OKI semiconductor

Preliminary

# MSM80C86A-10RS/GS/JS

## 16-BIT CMOS MICROPROCESSOR

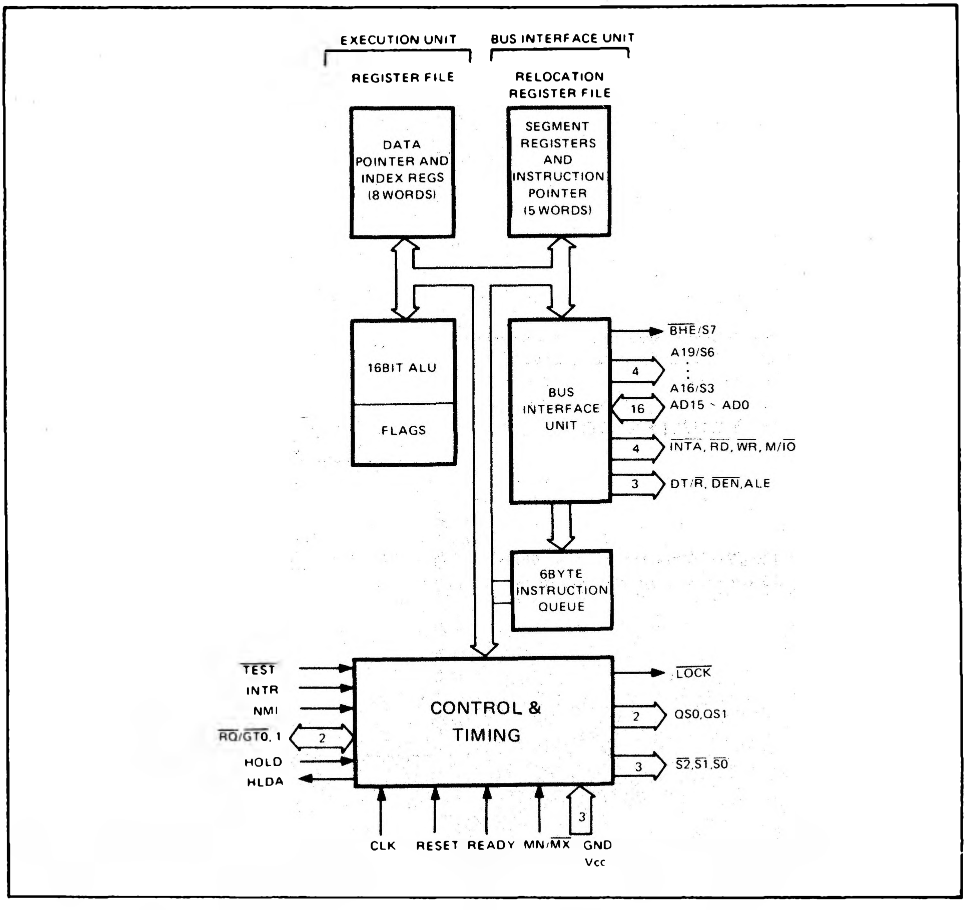
## GENERAL DESCRIPTION

The MSM80C86A-10 are complete 16-bit CPUs implemented in Silicon Gate CMOS technology. They are designed with same processing speed as the NMOS 8086-1 but have considerably less power consumption. They are directly compatible with MSM80C88A-10 software and MSM80C85A/MSM80C85A-2 hardware and peripherals.

## FEATURES

- 1 Mbyte Direct Addressable Memory Space
- Internal 14 Word by 16-bit Register Set
- 24 Operand Addressing Modes
- Bit, Byte, Word and String Operations
- 8 and 16-bit Signed and Unsigned Arithmetic Operation
- From DC to 10 MHz Clock Rate
- Low Power Dissipation 10 mA/MHz
- Bus Hold Circuitry Eliminates Pull-Up Resistors

## CIRCUIT CONFIGURATION



## PIN CONFIGURATION

**MSM80C86A-10RS (Top View)**
**40 Lead Plastic DIP**

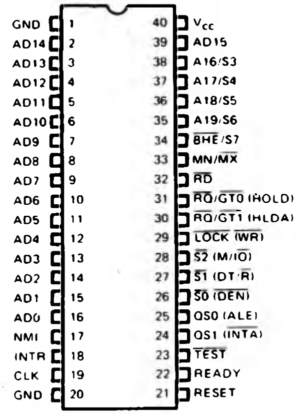



**Fig. 2a MSM80C86A-10RS**

**MSM80C86A-10GS (Top View)**
**56 Lead Plastic Flat Package**

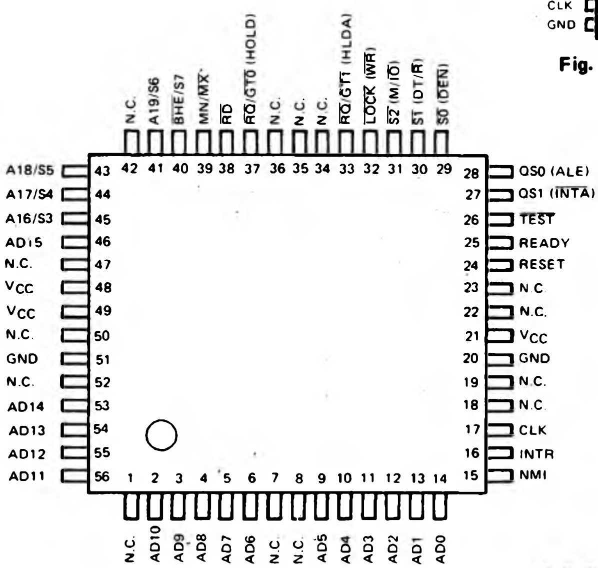
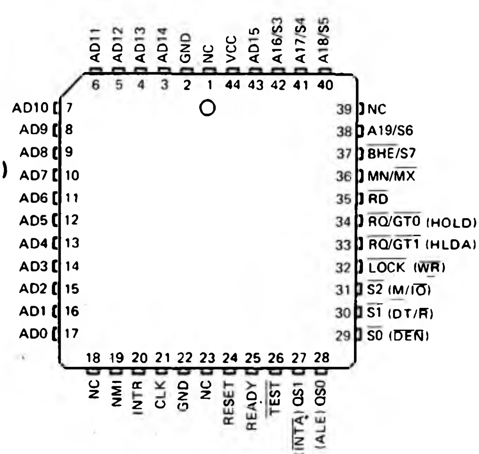



