Description

The μPD70116 (V30®) is a CMOS 16-bit microprocessor with an internal 16-bit architecture and a 16-bit external data bus. The μPD70116 instruction set is a superset of the μPD8086/8088; however, mnemonics and execution times are different. The μPD70116 additionally has a powerful instruction set including bit processing, packed BCD operations, and high-speed multiplication/ division operations. The μPD70116 can also execute the entire 8080 instruction set and comes with a standby mode that significantly reduces power consumption. It is software-compatible with the μPD70108 microprocessor.

Features

- ☐ Minimum instruction execution time: 250 ns (at 8 MHz), 200 ns to 10 MHz
- ☐ Maximum addressable memory: 1 Mbyte
- ☐ Abundant memory addressing modes
- ☐ 14 x 16-bit register set
- ☐ 101 instructions
- ☐ Instruction set is a superset of μPD8086/8088 instruction set
- ☐ Bit, byte, word, and block operations
- ☐ Bit field operation instructions
- ☐ Packed BCD instructions
- ☐ Multiplication/division instruction execution time: 2.4 μs to 7.1 μs at 8 MHz and 1.9 μs to 5.7 μs at 10 MHz
- ☐ High-speed block transfer instructions: 1 Mbyte/s at 8 MHz, 1.25 Mbyte/s at 10 MHz
- ☐ High-speed calculation of effective addresses: 2 clock cycles in any addressing mode
- ☐ Maskable (INT) and nonmaskable (NMI) interrupt inputs
- ☐ IEEE-796 bus compatible interface
- ☐ 8080 emulation mode
- ☐ CMOS technology
- ☐ Low power consumption
- ☐ Low-power standby mode
- ☐ Single power supply
- ☐ 8-MHz or 10-MHz clock

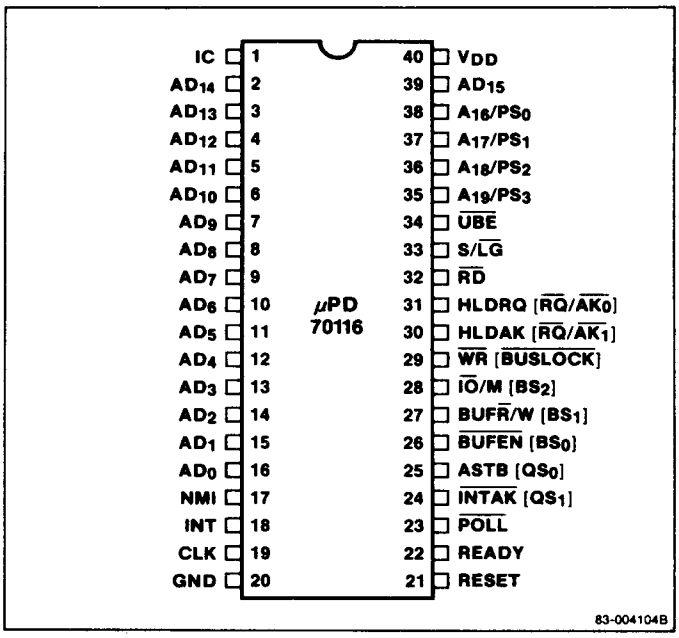
V30 is a registered trademark of NEC Corporation.

Ordering Information

Part Number	Max Frequency of Operation	Package Type
μPD70116C-8	8 MHz	40-pin plastic DIP
C-10	10 MHz	
L-8	8 MHz	44-pin PLCC
L-10	10 MHz	
GC-8	8 MHz	52-pin plastic QFP (P52GC-100-3B6)
GC-10	10 MHz	

Pin Configurations

40-Pin Plastic DIP

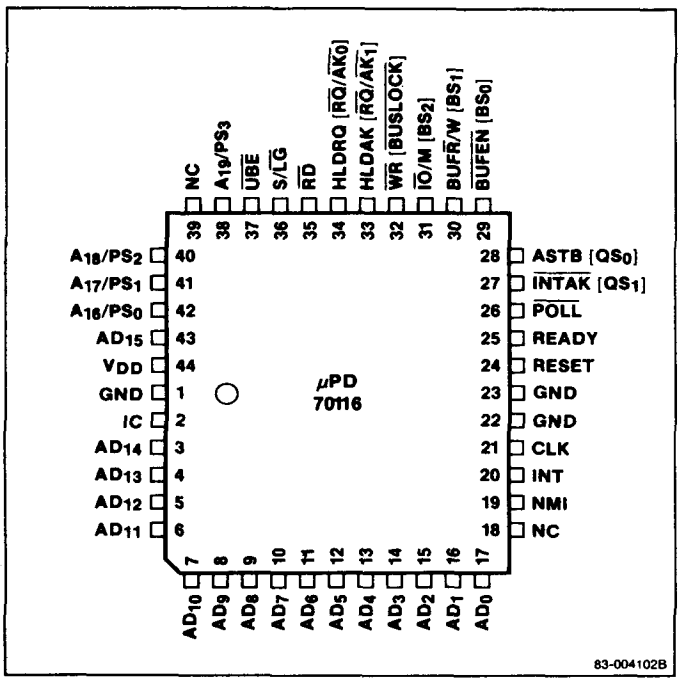


3

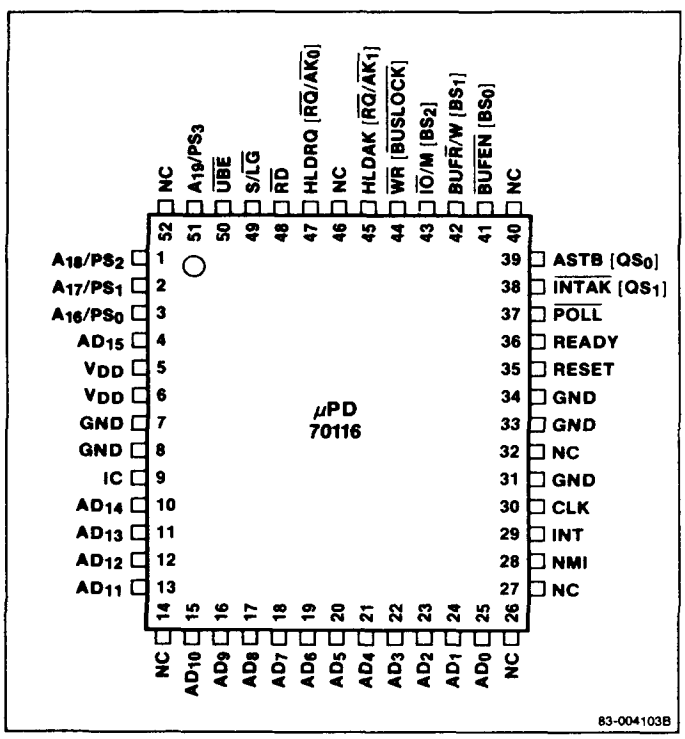
50029-1 (NECEL-062)

Pin Configurations (cont)

44-Pin Plastic Leaded Chip Carrier (PLCC)



52-Pin Plastic QFP



Pin Identification

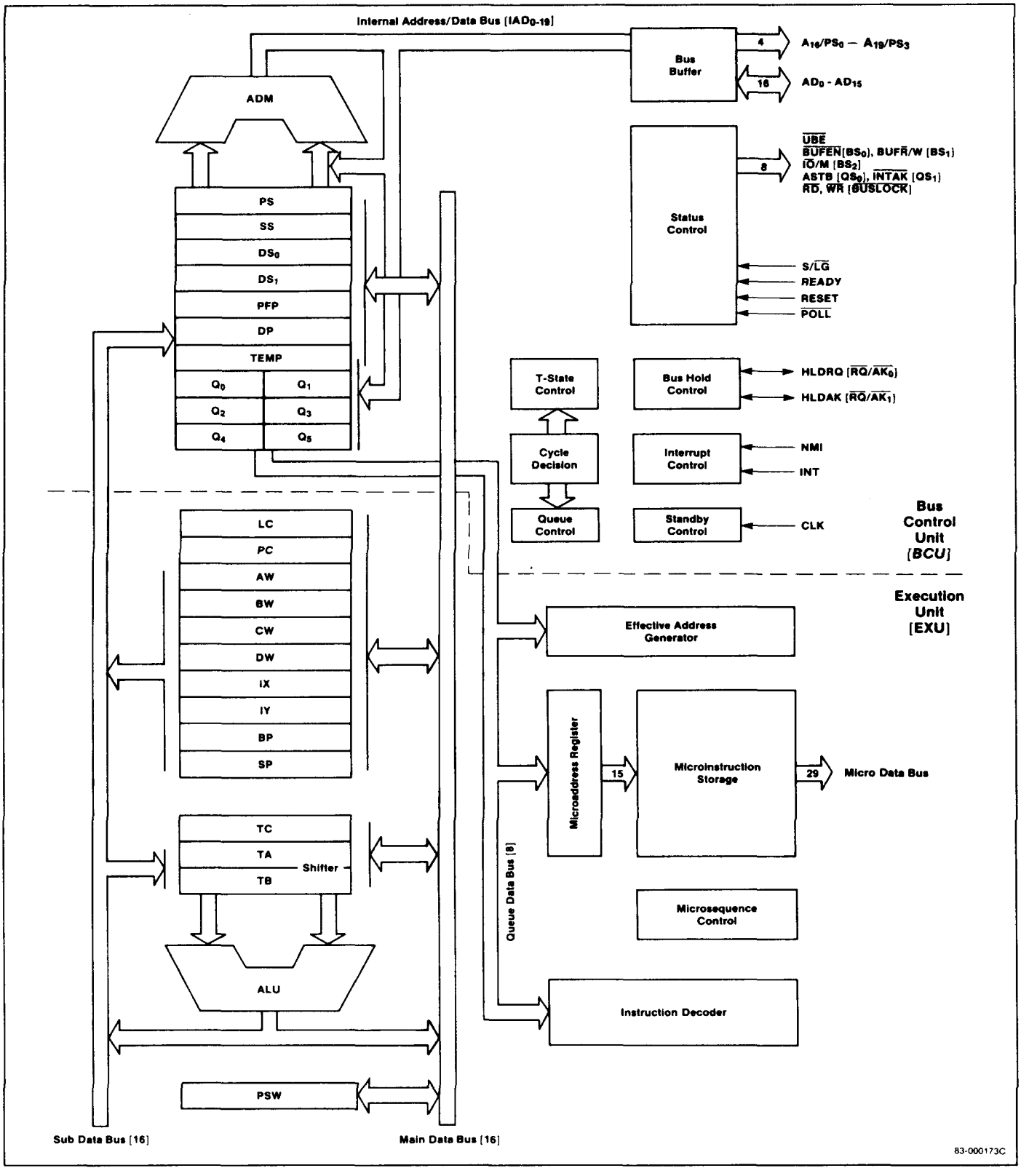
Symbol	Direction	Function
IC*		Internally connected
AD_{14} - AD_0	In/Out	Address/data bus
NMI	In	Nonmaskable interrupt input
INT	In	Maskable interrupt input
CLK	In	Clock input
GND		Ground potential
RESET	In	Reset input
READY	In	Ready input
$\overline{POLL}$	In	Poll input
$\overline{INTAK}$ (QS_1)	Out	Interrupt acknowledge output (queue status bit 1 output)
ASTB (QS_0)	Out	Address strobe output (queue status bit 0 output)
$\overline{BUFEN}$ (BS_0)	Out	Buffer enable output (bus status bit 0 output)
$BUF\overline{R}/W$ (BS_1)	Out	Buffer read/write output (bus status bit 1 output)
$\overline{IO}/M$ (BS_2)	Out	Access is I/O or memory (bus status bit 2 output)
$\overline{WR}$ ($\overline{BUSLOCK}$)	Out	Write strobe output (bus lock output)
HLDAK ($\overline{RQ}/\overline{AK_1}$)	Out (In/Out)	Hold acknowledge output, (bus hold request input/ acknowledge output 1)
HLDRQ ($\overline{RQ}/\overline{AK_0}$)	In (In/Out)	Hold request input (bus hold request input/ acknowledge output 0)
$\overline{RD}$	Out	Read strobe output
$S/\overline{LG}$	In	*Small-scale/large-scale* system input
$\overline{UBE}$	Out	Upper byte enable
A_{19}/PS_3 - A_{16}/PS_0	Out	Address bus, high bits or processor status output
AD_{15}	In/Out	Address/data bus, bit 15
V_{DD}		Power supply

Notes: * IC should be connected to ground.

