical device size.

The main features of the MC68040 are as follows:

- User Object-Code Compatibility with All Earlier M68000 Microprocessors
- Integer Unit Rated at 15 MIPS
- IEEE-Compatible Floating-Point Unit Rated at 4 MFLOPS
- 8K-Byte Instruction Cache and 8K-Byte Data Cache Can Be Accessed Simultaneously
- Multimaster/Multiprocessor Support via Bus Monitoring
- 32-Bit, Nonmultiplexed Address and Data Buses
- Concurrent Integer Unit, Floating-Point Unit, Bus Controller, and Snoop Controllers Maximize Throughput
- 4-Gigabyte Direct Addressing Range
- Software Support Including Optimizing C Compiler and UNIX System V Port

Figure 1 is a simplified block diagram of the MC68040. Instruction execution is pipelined in both integer and floating-point execution units, and the independent caches are attached to a multiple internal bus system.

The MC68040 IPU implements the MC68030 instruction set and is fully user object-code compatible with all previous members of the M68000 Family. The IPU has been optimized to significantly reduce the execution time of compiler-generated code.

The on-chip floating-point unit is user object-code compatible with the MC68882 floating-point coprocessor and conforms to the ANSI/IEEE Standard 754 for binary floating-point arithmetic. Dedicated hardware in the floating-point unit directly supports the most commonly used floating-point instructions and significantly reduces average instruction execution time. Less frequently used instructions are efficiently emulated in software maintaining compatibility with the MC68882.

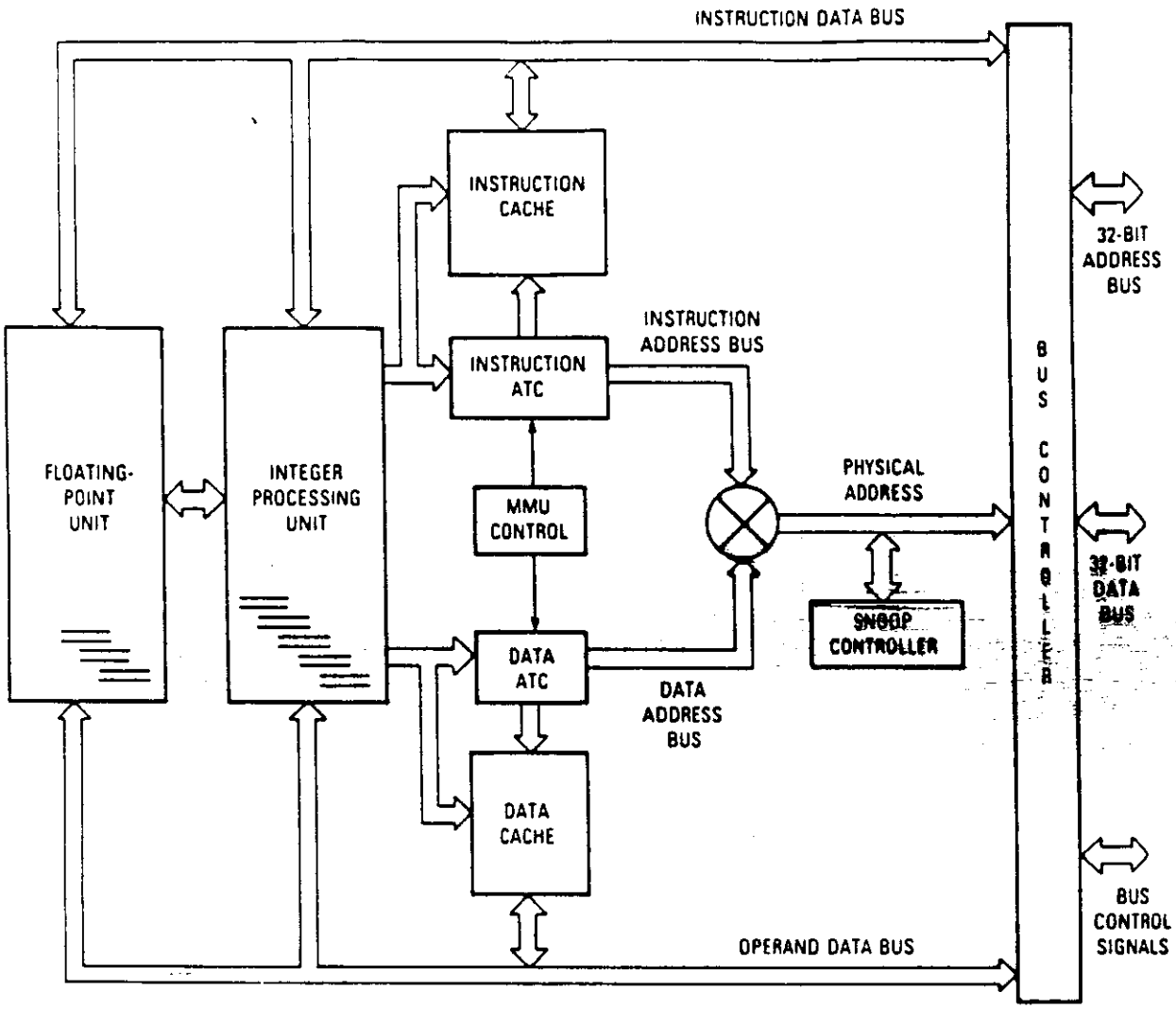


Figure 1. MC68040 Simplified Block Diagram

The MC68040, on-chip, paged-memory management unit (PMMU) supports either demand-paged virtual memory or real-time operating systems. The memory management unit (MMU) contains two, independent, 64-entry address translation caches (ATCs) allowing simultaneous translation of both instruction and data accesses.

The MC68040 contains independent instruction and data caches of 8K bytes each. These caches are located in the physical address space and are multiple set associative.

Coupled with the dual ATCs of the MMU, the independent caches allow the MC68040 integer and floating-point units to simultaneously access instructions and data operands at very high speed.

The MC68040 provides a synchronous, nonmultiplexed, 32-bit bus that is optimized to perform transfers between the internal caches and external memory.

The MC68040 also provides a bus monitor or 'snoop' capability to ensure cache coherency in multimaster and multiprocessing applications.

Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Center; 88 Tanners Drive, Blakelands Milton Keynes, MK145BP, England.
ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; P.O. Box 80300; Cheung Sha Wan Post Office; Kowloon Hong Kong.

Intel 80860: Advanced Semiconductor Technology

- Multiple resources on single chip
  - 32 bit integer and control unit
  - 32/64 bit pipelined floating point units
  - 64 bit 3-D graphics unit
  - Internal translation lookaside buffer
  - 4 KByte instruction cache
  - 8 KByte data cache
- Parallelism with pipelining, multiple functional units
- High peak performance with single chip
  - 80 MFlops single precision
  - 60 MFlops double precision
  - 64 bit and 128 bit internal data paths

Intel 80860: Multiple Resources on Single Chip