**Fig. 2b MSM80C86A-10GS**

**MSM80C86A-10JS (Top View)**
**44-pin Plastic Leaded Chip Carrier**



## ABSOLUTE MAXIMUM RATINGS

| Parameter | Symbol | Limits | | | Unit | Conditions |
|---|---|---|---|---|---|---|
| | | MSM80C86A-10RS | MSM80C86A-10GS | MSM80C86A-10JS | | |
| Power Supply Voltage | $V_{CC}$ | -0.5 ~ +7 | | | V | With respect to GND |
| Input Voltage | $V_{IN}$ | -0.5 ~ $V_{CC}$ +0.5 | | | V | |
| Output Voltage | $V_{OUT}$ | -0.5 ~ $V_{CC}$ +0.5 | | | V | |
| Storage Temperature | Tstg | -65 ~ +150 | | | °C | – |
| Power Dissipation | $P_D$ | 1.0 | 0.7 | | W | Ta = 25°C |

## OPERATING RANGE

| Parameter | Symbol | Limits | Unit |
|---|---|---|---|
| | | MSM80C86A-10 | |
| Power Supply Voltage | $V_{CC}$ | 4.75 ~ 5.25 | V |
| Operating Temperature | $T_{OP}$ | 0 ~ +70 | °C |

## RECOMMENDED OPERATING CONDITIONS

| Parameter | Symbol | MSM80C86A-10 | | | Unit |
|---|---|---|---|---|---|
| | | MIN | TYP | MAX | |
| Power Supply Voltage | $V_{CC}$ | 4.75 | 5.0 | 5.25 | V |
| Operating Temperature | $T_{OP}$ | 0 | +25 | +70 | °C |
| "L" Input Voltage | $V_{IL}$ | -0.5 | | +0.8 | V |
| "H" Input Voltage | $V_{IH}$ (*1) | $V_{CC}$-0.8 | | $V_{CC}$ +0.5 | V |
| | $V_{IH}$ (*2) | 2.0 | | $V_{CC}$ +0.5 | V |

*1 Only CLK, *2 Except CLK.

## DC CHARACTERISTICS

(MSM80C86A-10: $V_{CC}$ = 4.75 to 5.25V, Ta = 0°C to +70°C)

| Parameter | Symbol | MIN | TYP | MAX | Unit | Conditions |
|---|---|---|---|---|---|---|
| "L" Output Voltage | $V_{OL}$ | | | 0.4 | V | $I_{OL}$ = 2.5 mA |
| "H" Output Voltage | $V_{OH}$ | 3.0 | | | V | $I_{OH}$ = −2.5 mA |
| | | $V_{CC}$−0.4 | | | | $I_{OH}$ = −100 μA |
| Input Leak Current | $I_{LI}$ | −1.0 | | +1.0 | μA | $0 < V_I < V_{CC}$ |
| Output Leak Current | $I_{LO}$ | −10 | | +10 | μA | $V_O = V_{CC}$ or GND |
| Input Leakage Current (Bus Hold Low) | $I_{BHL}$ | 50 | | 400 | μA | $V_{IN}$ = 0.8V *3 |
| Input Leakage Current (Bus Hold High) | $I_{BHH}$ | −50 | | −400 | μA | $V_{IN}$ = 3.0V *4 |
| Bus Hold Low Overdrive | $I_{BHLO}$ | | | 600 | μA | *5 |
| Bus Hold High Overdrive | $I_{BHHO}$ | | | −600 | μA | *6 |
| Operating Power Supply Current | $I_{CC}$ | | | 10 | mA/MHz | $V_{IL}$ = GND, $V_{IH} = V_{CC}$ |
| Standby Power Supply Current | $I_{CCS}$ | | | 500 | μA | $V_{CC}$ = 5.5V, Outputs Unloaded, $V_{IN} = V_{CC}$ or GND |
| Input Capacitance | $C_{in}$ | | | 5 | pF | *7 |
| Output Capacitance | $C_{out}$ | | | 15 | pF | *7 |
| I/O Capacitance | $C_{I/O}$ | | | 20 | pF | *7 |

*3 Test condition is to lower $V_{IN}$ to GND and then raise $V_{IN}$ to 0.8V on pins 2–16, and 35–39

*4 Test condition is to raise $V_{IN}$ to $V_{CC}$ and then lower $V_{IN}$ to 3.0V on pins 2–16, 26–32, and 34–39.

*5 An external driver must source at least $I_{BHLO}$ to switch this node from LOW to HIGH.

*6 An external driver must sink at least $I_{BHHO}$ to switch this node from HIGH to LOW.

*7 Test Conditions: a) Freq = 1 MHz.
b) Ummeasured Pins at GND.
c) $V_{IN}$ at 5.0V or GND.

## A.C. CHARACTERISTICS

(MSM80C86A-10: $V_{CC}$ = 4.75V to 5.25V, Ta = 0°C to 70°C)

**Minimum Mode System**
**Timing Requirements**

| Parameter | Symbol | MSM80C86A-10 | | Unit |
|---|---|---|---|---|
| | | Min. | Max. | |
| CLK Cycle Period | TCLCL | 100 | DC | ns |
| CLK Low Time | TCLCH | 46 | | ns |
| CLK High Time | TCHCL | 44 | | ns |
| CLK Rise Time (From 1.0V to 3.5V) | TCH1CH2 | | 10 | ns |
| CLK Fall Time (From 3.5V to 1.0V) | TCL2CL1 | | 10 | ns |
| Data in Setup Time | TDVCL | 20 | | ns |
| Data in Hold Time | TCLDX | 10 | | ns |
| RDY Setup Time into MSM 82C84A (See Notes 1, 2) | TR1VCL | 35 | | ns |
| RDY Hold Time into MSM 82C84A (See Notes 1, 2) | TCLR1X | 0 | | ns |
| READY Setup Time into MSM80C86A | TRYHCH | 46 | | ns |
| READY Hold Time into MSM80C86A-2 | TCHRYX | 20 | | ns |
| READY inactive to CLK (See Note 3) | TRYLCL | -8 | | ns |
| HOLD Setup Time | THVCH | 20 | | ns |
| INTR, NMI, $\overline{\text{TEST}}$ Setup Time (See Note 2) | TINVCH | 15 | | ns |
| Input Rise Time (Except CLK) (From 0.8V to 2.0V) | TILIH | | 15 | ns |
| Input Fall Time (Except CLK) (From 2.0V to 0.8V) | TIHIL | | 15 | ns |