Where pins have different functions in small- and large-scale systems, the large-scale system pin symbol and function are in parentheses.

Unused input pins should be tied to ground or V_{DD} to minimize power dissipation and prevent the flow of potentially harmful currents.

Block Diagram



Pin Functions

Some pins of the μPD70116 have different functions according to whether the microprocessor is used in a small- or large-scale system. Other pins function the same way in either type of system.

AD_{15} - AD_0 [Address/Data Bus]

For small- and large-scale systems.

AD_{15} - AD_0 is a time-multiplexed address and data bus. When high, an AD bit is a one; when low, an AD bit is a zero. This bus contains the lower 16 bits of the 20-bit address during T1 of the bus cycle. It is used as a 16-bit data bus during T2, T3, and T4 of the bus cycle.

The address/data bus is a three-state bus and can be at a high or low level during standby mode. The bus will be high impedance during hold and interrupt acknowledge.

NMI [Nonmaskable Interrupt]

For small- and large-scale systems.

This pin is used to input nonmaskable interrupt requests. NMI cannot be masked by software. This input is positive edge triggered and must be held high for five clocks to guarantee recognition. Actual interrupt processing begins, however, after completion of the instruction in progress.

The contents of interrupt vector 2 determine the starting address for the interrupt-servicing routine. Note that a hold request will be accepted even during NMI acknowledge.

This interrupt will cause the μPD70116 to exit the standby mode.

INT [Maskable Interrupt]

For small- and large-scale systems.

This pin is an interrupt request that can be masked by software.

INT is active high level and is sensed during the last clock of the instruction. The interrupt will be accepted if the interrupt enable flag IE is set. The CPU outputs the $\overline{\text{INTAK}}$ signal to inform external devices that the interrupt request has been granted. INT must be asserted until the interrupt acknowledge signal is returned.

If NMI and INT interrupts occur at the same time, NMI has higher priority than INT and INT cannot be accepted. A hold request will be accepted during INT acknowledge.

This interrupt causes the μPD70116 to exit the standby mode.

CLK [Clock]

For small- and large-scale systems.

This pin is used for external clock input.

RESET [Reset]

For small- and large-scale systems.

This pin is used for the CPU reset signal. It is an active high level. Input of this signal has priority over all other operations. After the reset signal input returns to a low level, the CPU begins execution of the program starting at address FFFF0H. RESET input must be kept high for at least 4 clock cycles.

In addition to causing normal CPU start, RESET input will cause the μPD70116 to exit the standby mode.

READY [Ready]

For small- and large-scale systems.

When the memory or I/O device being accessed cannot complete data read or write within the CPU basic access time, it can generate a CPU wait state (Tw) by setting this signal to inactive (low level) and requesting a read/write cycle delay.

If the READY signal is active (high level) during either the T3 or Tw state, the CPU will not generate a wait state. READY is not synchronized internally. To guarantee correct operation, external logic must ensure that setup and hold times relative to CLK are met.

$\overline{\text{POLL}}$ [Poll]

For small- and large-scale systems.

The CPU checks this input upon execution of the POLL instruction. If the input is low, then execution continues. If the input is high, the CPU will check the $\overline{\text{POLL}}$ input every five clock cycles until the input becomes low again.

The $\overline{\text{POLL}}$ and READY functions are used to synchronize CPU program execution with the operation of external devices.

$\overline{\text{RD}}$ [Read Strobe]

For small- and large-scale systems.

The CPU outputs this strobe signal during data read from an I/O device or memory. The $\overline{\text{IO}}$/M signal is used to select between I/O and memory.

This three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

S/$\overline{\text{LG}}$ [Small/Large]

For small- and large-scale systems.

This signal determines the operation mode of the CPU. This signal is fixed at either a high or low level. When this signal is a high level, the CPU will operate in small-scale system mode, and when low, in the large-scale system mode. A small-scale system will have at most one bus master such as a DMA controller device on the bus. A large-scale system can have more than one bus master accessing the bus as well as the CPU.

$\overline{\text{INTAK}}$ [Interrupt Acknowledge]
For small-scale systems.

The CPU generates the $\overline{\text{INTAK}}$ signal low when it accepts an INT signal.

The interrupting device synchronizes with this signal and outputs the interrupt vector to the CPU via the data bus (AD_7 - AD_0). $\overline{\text{INTAK}}$ is held at a high-level in the standby mode.

ASTB [Address Strobe]
For small-scale systems.

The CPU outputs this strobe signal to latch address information at an external latch.

ASTB is held at a low level during standby mode and hold acknowledge.

$\overline{\text{BUFEN}}$ [Buffer Enable]
For small-scale systems.

This is used as the output enable signal for an external bidirectional buffer. The CPU generates this signal during data transfer operations with external memory or I/O devices or during input of an interrupt vector.

This three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

BUF$\overline{\text{R}}$/W [Buffer Read/Write]
For small-scale systems.

The output of this signal determines the direction of data transfer with an external bidirectional buffer. A high output causes transmission from the CPU to the external device; a low signal causes data transfer from the external device to the CPU.

BUF$\overline{\text{R}}$/W is a three-state output and enters the high-impedance state during hold acknowledge.

$\overline{\text{IO}}$/M [IO/Memory]
For small-scale systems.

The CPU generates this signal to specify either I/O access or memory access. A low-level output specifies I/O and a high-level signal specifies memory.

$\overline{\text{IO}}$/M's output is three state and becomes high impedance during hold acknowledge.

$\overline{\text{WR}}$ [Write Strobe]
For small-scale systems.

The CPU generates this strobe signal during data write to an I/O device or memory. Selection of either I/O or memory is performed by the $\overline{\text{IO}}$/M signal.

This three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

HLDAK [Hold Acknowledge]
For small-scale systems.

The HLDAK signal is used to indicate that the CPU accepts the hold request signal (HLDRQ). When this signal is a high level, the address bus, address/data bus, the control lines become high impedance.

HLDRQ [Hold Request]
For small-scale systems.

This input signal is used by external devices to request the CPU to release the address bus, address/data bus, and the control bus.

$\overline{\text{UBE}}$ [Upper Byte Enable]
For small- and large-scale systems.

UBE indicates the use of the upper eight bits (AD_{15} -AD_8) of the address/data bus during a bus cycle. This signal is active low during T1 for read, write, and interrupt acknowledge cycles when AD_{15} - AD_8 are to be used. Bus cycles in which $\overline{\text{UBE}}$ is active are shown in the following table.

Type of Bus Operation	$\overline{\text{UBE}}$	AD_0	Number of Bus Cycles
Word at even address	0	0	1
Word at odd address	0 1	1* 0**	2
Byte at even address	1	0	1
Byte at odd address	0	1	1

Notes: * First bus cycle
** Second bus cycle

$\overline{\text{UBE}}$ is low continuously during the interrupt acknowledge state.

The three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

A_{19}/PS_3 - A_{16}/PS_0 [Address Bus/Processor Status]
For small- and large-scale systems.

These pins are time multiplexed to operate as an address bus and as processor status signals.

When used as the address bus, these pins are the high 4 bits of the 20-bit memory address. During I/O access, all 4 bits output data 0.

The processor status signals are provided for both memory and I/O use. PS_3 is always 0 in the native mode and 1 in 8080 emulation mode. The interrupt enable flag (IE) is output on pin PS_2. Pins PS_1 and PS_0 indicate which memory segment is being accessed.

A_{17}/PS_1	A_{16}/PS_0	Segment
0	0	Data segment 1
0	1	Stack segment
1	0	Program segment
1	1	Data segment 0

The output of these pins is three state and becomes high impedance during hold acknowledge.

QS_1, QS_0 [Queue Status]

For large-scale systems.

The CPU uses these signals to allow external devices, such as the floating-point arithmetic processor chip (μPD72091), to monitor the status of the internal CPU instruction queue.

QS_1	QS_0	Instruction Queue Status
0	0	NOP (queue does not change)
0	1	First byte of instruction
1	0	Flush queue
1	1	Subsequent bytes of instruction

The instruction queue status indicated by these signals is the status when the execution unit (EXU) accesses the instruction queue. The data output from these pins is therefore valid only for one clock cycle immediately following queue access. These status signals are provided so that the floating-point processor chip can monitor the CPU's program execution status and synchronize its operation with the CPU when control is passed to it by the FPO (Floating Point Operation) instructions. These outputs are held low level in the standby mode.

BS_2 - BS_0 [Bus Status]

For large-scale systems.

The CPU uses these status signals to allow an external bus controller to monitor what the current bus cycle is.

The external bus controller decodes these signals and generates the control signals required to perform access of the memory or I/O device.

BS_2	BS_1	BS_0	Bus Cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Program fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive state

The output of these signals is three state and becomes high impedance during hold acknowledge.

$\overline{BUSLOCK}$ [Bus Lock]

For large-scale systems.

The CPU uses this signal to secure the bus while executing the instruction immediately following the BUSLOCK prefix instruction and during interrupt acknowledge cycles. It is a status signal to the other bus masters in a multiprocessor system inhibiting them from using the system bus during this time.

The output of this signal is three state and becomes high impedance during hold acknowledge. $\overline{BUSLOCK}$ is high during standby mode except if the HALT instruction has a BUSLOCK prefix.

$\overline{RQ}/\overline{AK}_1$, $\overline{RQ}/\overline{AK}_0$ [Hold Request/Acknowledge]

For large-scale systems.

These pins function as bus hold request inputs ($\overline{RQ}$) and as bus hold acknowledge outputs ($\overline{AK}$). $\overline{RQ}/\overline{AK}_0$ has a higher priority than $\overline{RQ}/\overline{AK}_1$.

These pins have three-state outputs with on-chip pull-up resistors which keep the pin at a high level when the output is high impedance. $\overline{RQ}$ inputs must be properly synchronized to CLK.

V_{DD} [Power Supply]

For small- and large-scale systems.

This pin is used for the +5 V power supply.

GND [Ground]

For small- and large-scale systems.

This pin is used for ground.

IC [Internally Connected]

This pin is used for tests performed at the factory by NEC. The μPD70116 is used with this pin at ground potential.