**Timing Responses**

| Parameter | Symbol | MSM80C86A-10 | | Unit |
|---|---|---|---|---|
| | | Min. | Max. | |
| Address Valid Delay | TCLAV | 10 | 60 | ns |
| Address Hold Time | TCLAX | 10 | | ns |
| Address Float Delay | TCLAZ | TCLAX | 50 | ns |
| ALE Width | TLHLL | TCLCH-10 | | ns |
| ALE Active Delay | TCLLH | | 40 | ns |
| ALE Inactive Delay | TCHLL | | 45 | ns |
| Address Hold Time to ALE Inactive | TLLAX | TCHCL-10 | | ns |
| Data Valid Delay | TCLDV | 10 | 60 | ns |
| Data Hold Time | TCHDX | 10 | | ns |
| Data Hold Time after $\overline{\text{WR}}$ | TWHDX | TCLCH-25 | | ns |
| Control Active Delay 1 | TCVCTV | 10 | 55 | ns |
| Control Active Delay 2 | TCHCTV | 10 | 50 | ns |
| Control Inactive Delay | TCVCTX | 10 | 55 | ns |
| Address Float to $\overline{\text{RD}}$ Active | TAZRL | 0 | | ns |
| $\overline{\text{RD}}$ Active Delay | TCLRL | 10 | 70 | ns |

| Parameter | Symbol | MSM80C86A-10 | | Unit |
|---|---|---|---|---|
| | | Min. | Max. | |
| $\overline{RD}$ Inactive Delay | TCLRH | 10 | 60 | ns |
| $\overline{RD}$ Inactive to Next Address Active | TRHAV | TCLCL-35 | | ns |
| HLDA Valid Delay | TCLHAV | 10 | 60 | ns |
| $\overline{RD}$ Width | TRLRH | 2TCLCL-40 | | ns |
| $\overline{WR}$ Width | TWLWH | 2TCLCL-35 | | ns |
| Address Valid to ALE Low | TAVAL | TCLCH-35 | | ns |
| Output Rise Time (From 0.8V to 2.0V) | TOLOH | | 15 | ns |
| Output Fall Time (From 2.0V to 0.8V) | TOHOL | | 15 | ns |

**Notes:**
1. Signal at MSM 82C84A or MSM 82C88 are shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state. (8 ns into T3)

**Maximum Mode System** (Using MSM 82C88 Bus Controller)
**Timing Requirements**

| Parameter | Symbol | MSM80C86A-10 | | Unit |
|---|---|---|---|---|
| | | Min. | Max. | |
| CLK Cycle Period | TCLCL | 100 | DC | ns |
| CLK Low Time | TCLCH | 46 | | ns |
| CLK High Time | TCHCL | 44 | | ns |
| CLK Rise Time (From 1.0V to 3.5V) | TCH1CH2 | | 10 | ns |
| CLK Fall Time (From 3.5V to 1.0V) | TCL2CL1 | | 10 | ns |
| Data in Setup Time | TDVCL | 15 | | ns |
| Data in Hold Time | TCLDX | 10 | | ns |
| RDY Setup Time into MSM 82C84A (See Notes 1, 2) | TR1VCL | 35 | | ns |
| RDY Hold Time into MSM 82C84A (See Notes 1, 2) | TCLR1X | 0 | | ns |
| READY Setup Time into MSM 80C86A | TRYHCH | 46 | | ns |
| READY Hold Time into MSM 80C86A | TCHRYX | 20 | | ns |
| READY inactive to CLK (See Note 3) | TRYLCL | -8 | | ns |
| Set up Time for Recognition (NMI, INTR, $\overline{TEST}$) (See Note 2) | TINVCH | 15 | | ns |
| $\overline{RQ}/\overline{GT}$ Setup Time | TGVCH | 15 | | ns |
| $\overline{RQ}$ Hold Time into MSM 80C86A | TCHGX | 20 | | ns |
| Input Rise Time (Except CLK) (From 0.8V to 2.0V) | TILIH | | 15 | ns |
| Input Fall Time (Except CLK) (From 2.0 to 0.8V) | TIHIL | | 15 | ns |

**Timing Responses**

| Parameter | Symbol | MSM80C86A-10 | | Unit |
|---|---|---|---|---|
| | | Min. | Max. | |
| Command Active Delay (See Note 1) | TCLML | 5 | 35 | ns |
| Command Inactive Delay (See Note 1) | TCLMH | 5 | 45 | ns |
| READY Active to Status Passive (See Note 4) | TRYHSH | | 45 | ns |
| Status Active Delay | TCHSV | 10 | 45 | ns |
| Status Inactive Delay | TCLSH | 10 | 60 | ns |
| Address Valid Delay | TCLAV | 10 | 60 | ns |
| Address Hold Time | TCLAX | 10 | | ns |
| Address Float Delay | TCLAZ | TCLAX | 50 | ns |
| Status Valid to ALE High (See Note 1) | TSVLH | | 25 | ns |
| Status Valid to MCE High (See Note 1) | TSVMCH | | 30 | ns |
| CLK low to ALE Valid (See Note 1) | TCLLH | | 25 | ns |
| CLK Low to MCE High (See Note 1) | TCLMCH | | 25 | ns |
| ALE Inactive Delay (See Note 1) | TCHLL | 4 | 25 | ns |
| Data Valid Delay | TCLDV | 10 | 60 | ns |
| Data Hold Time | TCHDX | 10 | | ns |

| Parameter | Symbol | MSM80C86A-10 | | Unit |
|---|---|---|---|---|
| | | Min. | Max. | |
| Control Active Delay (See Note 1) | TCVNV | 5 | 45 | ns |
| Control Inactive Delay (See Note 1) | TCVNX | 10 | 45 | ns |
| Address Float to $\overline{RD}$ Active | TAZRL | 0 | | ns |
| $\overline{RD}$ Active Delay | TCLRL | 10 | 70 | ns |
| $\overline{RD}$ Inactive Delay | TCLRH | 10 | 60 | ns |
| $\overline{RD}$ Inactive to Next Address Active | TRHAV | TCLCL-35 | | ns |
| Direction Control Active Delay (See Note 1) | TCHDTL | | 50 | ns |
| Direction Control Inactive Delay (See Note 1) | TCHDTH | | 30 | ns |
| $\overline{GT}$ Active Delay | TCLGL | 0 | 45 | ns |
| $\overline{GT}$ Inactive Delay | TCLGH | 0 | 45 | ns |
| $\overline{RD}$ Width | TRLRH | 2TCLCL-40 | | ns |
| Output Rise Time (From 0.8V to 2.0V) | TOLOH | | 15 | ns |
| Output Fall Time (From 2.0V to 0.8V) | TOHOL | | 15 | ns |

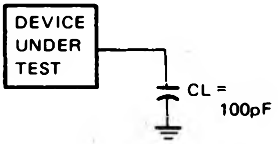
**Notes:**
1. Signal at MSM 82C84A or MSM 82C88 are shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state (8 ns into T3)
4. Applies only to T3 and wait states.

## TIMING CHART

**Input/Output**

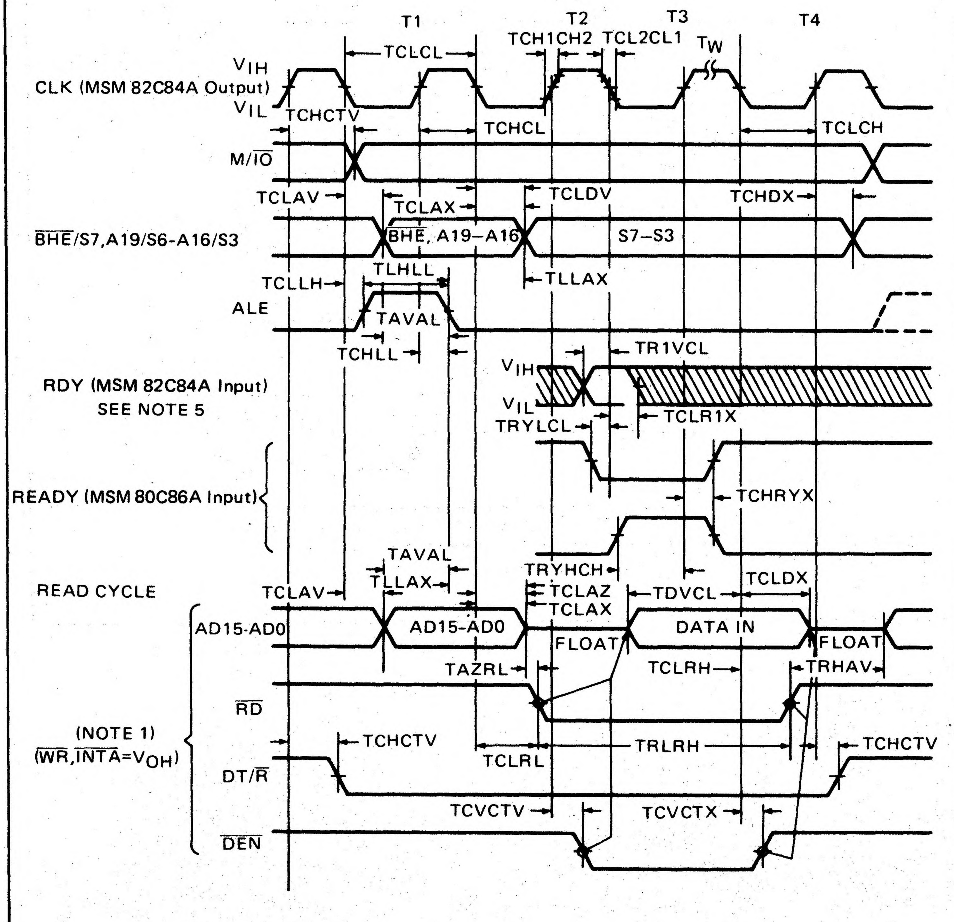

A.C. TESTING: INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0" TIMING MEASUREMENTS ARE 1.5V FOR BOTH A LOGIC "1" AND "0"