Absolute Maximum Ratings

$T_A = +25°C$

Power supply voltage, V_{DD}	−0.5 V to +7.0 V
Power dissipation, PD_{MAX}	0.5 W
Input voltage, V_I	−0.5 V to V_{DD} + 0.3 V
CLK input voltage, V_K	−0.5 V to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 V to V_{DD} + 0.3 V
Operating temperature at 5 MHz, T_{OPT}	−40°C to +85°C
Storage temperature, T_{STG}	−65°C to +150°C

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Capacitance

$T_A = +25°C$, $V_{DD} = 0$ V

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Input capacitance	C_I		15	pF	fc = 1 MHz Unmeasured pins returned to 0 V
I/O capacitance	C_{IO}		15	pF	

3

DC Characteristics

μPD70116-8, μPD70116-10, $T_A = -10°C$ to +70°C, $V_{DD} = +5$ V ± 5%

Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
Input voltage high	V_{IH}	2.2		$V_{DD}+0.3$	V	
Input voltage low	V_{IL}	−0.5		0.8	V	
CLK input voltage high	V_{KH}	3.9		$V_{DD}+1.0$	V	
CLK input voltage low	V_{KL}	−0.5		0.6	V	
Output voltage high	V_{OH}	0.7 x V_{DD}			V	$I_{OH} = -400$ μA
Output voltage low	V_{OL}			0.4	V	$I_{OL} = 2.5$ mA
Input leakage current high	I_{LIH}			10	μA	$V_I = V_{DD}$
Input leakage current low	I_{LIL}			−10	μA	$V_I = 0$ V
Output leakage current high	I_{LOH}			10	μA	$V_O = V_{DD}$
Output leakage current low	I_{LOL}			−10	μA	$V_O = 0$ V
Supply current	I_{DD}	70116-8 8 MHz	45	80	mA	Normal operation
			6	12	mA	Standby mode
		70116-10 10 MHz	60	100	mA	Normal operation
			7	14	mA	Standby mode

AC Characteristics

T_A = −10° to +70°C, V_{DD} = +5 V ± 5 %
T_A = −40°C to +85°C, V_{DD} = +5 V ± 10 %

		μPD70116-5		μPD70116-8		μPD70116-10			
Parameter	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**	**Conditions**
Small/Large Scale									
Clock cycle	t_{CYK}	200	500	125	500	100	500	ns	
Clock pulse width high	t_{KKH}	69		44		41		ns	V_{KH} = 3.0 V
Clock pulse width low	t_{KKL}	90		60		49		ns	V_{KL} = 1.5 V
Clock rise time	t_{KR}		10		10		5	ns	1.5 V to 3.0 V
Clock fall time	t_{KF}		10		10		5	ns	3.0 V to 1.5 V
READY inactive setup to CLK ↓	t_{SRYLK}	−8		−8		−10		ns	
READY inactive hold after CLK ↑	t_{HKRYH}	30		20		20		ns	
READY active setup to CLK ↑	t_{SRYHK}	t_{KKL}−8		t_{KKL}−8		t_{KKL}−10		ns	
READY active hold after CLK ↑	t_{HKRYL}	30		20		20		ns	
Data setup time to CLK ↓	t_{SDK}	30		20		10		ns	
Data hold time after CLK ↓	t_{HKD}	10		10		10	ns		
NMI, INT, $\overline{\text{POLL}}$ setup time to CLK ↑	t_{SIK}	30		15		15		ns	
Input rise time (except CLK)	t_{IR}		20		20		20	ns	0.8 V to 2.2 V
Input fall time (except CLK)	t_{IF}		12		12		12	ns	2.2 V to 0.8 V
Output rise time	t_{OR}		20		20		20	ns	0.8 V to 2.2 V
Output fall time	t_{OF}		12		12		12	ns	2.2 V to 0.8 V
Small Scale									
Address delay time from CLK ↓	t_{DKA}	10	90	10	60	10	48	ns	
Address hold time from CLK ↓	t_{HKA}	10		10		10		ns	
PS delay time from CLK ↓	t_{DKP}	10	90	10	60	10	50	ns	
PS float delay time from CLK ↑	t_{FKP}	10	80	10	60	10	50	ns	
Address setup time to ASTB ↓	t_{SAST}	t_{KKL}−60		t_{KKL}−30		t_{KKL}−30		ns	
Address float delay time from CLK ↓	t_{FKA}	t_{HKA}	80	t_{HKA}	60	t_{HKA}	50	ns	C_L = 100 pF
ASTB ↑ delay time from CLK ↓	t_{DKSTH}		80		50		40	ns	
ASTB ↓ delay time from CLK ↑	t_{DKSTL}		85		55		45	ns	
ASTB width high	t_{STST}	t_{KKL}−20		t_{KKL}−10		t_{KKL}−10		ns	
Address hold time from ASTB ↓	t_{HSTA}	t_{KKH}−10		t_{KKH}−10		t_{KKH}−10		ns	

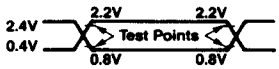
AC Characteristics

T_A = −10° to +70°C, V_{DD} = +5 V ± 5 %
T_A = −40°C to +85°C, V_{DD} = +5 V ± 10 %

Parameter	Symbol	μPD70116-5 Min	μPD70116-5 Max	μPD70116-8 Min	μPD70116-8 Max	μPD70116-10 Min	μPD70116-10 Max	Unit	Conditions
Small Scale (cont)									
Control delay time from CLK	t_{DKCT}	10	110	10	65	10	55	ns	
Address float to RD ↓	t_{AFRL}	0		0		0		ns	
$\overline{RD}$ ↓ delay time from CLK ↓	t_{DKRL}	10	165	10	80	10	70	ns	
$\overline{RD}$ ↑ delay time from CLK ↓	t_{DKRH}	10	150	10	80	10	60	ns	
Address delay time from RD ↑	t_{DRHA}	t_{CYK}−45		t_{CYK}−40		t_{CYK}−35		ns	
$\overline{RD}$ width low	t_{RR}	$2t_{CYK}$−75		$2t_{CYK}$−50		$2t_{CYK}$−40		ns	C_L = 100 pF
Data output delay time from CLK ↓	t_{DKD}	10	90	10	60	10	50	ns	
Data float delay time from CLK ↓	t_{FKD}	10	80	10	60	10	50	ns	
$\overline{WR}$ width low	t_{WW}	$2t_{CYK}$−60		$2t_{CYK}$−40		$2t_{CYK}$−35		ns	
HLDRQ setup time to CLK ↑	t_{SHQK}	35		20		20		ns	
HLDAK delay time from CLK ↓	t_{DKHA}	10	160	10	100	10	60	ns	
$\overline{WR}$ ↑ to $\overline{BUFEN}$ ↑	t_{WCT}	t_{KKL}−20		t_{KKL}−20		t_{KKL}−20		ns	
Large Scale									
Address delay time from CLK ↓	t_{DKA}	10	90	10	60	10	48	ns	
Address hold time from CLK ↓	t_{HKA}	10		10		10		ns	
PS delay time from CLK ↓	t_{DKP}	10	90	10	60	10	50	ns	
PS float delay time from CLK ↑	t_{FKP}	10	80	10	60	10	50	ns	
Address float delay time from CLK ↓	t_{FKA}	t_{HKA}	80	t_{HKA}	60	t_{HKA}	50	ns	
Address delay time from $\overline{RD}$ ↑	t_{DRHA}	t_{CYK}−45		t_{CYK}−40		t_{CYK}−35		ns	
ASTB delay time from BS ↓	t_{DBST}		15		15		15	ns	
BS ↓ delay time from CLK ↑	t_{DKBL}	10	110	10	60	10	50	ns	
BS ↑ delay time from CLK ↓	t_{DKBH}	10	130	10	65	10	50	ns	
$\overline{RD}$ ↓ delay time from address float	t_{DAFRL}	0		0		0		ns	C_L = 100 pf
$\overline{RD}$ ↓ delay time from CLK ↓	t_{DKRL}	10	165	10	80	10	70	ns	
$\overline{RD}$ ↑ delay time from CLK ↓	t_{DKRH}	10	150	10	80	10	60	ns	
$\overline{RD}$ width low	t_{RR}	$2t_{CYK}$−75		$2t_{CYK}$−50		$2t_{CYK}$−40		ns	
Data output delay time from CLK ↓	t_{DKD}	10	90	10	60	10	50	ns	
Data float delay time from CLK ↓	t_{FKD}	10	80	10	60	10	50	ns	
$\overline{AK}$ delay time from CLK ↓	t_{DKAK}		70		50		40	ns	
$\overline{RQ}$ setup time to CLK ↑	t_{SRQK}	20		10		9		ns	
$\overline{RQ}$ hold time from CLK ↓	t_{HKRQ1}	0		0		0		ns	
RQ hold time from CLK ↑	t_{HKRQ2}	40		30		20	ns		

Timing Waveforms

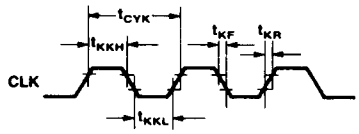
AC Test Input Waveform [Except CLK]



AC Output Test Points

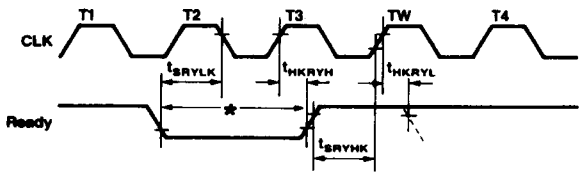
49-000247A

Clock Timing



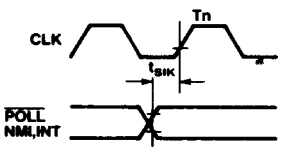
49-000248A

Wait [Ready] Timing

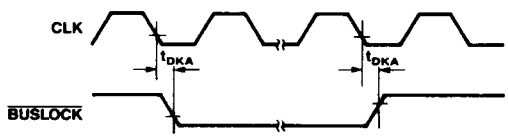


* READY input level must not be changed during this interval.

$\overline{\text{POLL}}$,NMI, INT Input Timing



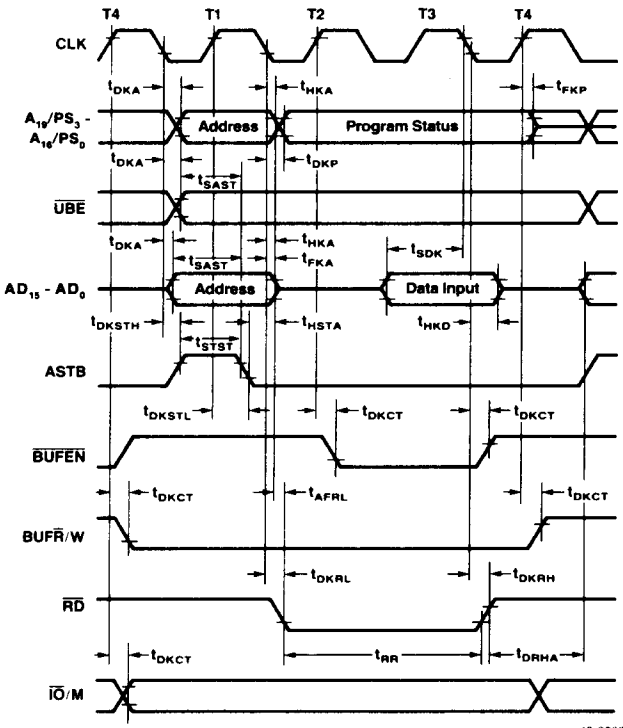
$\overline{\text{BUSLOCK}}$ Output Timing



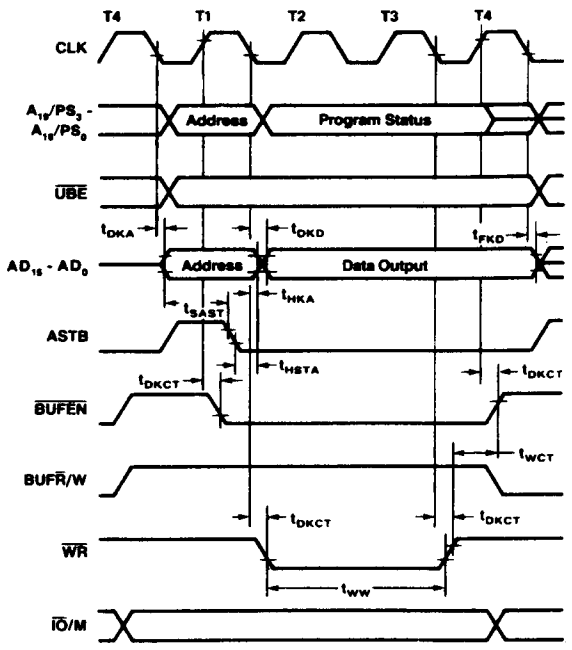
49-0002496

Timing Waveforms (cont)

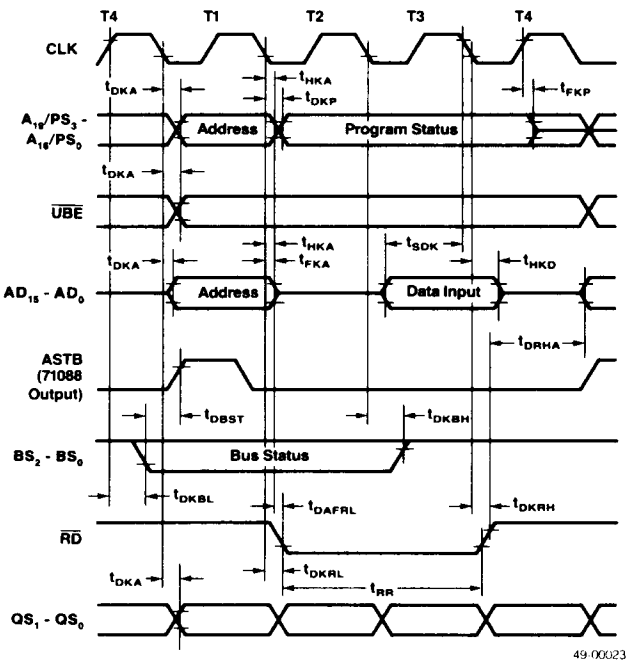
Read Timing [Small Scale]