**A.C. Testing Load Circuit**

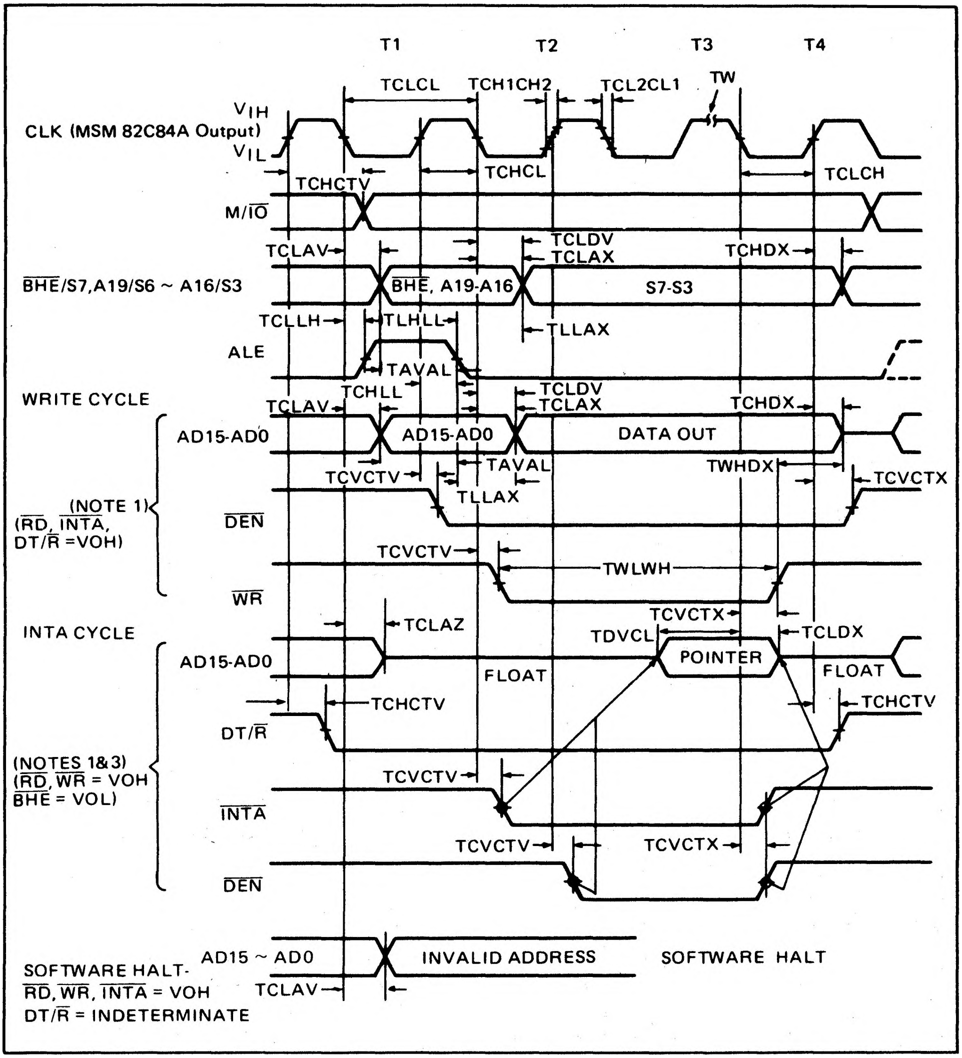



CL INCLUDES JIG CAPACITANCE

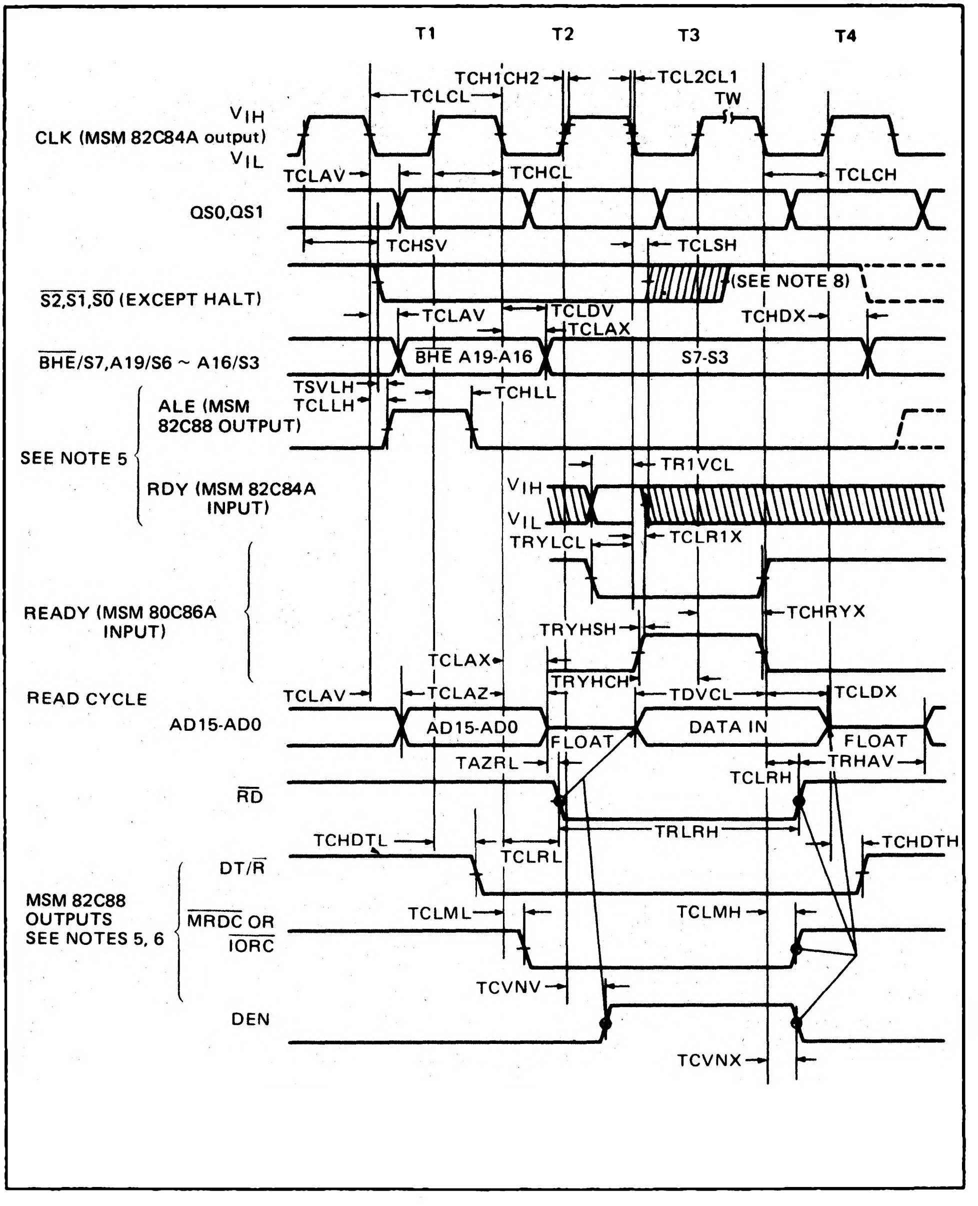
**Minimum Mode**



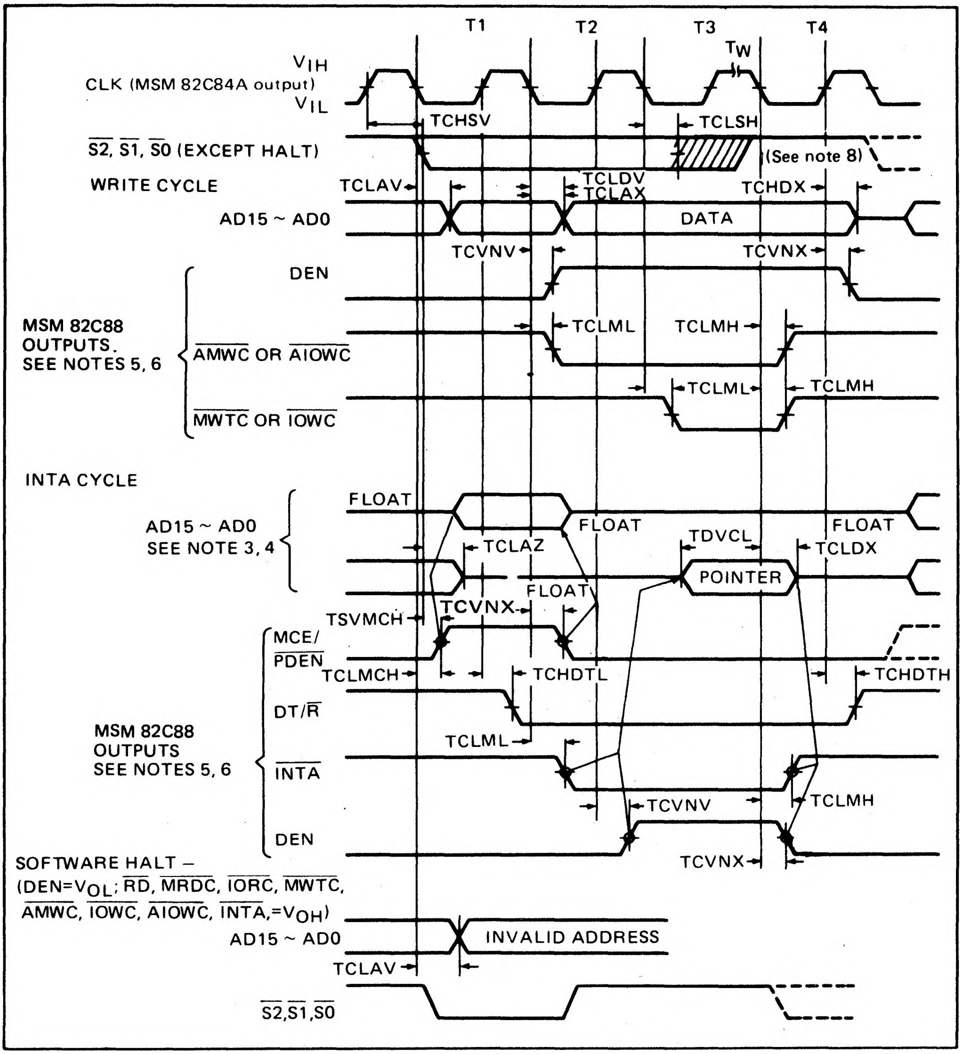
**Minimum Mode (Continued)**



**Maximum Mode**



**Maximum Mode (Continued)**

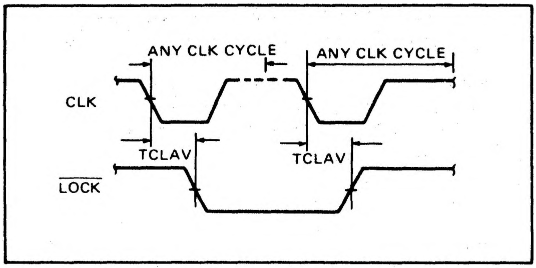



**Notes:**
1. All signals switch between $V_{OH}$ and $V_{OL}$ unless otherwise specified.
2. RDY is sampled near the end of T2,T3,$T_W$ to determine if $T_W$ machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycle.
4. Two INTA cycles run back-to-back. The MSM 80C86A LOCAL ADDR/DATA BUS is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
5. Signals at MSM 82C84A or MSM 82C88 are shown for reference only.
6. The issuance of the MSM 82C88 command and control signals ($\overline{MRDC}$,$\overline{MWTC}$,$\overline{AMWC}$,$\overline{IORC}$,$\overline{IOWC}$,$\overline{AIOWC}$, $\overline{INTA}$ and DEN) lags the active high MSM 82C88 CEN.
7. All timing measurements are made at 1.5V unless otherwise noted.
8. Status inactive in state just prior to T4.

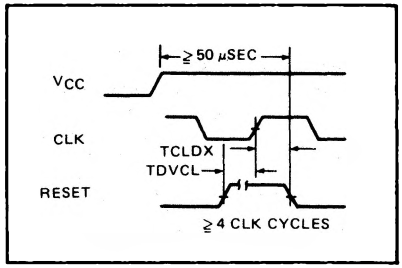
**Asynchronous Signal Recognition**


**NOTE:** 1. SETUP REQUIREMENTS FOR ASYNCHRONOUS SIGNALS ONLY TO GUARANTEE RECOGNITION AT NEXT CLK

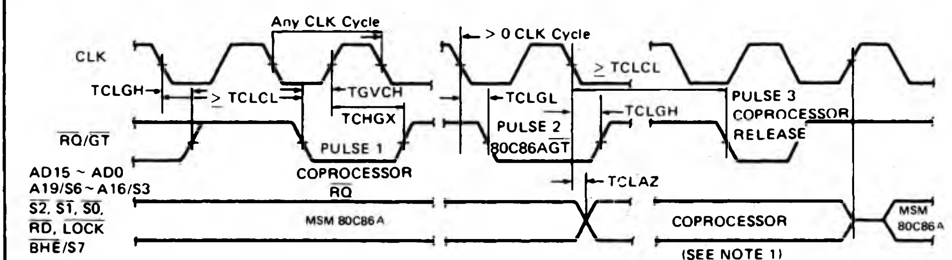
**Bus Lock Signal Timing (Maximum Mode Only)**




**Reset Timing**

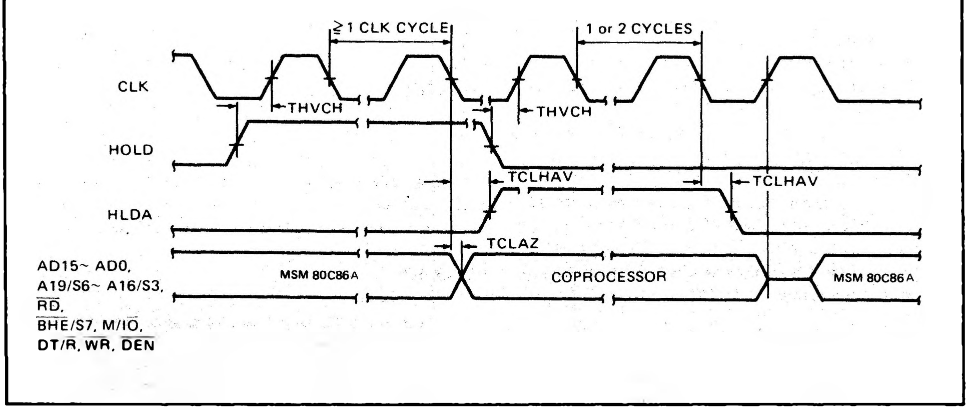



**Request/Grant Sequence Timing (Maximum Mode Only)**




**Note :** 1. The coprocessor may not drive the buses outside the region shown without risking contention.

**Hold/Hold Acknowledge Timing (Minimum Mode Only)**



## PIN DESCRIPTION

**AD0 – AD15**
ADDRESS DATA BUS: Input/Output

These lines are the multiplexed address and data bus.

These are the address bus at the T1 cycle and the data bus at the T2, T3, TW and T4 cycles.

At the T1 cycle, AD0 low indicates Data Bus Low (D0 – D7) Enable. These lines are high impedance during interrupt acknowledge and hold acknowledge.