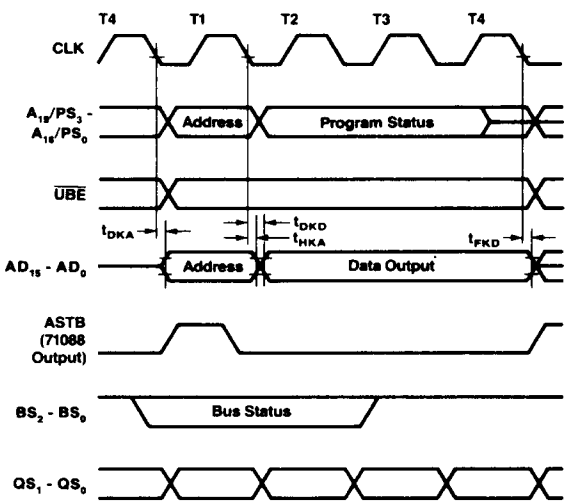
Write Timing [Small Scale]



Read Timing [Large Scale]

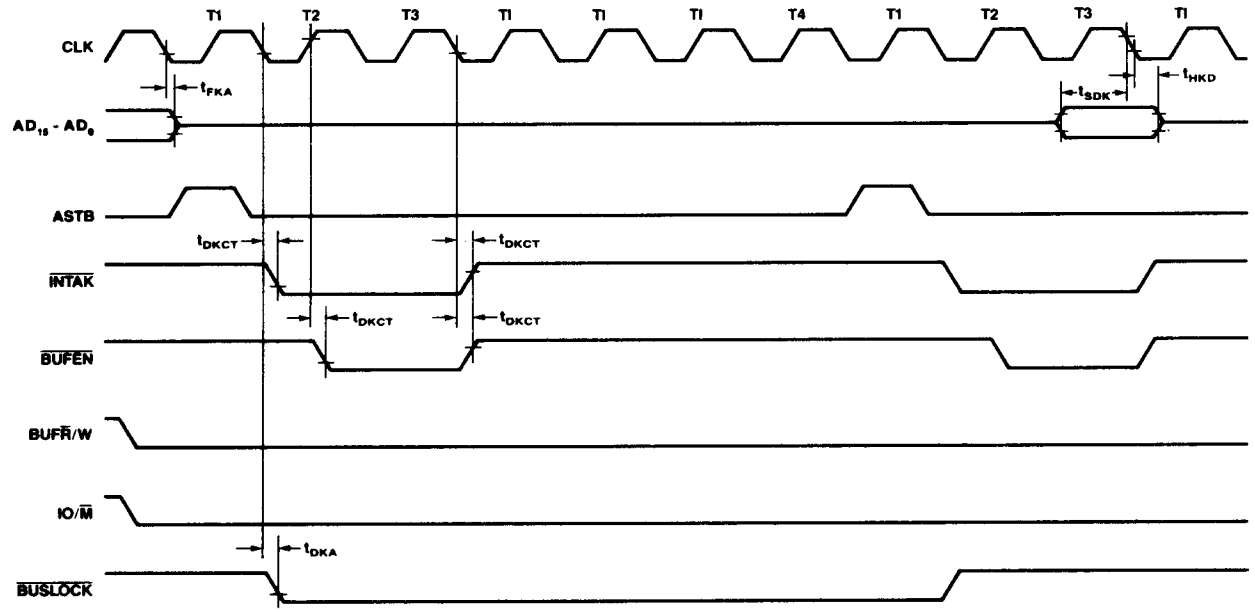


Write Timing [Large Scale]



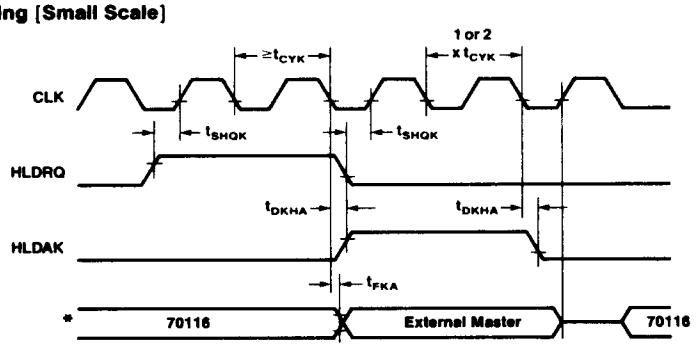
Timing Waveforms (cont)

Interrupt Acknowledge Timing



49-000235B

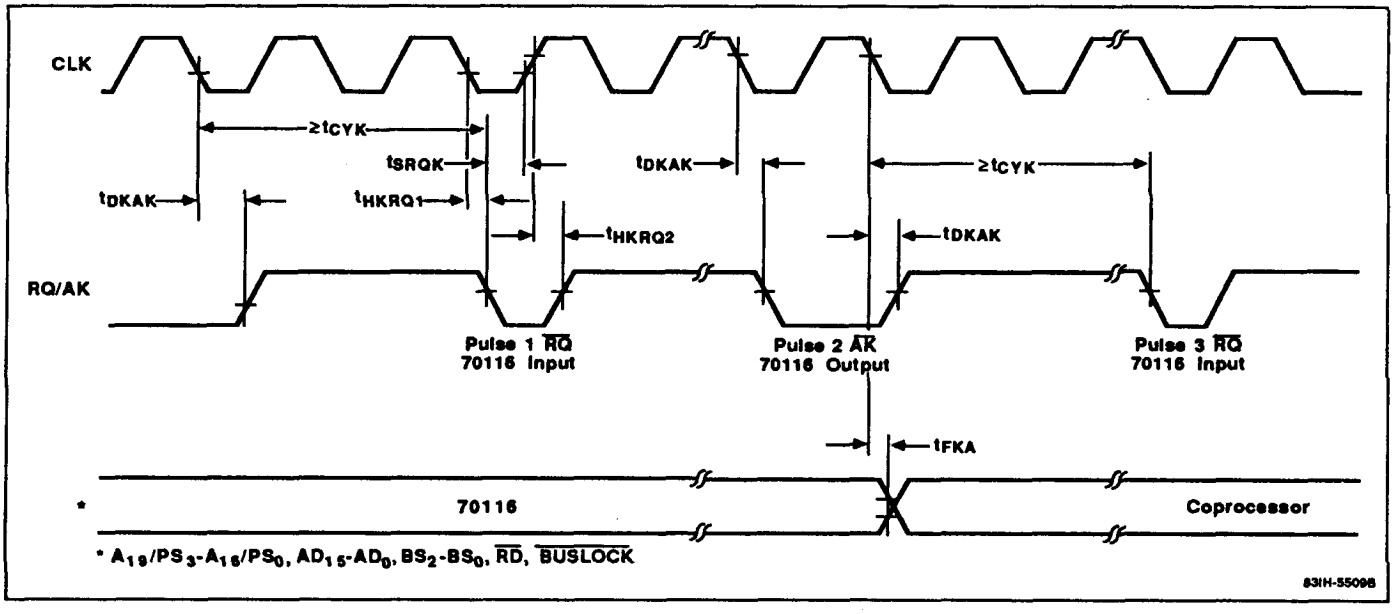
Hold Request/Acknowledge Timing [Small Scale]



*: A_{16}/PS_3 - A_{19}/PS_6, AD_{15} - AD_0, $\overline{RD}$, $\overline{UBE}$, IO/M, BUFR/W, $\overline{WR}$ $\overline{BUFEN}$

49-000236B

Timing Waveforms (cont)



3

Register Configuration

Program Counter [PC]

The program counter is a 16-bit binary counter that contains the segment offset address of the next instruction which the EXU is to execute.

The PC increments each time the microprogram fetches an instruction from the instruction queue. A new location value is loaded into the PC each time a branch, call, return, or break instruction is executed. At this time, the contents of the PC are the same as the Prefetch Pointer (PFP).

Prefetch Pointer [PFP]

The prefetch pointer (PFP) is a 16-bit binary counter which contains a segment offset which is used to calculate a program memory address that the bus control unit (BCU) uses to prefetch the next word for the instruction queue. The contents of PFP are an offset from the PS (Program Segment) register.

The PFP is incremented each time the BCU prefetches an instruction from the program memory. A new location will be loaded into the PFP whenever a branch, call, return, or break instruction is executed. At that time the contents of the PFP will be the same as those of the PC (Program Counter).

Segment Registers [PS, SS, DS_0, and DS_1]

The memory addresses accessed by the μPD70116 are divided into 64K-byte logical segments. The starting (base) address of each segment is specified by a 16-bit segment register, and the offset from this starting address is specified by the contents of another register or by the effective address.

These are the four types of segment registers used.

Segment Register	Default Offset
PS (Program Segment)	PFP
SS (Stack Segment)	SP, effective address
DS_0 (Data Segment 0)	IX, effective address
DS_1 (Data Segment 1)	IY

General-Purpose Registers [AW, BW, CW, and DW]

There are four 16-bit general-purpose registers. Each one can be used as one 16-bit register or as two 8-bit registers by dividing them into their high and low bytes (AH, AL, BH, BL, CH, CL, DH, DL).

Each register is also used as a default register for processing specific instructions. The default assignments are:

AW: Word multiplication/division, word I/O, data conversion, translation, BCD rotation.

AL: Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation

AH: Byte multiplication/division

BW: Translation

CW: Loop control branch, repeat prefix

CL: Shift instructions, rototation instructions, BCD operations

DW: Word multiplication/division, indirect addressing I/O

Pointers [SP, BP] and Index Registers [IX, IY]

These registers serve as base pointers or index registers when accessing the memory using based addressing, indexed addressing, or based indexed addressing.

These registers can also be used for data transfer and arithmetic and logical operations in the same manner as the general-purpose registers. They cannot be used as 8-bit registers.

Also, each of these registers acts as a default register for specific operations. The default assignments are:

SP: Stack operations

IX: Block transfer (source), BCD string operations

IY: Block transfer (destination), BCD string operations

Program Status Word [PSW]

The program status word consists of the following six status and four control flags.

Status Flags

- V (Overflow)
- S (Sign)
- Z (Zero)
- AC (Auxiliary Carry)
- P (Parity)
- CY (Carry)

Control Flags

- MD (Mode)
- DIR (Direction)
- IE (Interrupt Enable)
- BRK (Break)

When the PSW is pushed on the stack, the word images of the various flags are as shown here.

PSW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MD	1	1	1	V	DIR	IE	BRK	S	Z	0	AC	0	P	1	CY

The status flags are set and reset depending upon the result of each type of instruction executed.

Instructions are provided to set, reset, and complement the CY flag directly.

Other instructions set and reset the control flags and control the operation of the CPU. The MD flag can be set/reset only by the BRKEM, RETEM, CALLN, and RETI instructions.

The MD flag can be set/reset only in between executions of BRKEM and RETEM. MD will not be restored, even as the RETI or POP PSW instruction is executed.