**A16/S3. A17/S4, A18/S5, A19/S6**
ADDRESS/STATUS: Output

Tses are the four most significant addresses, at the T1 cycle. Accessing I/O port address, these are low at T1 cycles. These lines are Status lines at T2, T3, TW and T4 cycles. S3 and S4 are encoded as shown.

| S3 | S4 | Characteristics |
|---|---|---|
| 0 | 0 | Alternate Data |
| 1 | 0 | Stack |
| 0 | 1 | Code or None |
| 1 | 1 | Data |

These lines are high impedance during hold acknowledge.

**$\overline{BHE}$/S7**
BUS HIGH ENABLE/STATUS: Output

This line indicates Data Bus High Enable (BHE) at the T1 cycle.

This line indicates Data Bus High Enable (BHE) at the T1 cycles.

This line is status line at T2, T3, TW and T4 cycles.

**$\overline{RD}$**
READ: Output

This line indicates that CPU is in the memory or I/O read cycle.

This line is the read strobe signal when CPU read data from memory or I/O device.

This line is active low.

This line is high impedance during hold acknowledge.

**READY**
READY: Input

This line indicates to the CPU that the addressed memory or I/O device is ready to read or write.

This line is active high.

If the setup and hold time is out of specification, illegal operation will occur.

**INTR**
INTERRUPT REQUEST: Input

This line is the level triggered interrupt request signal which is sampled during the last clock cycle of instruction and string manipulation.

It can be internally masked by software.

This signal is active high and internally synchronized.

**$\overline{TEST}$**
TEST: Input

This line is examined by the WAIT instruction.

When $\overline{TEST}$ is high, the CPU enters idle cycle.

When $\overline{TEST}$ is low, the CPU exits the idle cycle.

**NMI**
NON MASKABLE INTERRUPT: Input

This line causes a type 2 interrupt.

NMI is not maskable.

This signal is internally synchronized and needs 2 clock cycles of pulse width.

**RESET**
RESET: Input

This signal causes the CPU to initialize immediately.

This signal is active high and must be at least four clock cycles.

**CLK**
CLOCK: Input

This signal provides the basic timing for the internal circuit.

**MN/$\overline{MX}$**
MINIMUM/MAXIMUM: Input

This signal selects the CPU's operating mode.

When $V_{CC}$ is connected, the CPU operates in Minimum mode.

Whin GND is connected, the CPU orerates Maximum mode.

**$V_{CC}$**
$V_{CC}$: +5V supplied.

**GND**
GROUND

The following pin function descriptions are maximum mode only.

Other pin functions are already described.

**$\overline{S0}$, $\overline{S1}$, $\overline{S2}$**
STATUS: Output

These lines indicate bus status and they are used by the MSM82C88 Bus Controller to generate all memory and I/O access control signals.

These lines are high impedance during hold acknowledge.

These status lines are encoded as shown.

| $\overline{S2}$ | $\overline{S1}$ | $\overline{S0}$ | Characteristics |
|---|---|---|---|
| 0 (LOW) | 0 | 0 | Interrupt acknowledge |
| 0 | 0 | 1 | Read I/O Port |
| 0 | 1 | 0 | Write I/O Port |
| 0 | 1 | 1 | Halt |
| 1 (HIGH) | 0 | 0 | Code Access |
| 1 | 0 | 1 | Read Memory |
| 1 | 1 | 0 | Write Memory |
| 1 | 1 | 1 | Passive |

**$\overline{RQ}/\overline{GT0}$**
**$\overline{RQ}/\overline{GT1}$**
REQUEST/GRANT: Input/Output

These lines are used for Bus Request from other devices and Bus GRANT to other deivces.

These lines are bidirectional and active low.

**$\overline{LOCK}$**
LOCK: Output

This line is active low.

When this line is low, other devices can not gain control of the bus.

This line is high impedance during hold acknowledge.

**QS0/QS1**
QUEUE STATUS: Output

These lines are Queue Status, and indicate internal instrucion queue status.

| QS1 | QS0 | Characteristics |
|---|---|---|
| 0 (LOW) | 0 | No Operation |
| 0 | 1 | First Byte of Op Code from Queue |
| 1 (HIGH) | 0 | Empty the Queue |
| 1 | 1 | Subsequent Byte from Queue |

The following pin function descriptions are minimum mode only. Other pin functions are already described.

**M/$\overline{IO}$**
STATUS: Output

This line selects memory address space or I/O address space.

When this line is high, the CPU selects memory address space and when it is low, the CPU selects I/O address space.

This line is high impedance during hold acknowledge.

**$\overline{WR}$**
WRITE: Output

This line indicates that the CPU is in the memory or I/O write cycle.

This line is a write strobe signal when the CPU writes data to memory or I/O device.

This line is active low.

This line is high impedance during hold acknowledge.

**$\overline{INTA}$**
INTERRUPT ACKNOWLEDGE: Output

This line is a read strobe signal for the interrupt acknowledge cycle.

This line is active low.

**ALE**
ADDRESS LATCH ENABLE: Output

This line is used for latching the address into the MSM82C12 address latch, It is a possitive pulse and its trailing edge is used to strobe the address, This line is never floated.

**DT/$\overline{R}$**
DATA TRANSMIT/RECEIVE: Output

This line is used to control the output enable of the bus transceiver.

When this line is high, the CPU transmits data, and when it is low, the CPU receives data.

This line is high impedance during hold acknowledge.

**$\overline{DEN}$**
DATA ENABLE: Output

This line is used to control the output enable of the bus transceiver.

This line is active low. This line is high impedance during hold acknowledge.

**HOLD**
HOLD REQUEST: Input

This line is used for Bus Request from other devices.
This line is active high.

**HLDA**
HOLD ACKNOWLEDGE: Output

This line is used for Bus Grant to other devices.

This line is active high.

# FUNCTIONAL DESCRIPTION

## STATIC OPERATION

All MSM80C86A circuitry is of static design. Internal registers, counters and latches are static and require no refresh as with dynamic circuit design. This eliminates the minimum operating frequency restriction placed on other microprocessors. The MSM80C86A can operate from DC to the appropriate upper frequency limit. The processor clock may be stopped in either state (high/low) and held there indefinitely. This type of operation is especially useful for system debug or power critical applications.

The MSM80C86A can be single stepped using only the CPU clock. This state can be maintained as long as is necessary. Sigle step clock operation allows simple interface circuitry to provice critical information for bringing up your system.