High-Speed Execution of Instructions

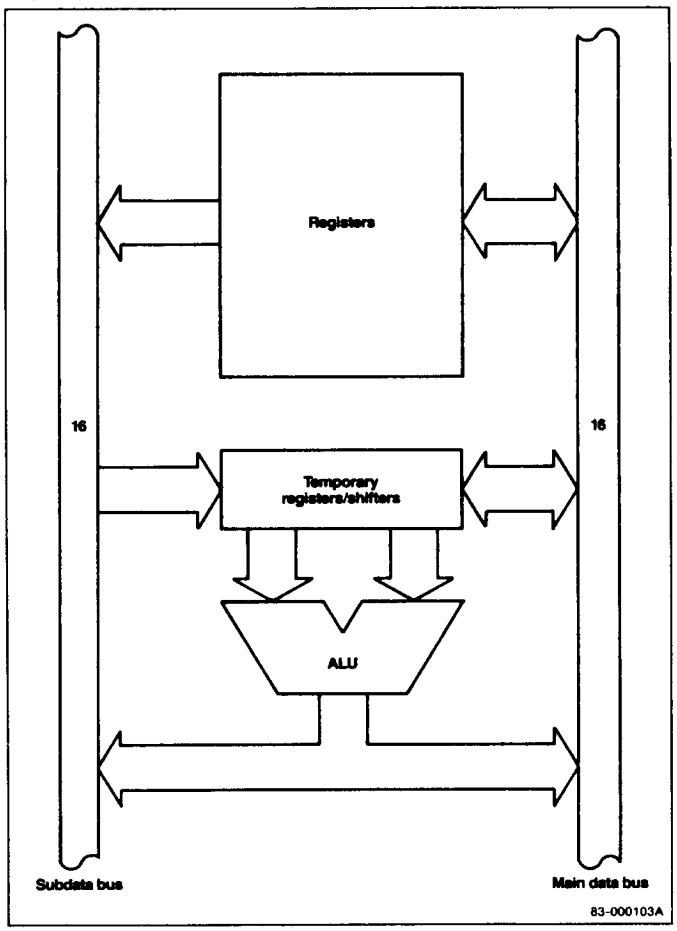
This section highlights the major architectural features that enhance the performance of the μPD70116.

- Dual data bus in EXU
- Effective address generator
- 16/32-bit temporary registers/shifters (TA, TB)
- 16-bit loop counter (LC)
- PC and PFP

Dual Data Bus Method

To reduce the number of processing steps for instruction execution, the dual data bus method has been adopted for the μPD70116 (figure 1). The two data buses (the main data bus and the subdata bus) are both 16 bits wide. For addition/subtraction and logical and comparison operations, processing time has been reduced by some 30% over single-bus systems.

Figure 1. Dual Data Buses



Example

```
ADD    AW, BW    ; AW ← AW + BW
```

Single Bus	Dual Bus
Step 1 TA ← AW	TA ← AW, TB ← BW
Step 2 TB ← BW	AW ← TA + TB
Step 3 AW ← TA + TB	

Effective Address Generator [EAG]

The Effective Address Generator (EAG) (figure 2) is a dedicated block of high-speed logic that computes effective addresses in two clock cycles. If an instruction uses memory, EAG decodes the second and/or third instruction bytes to determine the addressing mode, initiates any bus cycles needed to fetch data required to compute the effective address, and stores the computed effective address in the Data Pointer (DP) register.

Calculating an effective address by the microprogramming method normally requires 5 to 12 clock cycles. This circuit requires only two clock cycles for addresses to be generated for any addressing mode. Thus, processing is several times faster.

16/32-Bit Temporary Registers/Shifters [TA, TB]

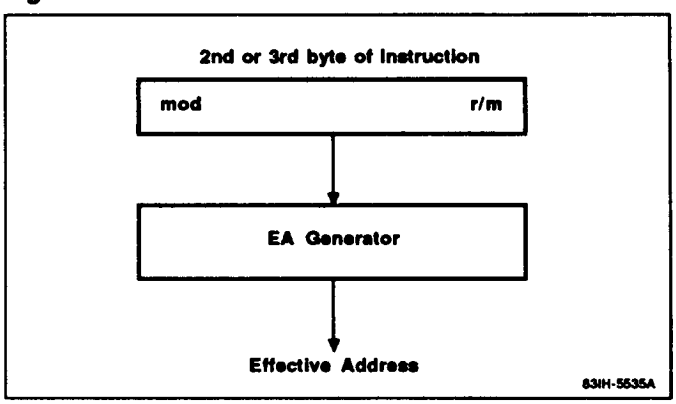
These 16-bit temporary registers/shifters (TA, TB) are provided for multiplication/division and shift/rotation instructions.

These circuits have decreased the execution time of multiplication/division instructions. In fact, these instructions can be executed about four times faster than with the microprogramming method.

TA + TB: 32-bit temporary register/shifter for multiplication and division instructions.

TB: 16-bit temporary register/shifter for shift/rotation instructions.

Figure 2. Effective Address Generator



Loop Counter [LC]

This counter is used to count the number of loops for a primitive block transfer instruction controlled by a repeat prefix instruction and the number of shifts that will be performed for a multiple bit shift/rotation instruction.

The processing performed for a multiple bit rotation of a register is shown below. The average speed is approximately doubled over the microprogram method.

Example

RORC AW, CL ; CL = 5

Microprogram method	LC method
8 + (4 x 5) = 28 clocks	7 + 5 = 12 clocks

Program Counter and Prefetch Pointer [PC and PFP]

The μPD70116 microprocessor has a program counter, (PC) which addresses the program memory location of the instruction to be executed next, and a prefetch pointer(PFP), which addresses the program memory location to be accessed next. Both functions are provided in hardware. A time saving of several clocks is realized for branch, call, return, and break instruction execution, compared with microprocessors that have only one instruction pointer.

Enhanced Instructions

In addition to the μPD8088/86 instructions, the μPD70116 has the following enhanced instructions.

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes 8 general registers onto stack
POP R	Pops 8 general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Enhanced Stack Operation Instructions

PUSH imm

This instruction allows immediate data to be pushed onto the stack.

PUSH R/POP R

These instructions allow the contents of the eight general registers to be pushed onto or popped from the stack with a single instruction.

Enhanced Multiplication Instructions

MUL reg16, imm16/MUL mem16, imm16

These instructions allow the contents of a register or memory location to be multiplied by immediate data.

Enhanced Shift and Rotate Instructions

SHL reg, imm8/SHR reg, imm8/SHRA reg, imm8

These instructions allow the contents of a register to be shifted by the number of bits defined by the immediate data.

ROL reg, imm8/ROR reg, imm8/ROLC reg, imm8/ RORC reg, imm8

These instructions allow the contents of a register to be rotated by the number of bits defined by the immediate data.

Check Array Boundary Instruction

CHKIND reg16, mem32

This instruction is used to verify that index values pointing to the elements of an array data structure are within the defined range. The lower limit of the array should be in memory location mem32, the upper limit in mem32 + 2. If the index value in reg16 is not between these limits when CHKIND is executed, a BRK 5 will occur. This causes a jump to the location in interrupt vector 5.

Block I/O Instructions

OUTM DW, src-block/INM dst-block, DW

These instructions are used to output or input a string to or from memory, when preceded by a repeat prefix.

Stack Frame Instructions

PREPARE imm16, imm8

This instruction is used to generate the stack frames required by block-structured languages, such as PASCAL and Ada. The stack frame consists of two areas. One area has a pointer that points to another frame which has variables that the current frame can access. The other is a local variable area for the current procedure.

DISPOSE

This instruction releases the last stack frame generated by the PREPARE instruction. It returns the stack and base pointers to the values they had before the PREPARE instruction was used to call a procedure.

Unique Instructions

In addition to the μPD8088/86 instructions and the enhanced instructions, the μPD70116 has the following unique instructions.

Instruction	Function
INS	Insert bit field
EXT	Extract bit field
ADD4S	Adds packed decimal strings
SUB4S	Subtracts one packed decimal string from another
CMP4S	Compares two packed decimal strings
ROL4	Rotates one BCD digit left through AL lower 4 bits
ROR4	Rotates one BCD digit right through AL lower 4 bits
TEST1	Tests a specified bit and sets/resets Z flag
NOT1	Inverts a specified bit
CLR1	Clears a specified bit
SET1	Sets a specified bit
REPC	Repeats next instruction until CY flag is cleared
REPNC	Repeats next instruction until CY flag is set
FPO2	Additional floating point processor call

Variable Length Bit Field Operation Instructions

This category has two instructions: INS (Insert Bit Field) and EXT (Extract Bit Field). These instructions are highly effective for computer graphics and high-level languages. They can, for example, be used for data structures such as packed arrays and record type data used in PASCAL.

INS reg8, reg8/INS reg8, imm4

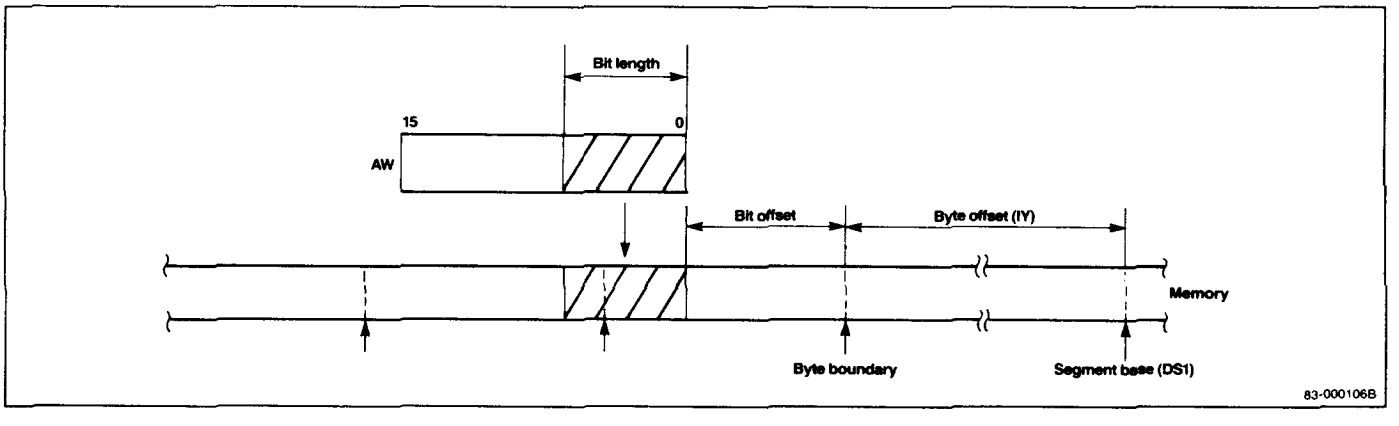
This instruction (figure 3) transfers low bits from the 16-bit AW register (the number of bits is specified by the second operand) to the memory location specified by the segment base (DS_1 register) plus the byte offset (IY register). The starting bit position within this byte is specified as an offset by the lower 4-bits of the first operand.

After each complete data transfer, the IY register and the register specified by the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may specify the number of bits transferred (second operand). Because the maximum transferable bit length is 16-bits, only the lower 4-bits of the specified register (00H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Figure 3. Bit Field Insertion



EXT reg8, reg8/EXT reg8, imm4

This instruction (figure 4) loads to the AW register the bit field data whose bit length is specified by the second operand of the instruction from the memory location that is specified by the DS0 segment register (segment base), the IX index register (byte offset), and the lower 4-bits of the first operand (bit offset).

After the transfer is complete, the IX register and the lower 4-bits of the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may be specified for the second operand. Because the maximum transferrable bit length is 16 bits, however, only the lower 4-bits of the specified register (0H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Packed BCD Operation Instructions

The instructions described here process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte-format operands (ROR4, ROL4). Packed BCD strings may be from 1 to 254 digits in length.

When the number of digits is even, the zero and carry flags will be set according to the result of the operation. When the number of digits is odd, the zero and carry flags may not be set correctly in this case, (CL = odd), the zero flag will not be set unless the upper 4 bits of the highest byte are all zero. The carry flag will not be set unless there is a carry out of the upper 4 bits of the highest byte. When CL is odd, the contents of the upper 4 bits of the highest byte of the result are undefined.

ADD4S

This instruction adds the packed BCD string addressed by the IX index register to the packed BCD string addressed by the IY index register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the overflow flag (V), the carry flag (CY), and zero flag (Z).

BCD string (IY, CL) ← BCD string (IY, CL) + BCD string (IX, CL)

SUB4S

This instruction subtracts the packed BCD string addressed by the IX index register from the packed BCD string addressed by the IY register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the overflow flag (V), the carry flag (CY), and zero flag (Z).

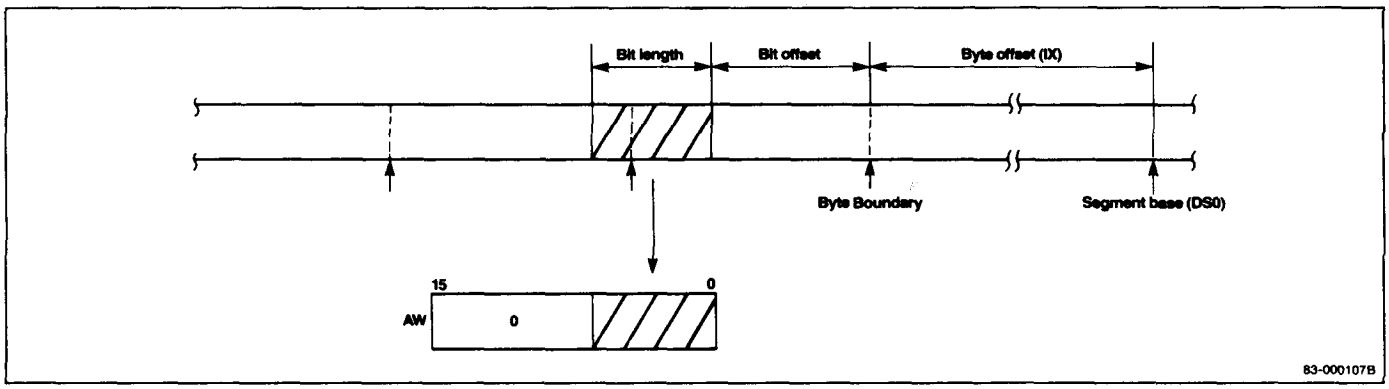
BCD string (IY, CL) ← BCD string (IY, CL) − BCD String (IX, CL)

CMP4S

This instruction performs the same operation as SUB4S except that the result is not stored and only the overflow (V), carry flags (CY) and zero flag (Z) are affected.

BCD string (IY, CL) − BCD string (IX, CL)

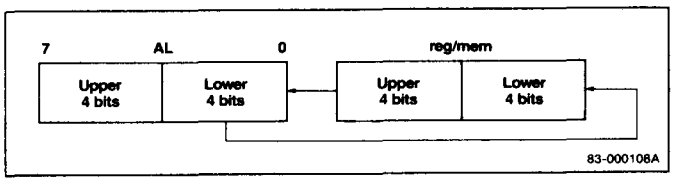
Figure 4. Bit Field Extraction



ROL4

This instruction (figure 5) treats the byte data of the register or memory operand specified by the instruction as BCD data and uses the lower 4 bits of the AL register (AL_L) to rotate that data one BCD digit to the left.

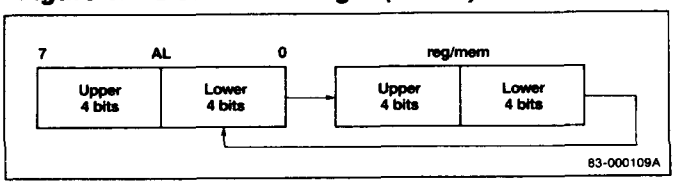
Figure 5. BCD Rotate Left (ROL4)



ROR4

This instruction (figure 6) treats the byte data of the register or memory specified by the instruction as BCD data and uses the lower 4 bits of the AL register (AL_L) to rotate that data one BCD digit to the right.

Figure 6. BCD Rotate Right (ROR4)



Bit Manipulation Instructions

TEST1

This instruction tests a specific bit in a register or memory location. If the bit is 1, the Z flag is reset to 0. If the bit is 0, the Z flag is set to 1.

NOT1

This instruction inverts a specific bit in a register or memory location.

CLR1

This instruction clears a specific bit in a register or memory location.

SET1

This instruction sets a specific bit in a register or memory location.

Repeat Prefix Instructions

REPC

This instruction causes the μPD70116 to repeat the following primitive block transfer instruction until the CY flag becomes cleared or the CW register becomes zero.

REPNC

This instruction causes the μPD70116 to repeat the following primitive block transfer instruction until the CY flag becomes set or the CW register becomes zero.

Floating Point Instruction

FPO2

This instruction is in addition to the μPD8088/86 floating point instruction, FPO1. These instructions are covered in a later section.

Mode Operation Instruction

The μPD70116 has two operating modes (figure 7). One is the native mode which executes μPD8088/86, enhanced and unique instructions. The other is the 8080 emulation mode in which the instruction set of the μPD8080AF is emulated. A mode flag (MD) is provided to select between these two modes. Native mode is selected when MD is 1 and emulation mode when MD is 0. MD is set and reset, directly and indirectly, by executing the mode manipulation instructions.

Two instructions are provided to switch operation from the native mode to the emulation mode and back.

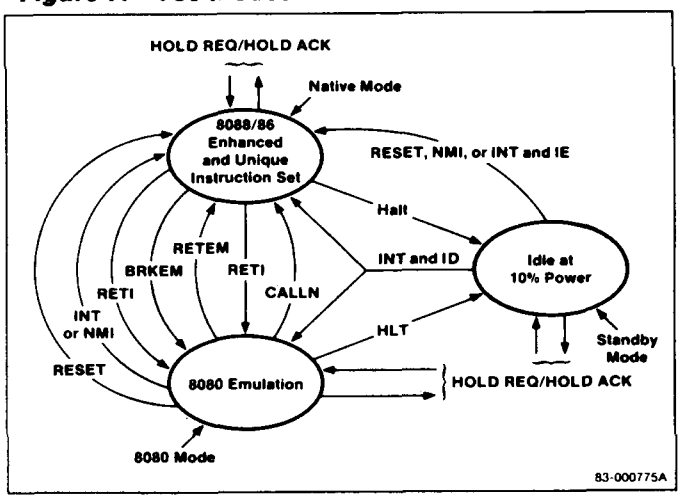
BRKEM (Break for Emulation)
RETEM (Return from Emulation)

Two instructions are used to switch from the emulation mode to the native mode and back.

CALLN (Call Native Routine)
RETI (Return from Interrupt)

The system will return from the 8080 emulation mode to the native mode when the RESET signal is present, or when an external interrupt (NMI or INT) is present.

Figure 7. V30 Modes



BRKEM imm8

This is the basic instruction used to start the 8080 emulation mode. This instruction operates exactly the same as the BRK instruction, except that BRKEM resets the mode flag (MD) to 0. PSW, PS, and PC are saved to the stack. MD is then reset and the interrupt vector specified by the operand imm8 of this command is loaded into PS and PC.

The instruction codes of the interrupt processing routine jumped to are then fetched. Then the CPU executes these codes as μPD8080AF instructions.

In 8080 emulation mode, registers and flags of the μPD8080AF are performed by the following registers and flags of the μPD70116.

	μPD8080AF	μPD70116
Registers:	A	AL
	B	CH
	C	CL
	D	DH
	E	DL
	H	BH
	L	BL
	SP	BP
	PC	PC
Flags:	C	CY
	Z	Z
	S	S
	P	P
	AC	AC

In the native mode, SP is used for the stack pointer. In the 8080 emulation mode this function is performed by BP.

This use of independent stack pointers allows independent stack areas to be secured for each mode and keeps the stack of one of the modes from being destroyed by an erroneous stack operation in the other mode.

The SP, IX, IY and AH registers and the four segment registers (PS, SS, DS_0, and DS_1) used in the native mode are not affected by operations in 8080 emulation mode.

In the 8080 emulation mode, the segment register for instructions is determined by the PS register (set automatically by the interrupt vector) and the segment register for data is the DS_0 register (set by the programmer immediately before the 8080 emulation mode is entered).

It is prohibited to nest BRKEM instructions.

RETEM [no operand]

When RETEM is executed in 8080 emulation mode (interpreted by the CPU as a μPD8080AF instruction), the CPU restores PS, PC, and PSW (as it would when returning from an interrupt processing routine), and returns to the native mode. At the same time, the contents of the mode flag (MD) which was saved to the stack by the BRKEM instruction, is restored to MD = 1. The CPU is set to the native mode.

CALLN imm8

This instruction makes it possible to call the native mode subroutines from the 8080 emulation mode. To return from subroutine to the 8080 emulation mode, the RETI instruction is used.

The processing performed when this instruction is executed in the 8080 emulation mode (it is interpreted by the CPU as μPD8080AF instruction), is similar to that performed when a BRK instruction is executed in the native mode. The imm8 operand specifies an interrupt vector type. The contents of PS, PC, and PSW are pushed on the stack and an MD flag value of 0 is saved. The mode flag is set to 1 and the interrupt vector specified by the operand is loaded into PS and PC.

RETI [no operand]

This is a general-purpose instruction used to return from interrupt routines entered by the BRK instruction or by an external interrupt in the native mode. When this instruction is executed at the end of a subroutine entered by the execution of the CALLN instruction, the operation that restores PS, PC, and PSW is exactly the same as the native mode execution. When PSW is restored, however, the 8080 emulation mode value of the mode flag (MD) is restored, the CPU is set in emulation mode, and all subsequent instructions are interpreted and executed as μPD8080AF instructions.

RETI is also used to return from an interrupt procedure initiated by an NMI or INT interrupt in the emulation mode.

Floating Point Operation Chip Instructions

FPO1 fp-op, mem

FPO2 fp-op, mem

These instructions are used for the external floating point processor. The floating point operation is passed to the floating point processor when the CPU fetches one of these instructions. From this point the CPU performs only the necessary auxiliary processing (effective address calculation, generation of physical addresses, and start-up of the memory read cycle).

The floating point processor always monitors the instructions fetched by the CPU. When it interprets one as an instruction to itself, it performs the appropriate processing. At this time, the floating point processor chip uses either the address alone or both the address and read data of the memory read cycle executed by the CPU. This difference in the data used depends on which of these instructions is executed.

Note: During the memory read cycle initiated by the CPU for FPO1 or FPO2 execution, the CPU does not accept any read data on the data bus from memory. Although the CPU generates the memory address, the data is used by the floating point processor.

Interrupt Operation

The interrupts used in the μPD70116 can be divided into two types: interrupts generated by external interrupt requests and interrupts generated by software processing. These are the classifications.

External Interrupts

(a) NMI input (nonmaskable)
(b) INT input (maskable)

Software processing

As the result of instruction execution

— When a divide error occurs during execution of the DIV or DIVU instruction
— When a memory-boundary-over error is detected by the CHKIND instruction

Conditional break instruction

— When V = 1 during execution of the BRKV instruction

Unconditional break instructions

— 1-byte break instruction: BRK3
— 2-byte break instruction: BRK imm8

Flag processing (Single-step)

— When stack operations are used to set the BRK flag

8080 Emulation mode instructions

— BRKEM imm8
— CALLN imm8

Starting addresses for interrupt processing routines are either determined automatically by a single location of the interrupt vector table or selected each time interrupt processing is entered.

The interrupt vector table is shown in figure 8. The table uses 1K bytes of memory addresses 000H to 3FFH and can store starting address data for a maximum of 256 vectors (4 bytes per vector).

The corresponding interrupt sources for vectors 0 to 5 are predetermined and vectors 6 to 31 are reserved. Consequently these vectors cannot be use for general applications.

The BRKEM instruction and CALLN instruction (in the emulation mode) and the INT input are available for general applications for vectors 32 to 255.

A single interrupt vector is made up of 4 bytes (figure 9). The 2 bytes in the low addresses of memory are loaded into PC as the offset, and the high 2 bytes are loaded into PS as the base address. The bytes are combined in reverse order. The lower-order bytes in the vector become the most significant bytes in the PC and PS, and the higher-order bytes become the least significant bytes.

Figure 8. Interrupt Vector Table

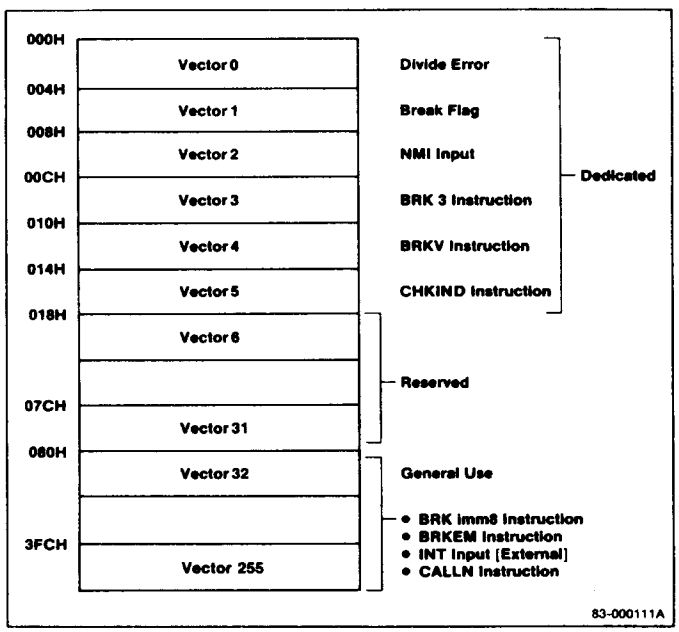
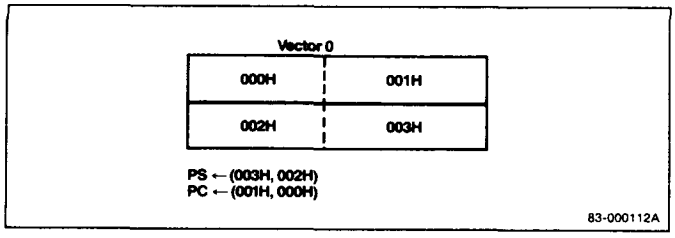


Figure 9. Interrupt Vector 0



Based on this format, the contents of each vector should be initialized at the beginning of the program.

The basic steps to jump to an interrupt processing routine are now shown.

(SP - 1, SP - 2) ← PSW
(SP - 3, SP - 4) ← PS
(SP - 5, SP - 6) ← PC
SP ← SP - 6
IE ← 0, BRK ← 0, MD ← 0
PS ← vector high bytes
PC ← vector low bytes

Standby Function

The μPD70116 has a standby mode to reduce power consumption during program wait states. This mode is set by the HALT instruction in both the native and the emulation mode.

In the standby mode, the internal clock is supplied only to those circuits related to functions required to release this mode and bus hold control functions. As a result, power consumption can be reduced to 1/10 the level of normal operation in either native or emulation mode.

The standby mode is released by inputting a RESET signal or an external interrupt (NMI, INT).

The bus hold function is effective during standby mode. The CPU returns to standby mode when the bus hold request is removed.

During standby mode, all control outputs are disabled and the addres/data bus will be at either high or low levels.

Instruction Set

Symbols

Preceding the instruction set, several tables explain symbols, abbreviations, and codes.

Clocks

In the Clocks column of the instruction set, the numbers cover these operations: instruction decoding, effective address calculation, operand fetch, and instruction execution.

Clock timings assume the instruction has been pre-fetched and is present in the four-byte instruction queue. Otherwise, add four clocks for each byte not present.

For instructions that reference memory operands, the number on the left side of the slash (/) is for byte operands and the number on the right side is for word operands.

For conditional control transfer or branch instructions, the number on the left side of the slash is applicable if the transfer or branch takes place. The number on the right side is applicable if it does not take place.

If a range of numbers is given, the execution time depends on the operands involved.

Symbols

Symbol	Meaning
acc	Accumulator (AW or AL)
disp	Displacement (8 or 16 bits)
dmem	Direct memory address
dst	Destination operand or address
dst-block	Name of block addressed by IY register
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
far_label	Label within a different program segment
far_proc	Procedure within a different program segment
fp_op	Floating point instruction operation
imm	8- or 16-bit immediate operand

Symbols

Symbol	Meaning
imm3/4	*3- or 4-bit immediate bit offset*
imm8	8-bit immediate operand
imm16	16-bit immediate operand
mem	Memory field (000 to 111); 8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
memptr16	Word containing the destination address within the current segment
memptr32	Double word containing a destination address in another segment
mod	Mode field (00 to 10)
near_label	Label within the current segment
near_proc	Procedure within the current segment
offset	Immediate offset data (16 bits)
pop_value	Number of bytes to discard from the stack
reg	Register field (000 to 111); 8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
regptr	16-bit register containing a destination address within the current segment
regptr16	Register containing a destination address within the current segment
seg	Immediate segment data (16 bits)
short_label	Label between −128 and +127 bytes from the end of the current instruction
sr	Segment register
src	Source operand or address
src-block	Name of block addressed by IX register
src-table	Name of 256-byte translation table
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1 bit)
AC	Auxiliary carry flag
AH	Accumulator (high byte)
AL	Accumulator (low byte)
AND ∧	Logical product
AW	Accumulator (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
BP	Base pointer (16 bits)
BRK	Break flag
BW	BW register (16 bits)
CH	CW register (high byte)

Symbol	Meaning
CL	*CW register (low byte)*
CW	CW register (16 bits)
CY	Carry flag
DH	DW register (high byte)
DIR	Direction flag
DL	DW register (low byte)
DS0	Data segment 0 register (16 bits)
DS1	Data segment 1 register (16 bits)
DW	DW register (16 bits)
IE	Interrupt enable flag
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
MD	Mode flag
OR ∨	Logical sum
P	Parity flag
PC	Program counter (16 bits)
PS	Program segment register (16 bits)
PSW	Program status word (16 bits)
R	Register set
S	Sign extend operand field *S = 0 No sign extension* S = 1 Sign extend immediate byte operand
S	Sign flag
SP	Stack pointer (16 bits)
SS	Stack segment register (16 bits)
TA	Temporary register A (16 bits)
TB	Temporary register B (16 bits)
TC	Temporary register C (16 bits)
V	Overflow flag
W	Word/byte field (0 to 1)
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
XOR ∨̄	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value
Z	Zero flag
( )	Values in parentheses are memory contents
←	Transfer direction
+	Addition
−	Subtraction
x	Multiplication
÷	Division
%	Modulo

Flag Operations

Symbol	Meaning
(blank)	No change
0	Cleared to 0
1	Set to 1
x	Set or cleared according to result
u	Undefined
R	Restored to previous state

Memory Addressing Modes

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Register Selection (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Register Selection

sr	Segment Register
00	DS1
01	PS
10	SS
11	DS0

Instruction Set

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Transfer Instructions																									
MOV	reg, reg	1	0	0	0	1	0	1	W	1	1		reg			reg		2	2						
	mem, reg	1	0	0	0	1	0	0	W	mod			reg			mem		9/13	2-4						
	reg, mem	1	0	0	0	1	0	1	W	mod			reg			mem		11/15	2-4						
	mem, imm	1	1	0	0	0	1	1	W	mod			0	0	0	mem		11/15	3-6						
	reg, imm	1	0	1	1	W		reg										4	2-3						
	acc, dmem	1	0	1	0	0	0	0	W									10/14	3						
	dmem, acc	1	0	1	0	0	0	1	W									9/13	3						
	sr, reg16	1	0	0	0	1	1	1	0	1	1	0	sr			reg		2	2						
	sr, mem16	1	0	0	0	1	1	1	0	mod		0	sr			mem		11/15	2-4						
	reg16, sr	1	0	0	0	1	1	0	0	1	1	0	sr			reg		2	2						
	mem16, sr	1	0	0	0	1	1	0	0	mod		0	sr			mem		10/14	2-4						
	DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod			reg			mem		18/26	2-4						
	DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod			reg			mem		18/26	2-4						
	AH, PSW	1	0	0	1	1	1	1	1									2	1						
	PSW, AH	1	0	0	1	1	1	1	0									3	1	x	x		x	x	x
LDEA	reg16, mem16	1	0	0	0	1	1	0	1	mod			reg			mem		4	2-4						
TRANS	src_table	1	1	0	1	0	1	1	1									9	1						
XCH	reg, reg	1	0	0	0	0	1	1	W	1	1		reg			reg		3	2						
	mem, reg	1	0	0	0	0	1	1	W	mod			reg			mem		16/24	2-4						
	AW, reg16	1	0	0	1	0		reg										3	1						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7 6 5 4 3 2 1 0	Opcode 7 6 5 4 3 2 1 0	Clocks	Bytes	Flags AC CY V P S Z
Repeat Prefixes						
REPC		0 1 1 0 0 1 0 1		2	1	
REPNC		0 1 1 0 0 1 0 0		2	1	
REP REPE REPZ		1 1 1 1 0 0 1 1		2	1	
REPNE REPNZ		1 1 1 1 0 0 1 0		2	1	
Block Transfer Instructions						
MOVBK	dst, src	1 0 1 0 0 1 0 W		11 + 8n	1	
CMPBK	dst, src	1 0 1 0 0 1 1 W		7 + 14n	1	x x x x x x
CMPM	dst	1 0 1 0 1 1 1 W		7 + 10n	1	x x x x x x
LDM	src	1 0 1 0 1 1 0 W		7 + 9n	1	
STM	dst	1 0 1 0 1 0 1 W		7 + 4n	1	
			n = number of transfers			
I/O Instructions						
IN	acc, imm8	1 1 1 0 0 1 0 W		9/13	2	
	acc, DW	1 1 1 0 1 1 0 W		8/12	1	
OUT	imm8, acc	1 1 1 0 0 1 1 W		8/12	2	
	DW, acc	1 1 1 0 1 1 1 W		8/12	1	
INM	dst, DW	0 1 1 0 1 1 0 W		9 + 8n	1	
OUTM	DW, src	0 1 1 0 1 1 1 W		9 + 8n	1	
			n = number of transfers			
BCD Instructions						
ADJBA		0 0 1 1 0 1 1 1		3	1	x x u u u u
ADJ4A		0 0 1 0 0 1 1 1		3	1	x x u x x x
ADJBS		0 0 1 1 1 1 1 1		7	1	x x u u u u
ADJ4S		0 0 1 0 1 1 1 1		7	1	x x u x x x
ADD4S	dst, src	0 0 0 0 1 1 1 1	0 0 1 0 0 0 0 0	7 + 19n	2	u x u u u x
SUB4S	dst, src	0 0 0 0 1 1 1 1	0 0 1 0 0 0 1 0	7 + 19n	2	u x u u u x
CMP4S	dst, src	0 0 0 0 1 1 1 1	0 0 1 0 0 1 1 0	7 + 19n	2	u x u u u x
ROL4	reg8	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 0 1 0 0 0	25	3	
	mem8	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 1 0 1 0 0 0	28	3-5	
ROR4	reg8	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 0 1 0 1 0	29	3	
	mem8	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 1 0 1 0 1 0	33	3-5	
			n = number of BCD digits divided by 2			

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Type Conversion Instructions																									
CVTBD		1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	15	2	u	u	u	x	x	x
CVTDB		1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	7	2	u	u	u	x	x	x
CVTBW		1	0	0	1	1	0	0	0									2	1						
CVTWL		1	0	0	1	1	0	0	1									4-5	1						
Arithmetic Instructions																									
ADD	reg, reg	0	0	0	0	0	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	0	0	0	0	0	W	mod		reg			mem			16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	0	0	0	1	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	0	0	reg			4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	0	0	mem			18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	0	0	1	0	W									4	2-3	x	x	x	x	x	x
ADDC	reg, reg	0	0	0	1	0	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	0	0	0	W	mod		reg			mem			16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	0	0	1	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	0	reg			4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	0	mem			18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	0	1	0	W									4	2-3	x	x	x	x	x	x
SUB	reg, reg	0	0	1	0	1	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	1	0	1	0	0	W	mod		reg			mem			16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	0	1	0	1	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	0	1	reg			4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	0	1	mem			18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	0	1	1	0	W									4	2-3	x	x	x	x	x	x
SUBC	reg, reg	0	0	0	1	1	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	1	0	0	W	mod		reg			mem			16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	1	0	1	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	1	reg			4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	1	mem			18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	1	1	0	W									4	2-3	x	x	x	x	x	x
INC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	0	reg			2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	0	mem			16/24	2-4	x		x	x	x	x
	reg16	0	1	0	0	0	reg											2	1	x		x	x	x	x
DEC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	1	reg			2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	1	mem			16/24	2-4	x		x	x	x	x
	reg16	0	1	0	0	1	reg											2	1	x		x	x	x	x
MULU	reg	1	1	1	1	0	1	1	W	1	1	1	0	0	reg			21-30	2	u	x	x	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	0	0	mem			27-36	2-4	u	x	x	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Arithmetic Instructions (cont)																									
MUL	reg	1	1	1	1	0	1	1	W	1	1	1	0	1	reg			33-47	2	u	x	x	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	0	1	mem			39-53	2-4	u	x	x	u	u	u
	reg16,reg16,imm8	0	1	1	0	1	0	1	1	1	1	reg			reg			28-34	3	u	x	x	u	u	u
	reg16,mem16,imm8	0	1	1	0	1	0	1	1	mod		reg			mem			34-40	3-5	u	x	x	u	u	u
	reg16,reg16,imm16	0	1	1	0	1	0	0	1	1	1	reg			reg			36-42	4	u	x	x	u	u	u
	reg16,mem16,imm16	0	1	1	0	1	0	0	1	mod		reg			mem			46-48	4-6	u	x	x	u	u	u
DIVU	reg	1	1	1	1	0	1	1	W	1	1	1	1	0	reg			19-25	2	u	u	u	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	1	0	mem			25-31	2-4	u	u	u	u	u	u
DIV	reg	1	1	1	1	0	1	1	W	1	1	1	1	1	reg			29-43	2	u	u	u	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	1	1	mem			35-49	2-4	u	u	u	u	u	u
Comparison Instructions																									
CMP	reg, reg	0	0	1	1	1	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	1	1	1	0	0	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	1	1	0	1	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	1	1	reg			4	3-4	x	x	x	x	x	x
	cem, imm	1	0	0	0	0	0	S	W	mod		1	1	1	mem			13/17	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	1	1	1	0	W									4	2-3	x	x	x	x	x	x
Logical Instructions																									
NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0	reg			2	2						
	mem	1	1	1	1	0	1	1	W	mod		0	1	0	mem			16/24	2-4						
NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1	reg			2	2	x	x	x	x	x	x
	mem	1	1	1	1	0	1	1	W	mod		0	1	1	mem			16/24	2-4	x	x	x	x	x	x
TEST	reg, reg	1	0	0	0	0	1	0	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	1	0	0	0	0	1	0	W	mod		reg			mem			10/14	2-4	u	0	0	x	x	x
	reg, imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem			11/15	3-6	u	0	0	x	x	x
	acc, imm	1	0	1	0	1	0	0	W									4	2-3	u	0	0	x	x	x
AND	reg, reg	0	0	1	0	0	0	1	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	1	0	0	0	0	W	mod		reg			mem			16/24	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	0	0	0	1	W	mod		reg			mem			11/15	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	0	0	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	0	0	mem			18/26	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	0	0	1	0	W									4	2-3	u	0	0	x	x	x
OR	reg, reg	0	0	0	0	1	0	1	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	0	0	1	0	0	W	mod		reg			mem			16/24	2-4	u	0	0	x	x	x
	reg, mem	0	0	0	0	1	0	1	W	mod		reg			mem			11/15	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	0	0	1	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		0	0	1	mem			18/26	3-6	u	0	0	x	x	x
	acc, imm	0	0	0	0	1	1	0	W									4	2-3	u	0	0	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Logical Instructions (cont)																									
XOR	reg, reg	0	0	1	1	0	0	1	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	1	1	0	0	0	W	mod		reg			mem			16/24	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	1	0	0	1	W	mod		reg			mem			11/15	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	1	0	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	1	0	mem			18/26	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	1	0	1	0	W									4	2-3	u	0	0	x	x	x
Bit Manipulation Instructions																									
INS	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	31-117	3						
		1	1		reg			reg																	
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	31-117	4						
		1	1	0	0	0		reg																	
EXT	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	26-55	3						
		1	1		reg			reg																	
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1	26-55	4						
		1	1	0	0	0		reg																	
TEST1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	3	3	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	12	3-5	u	0	0	u	u	x
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	4	4	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	13	4-6	u	0	0	u	u	x
		mod		0	0	0		mem																	
SET1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	4	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	13	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	5	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	14	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	1									2	1		1				
	DIR	1	1	1	1	1	1	0	1									2	1						
CLR1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	5	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	14	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	6	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	15	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	0									2	1		0				
	DIR	1	1	1	1	1	1	0	0									2	1						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Bit Manipulation Instructions (cont)																									
NOT1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	4	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	18	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	5	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	19	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	0	1	0	1									2	1		x				
Shift/Rotate Instructions																									
SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	0	reg			2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod			1	0	0	mem		16/24	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0	reg			7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod			1	0	0	mem		19 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0	reg			7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod			1	0	0	mem		19 + n	3-5	u	x	u	x	x	x
SHR	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	1	reg			2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod			1	0	1	mem		16/24	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	1	reg			7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod			1	0	1	mem		19 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	1	reg			7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod			1	0	1	mem		19 + n	3-5	u	x	u	x	x	x
										n = number of shifts															
SHRA	reg, 1	1	1	0	1	0	0	0	W	1	1	1	1	1	reg			2	2	u	x	0	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod			1	1	1	mem		16/24	2-4	u	x	0	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	1	1	reg			7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod			1	1	1	mem		19 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	1	1	reg			7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod			1	1	1	mem		19 + n	3-5	u	x	u	x	x	x
ROL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	0	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod			0	0	0	mem		16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	0	reg			7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod			0	0	0	mem		19 + n	2-4		x	u			
	reg, imm	1	1	0	0	0	0	0	W	1	1	0	0	0	reg			7 + n	3		x	u			
	mem, imm	1	1	0	0	0	0	0	W	mod			0	0	0	mem		19 + n	3-5		x	u			
ROR	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	1	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod			0	0	1	mem		16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	1	reg			7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod			0	0	1	mem		19 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	1	reg			7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod			0	0	1	mem		19 + n	3-5		x	u			

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Shift/Rotate Instructions (cont)																									
ROLC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	0		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	0		mem		16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	0		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	0		mem		19 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	0		reg		7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	0		mem		19 + n	3-5		x	u			
RORC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	1		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	1		mem		16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	1		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	1		mem		19 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	1		reg		7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	1		mem		19 + n	3-5		x	u			
										n = number of shifts															
Stack Manipulation Instructions																									
PUSH	mem16	1	1	1	1	1	1	1	1	mod		1	1	0		mem		18/26	2-4						
	reg16	0	1	0	1	0		reg										8/12	1						
	sr	0	0	0		sr		1	1	0								8/12	1						
	PSW	1	0	0	1	1	1	0	0									8/12	1						
	R	0	1	1	0	0	0	0	0									35/67	1						
	imm	0	1	1	0	1	0	S	0									7-8	2-3						
POP	mem16	1	0	0	0	1	1	1	1	mod	0	0	0		mem			17/25	2-4						
	reg16	0	1	0	1	1		reg										8/12	1						
	sr	0	0	0		sr		1	1	1								8/12	1						
	PSW	1	0	0	1	1	1	0	1									8/12	1	R	R	R	R	R	
	R	0	1	1	0	0	0	0	1									43/75	1						
PREPARE	imm16, imm8	1	1	0	0	1	0	0	0									*	4						
										*imm8 = 0 : 12/16 imm8 ≥ 1 : 19 + 8 (imm8 − 1): even address 23 + 16 (imm8-1): odd address														
DISPOSE		1	1	0	0	1	0	0	1									6/10	1						
Control Transfer Instructions																									
CALL	near_proc	1	1	1	0	1	0	0	0									16/20	3						
	regptr	1	1	1	1	1	1	1	1	1	1	0	1	0		reg		14/18	2						
	memptr16	1	1	1	1	1	1	1	1	mod		0	1	0		mem		23/31	2-4						
	far_proc	1	0	0	1	1	0	1	0									21/29	5						
	memptr32	1	1	1	1	1	1	1	1	mod		0	1	1		mem		31/47	2-4						
RET		1	1	0	0	0	0	1	1									15/19	1						
	pop_value	1	1	0	0	0	0	1	0									20/24	3						
		1	1	0	0	1	0	1	1									21/29	1						
	pop_value	1	1	0	0	1	0	1	0									24/32	3						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Control Transfer Instructions (cont)																									
BR	near_label	1	1	1	0	1	0	0	1		13	3													
	short_label	1	1	1	0	1	0	1	1		12	2													
	regptr	1	1	1	1	1	1	1	1	1	1	1	0	0		reg		11	2						
	memptr16	1	1	1	1	1	1	1	1	mod		1	0	0		mem		20/24	2-4						
	far_label	1	1	1	0	1	0	1	0									15	5						
	memptr32	1	1	1	1	1	1	1	1	mod		1	0	1		mem		27/35	2-4						
BV	short_label	0	1	1	1	0	0	0	0									14/4	2						
BNV	short_label	0	1	1	1	0	0	0	1									14/4	2						
BC, BL	*short_label*	*0*	*1*	*1*	*1*	*0*	*0*	*1*	*0*									*14/4*	*2*						
BNC, BNL	short_label	0	1	1	1	0	0	1	1									14/4	2						
BE, BZ	short_label	0	1	1	1	0	1	0	0									14/4	2						
BNE, BNZ	short_label	0	1	1	1	0	1	0	1									14/4	2						
BNH	short_label	0	1	1	1	0	1	1	0									14/4	2						
BH	short_label	0	1	1	1	0	1	1	1									14/4	2						
BN	short_label	0	1	1	1	1	0	0	0									14/4	2						
BP	short_label	0	1	1	1	1	0	0	1									14/4	2						
BPE	short_label	0	1	1	1	1	0	1	0									14/4	2						
BPO	short_label	0	1	1	1	1	0	1	1									14/4	2						
BLT	short_label	0	1	1	1	1	1	0	0									14/4	2						
BGE	short_label	0	1	1	1	1	1	0	1									14/4	2						
BLE	short_label	0	1	1	1	1	1	1	0									14/4	2						
BGT	short_label	0	1	1	1	1	1	1	1									14/4	2						
DBNZNE	short_label	1	1	1	0	0	0	0	0									14/5	2						
DBNZE	short_label	1	1	1	0	0	0	0	1									14/5	2						
DBNZ	short_label	1	1	1	0	0	0	1	0									13/5	2						
BCWZ	short_label	1	1	1	0	0	0	1	1									13/5	2						
Interrupt Instructions																									
BRK	3	1	1	0	0	1	1	0	0									38/50	1						
	imm8	1	1	0	0	1	1	0	1									38/50	2						
BRKV	imm8	1	1	0	0	1	1	1	0									40/3	1						
RETI		1	1	0	0	1	1	1	1									27/39	1	R	R	R	R	R	R
CHKIND	reg16, mem32	0	1	1	0	0	0	1	0	mod			reg			mem		53-56/18	2-4						
BRKEM	imm8	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	38/50	3						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
CPU Control Instructions																									
HALT		1	1	1	1	0	1	0	0									2	1						
BUSLOCK		1	1	1	1	0	0	0	0									2	1						
FPO1	fp_op	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z	2	2						
	fp_op, mem	1	1	0	1	1	X	X	X	mod		Y	Y	Y		mem		11/15	2-4						
FPO2	fp_op	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z	2	2						
	fp_op, mem	0	1	1	0	0	1	1	X	mod		Y	Y	Y		mem		11/15	2-4						
POLL		1	0	0	1	1	0	1	1									2 + 5n	1						
		n = number of times POLL pin is sampled.																							
NOP		1	0	0	1	0	0	0	0									3	1						
DI		1	1	1	1	1	0	1	0									2	1						
EI		1	1	1	1	1	0	1	1									2	1						
8080 Instruction Set Enhancements																									
RETEM		1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	1	27/39	2	R	R	R	R	R	R
CALLN	imm8	1	1	1	0	1	1	0	1	1	1	1	0	1	1	0	1	38/58	3						