Static design also allows very low frequency operation (down to DC). In a power critical situation, this can provide extremely low power operation since MSM80C86A power dissipation is directly related to operating frequency. As the system frequency is reduced, so is the operating power until, ultimately, at a DC input frequency, MSM80C86A power requirement is the standby current (500 μA maximum).

### GENERAL OPERATION

The internal function of the MSM80C86 consists of a Bus Interface Unit (BIU) and an Execution Unit (EU). These units operate mutually but perform as separate processors.

BIU performs instruction fetch and queueing, operand fetch, DATA read and write address relocation and basic bus control. Instruction pre-fetch is perfomed while waiting for decording and execution of instructions. Thus, the CPU's performance is increased. Up to 6-bytes of instruction stream can be queued.

The EU receives pre-fetched instructions from the BIU queue, decodes and executes the instructions, and provides the un-relocated operand address to BIU.

### MEMORY ORGANIZATION

The MSM80C86A has a 20-bit address to memory. Each address has an 8-bit data width. Memory is organized 00000H to FFFFFH and is logicaly divided into four segments: code, data, extra data and stack segment. Each segment contains up to 64 Kbytes and locates on a 16-byte boundary. (Fig. 3a)

All memory references are made relative to the segment register which functions in accordance with a select rule. Word operands can be located on even or odd address boundary.

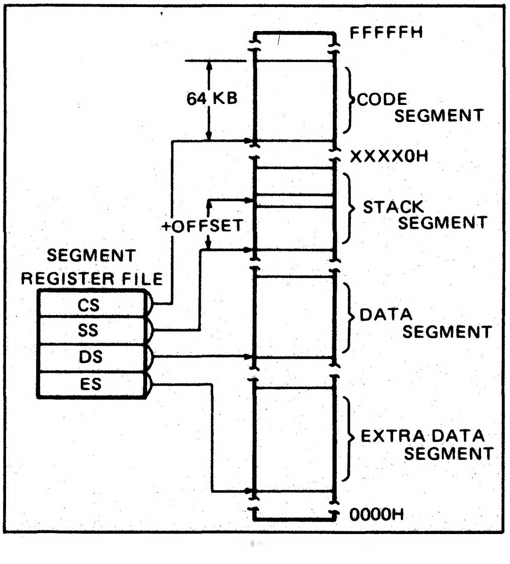
The BIU automatically performs the proper number of memory accesses. Memory consists of an even address and an odd address. Byte data of even address is transfered on the D0 — D7 and byte data of odd address is transfered on the D8 — D15.

The CPU prevides two enable signals $\overline{BHE}$ and A0 to access either an odd address, even address or both:

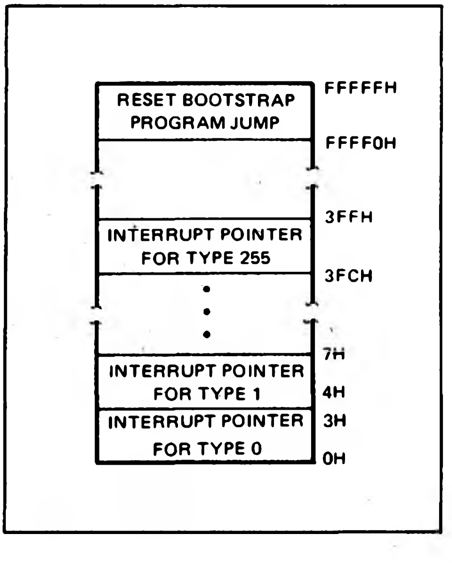
Memory location FFFF0H is the start address after reset, and 00000H through 003FFH are reserved as an interrupt pointer, where there are 256 types of interrupt pointers.

Each interrupt type has a 4-byte pointer element consisting of a 16-bit segment address and a 16-bit offset address.

**Memory Organization**




**Reserved Memory Locations**



| Memory Reference Need | Segment Register Used | Segment Selection Rule |
|---|---|---|
| Instructions | CODE (CS) | Automatic with all instruction prefetch. |
| Stack | STACK (SS) | All stack pushes and pops. Memory references relative to BP base register except data references. |
| Local Data | DATA (DS) | Data references when relative to stack destination of string operation, or explicitly overridden. |
| External (Global) Data | EXTRA (ES) | Destination of string operations: Explicitly selected using a segment overriden. |

**MINIMUM AND MAXIMUM MODES**

The MSM80C86A has two system modes: minimum and maximum. When using maximum mode, it is easy to organize a multi-CPU system with a 82C88 Bus Controller which generate the bus control signal.

When using minimum mode, it is easy to organize a simple system by generating bus control signal by itself.

MN/$\overline{MX}$ is the mode select pin. Definition of 24-31 pin changes depend on the MN/$\overline{MX}$ pin.

**BUS OPERATION**

The MSM80C86A has a time multiplexed address and data bus. If a non-multiplexed bus is desired for a system, it is only to add the address latch.

A CPU bus cycle consists of at least four clock cycles: T1, T2 T3 and T4. (Fig. 4)

The address output occurs during T1 and data transfer occurs during T3 and T4. T2 is used for changing the direction of the bus at the read operation. When the device which is accessed by the CPU is not ready for The data transfer and the CPU "NOT READY", TW cycles are inserted between T3 and T4.

When a bus cycle is not needed, T1 cycles are inserted between the bus cycles for internal execution. During the T1 cycle, the ALE signal is output from the CPU or the MSM82C88 depending on MN/$\overline{MX}$. At the trailing edge of ALE, a valid address may be latched.

Status bits $\overline{S0}$, $\overline{S1}$ and $\overline{S2}$ are used in the maximum mode by the bus controller to recognize the type of bus operation according to the following table.

| $\overline{S2}$ | $\overline{S1}$ | $\overline{S0}$ | Characteristics |
|---|---|---|---|
| 0 (LOW) | 0 | 0 | Interrupt acknowledge |
| 0 | 0 | 1 | Read I/O |
| 0 | 1 | 0 | Write I/O |
| 0 | 1 | 1 | Halt |
| 1 (HIGH) | 0 | 0 | Instruction Fetch |
| 1 | 0 | 1 | Read Data from Memory |
| 1 | 1 | 0 | Write Data to Memory |
| 1 | 1 | 1 | Passive (no bus cycle) |

Status bits S3 through S7 are multiplexed with A16 ~ A19, and $\overline{BHE}$: therefore, they are valid during T2 through T4.

S3 and S4 indicate which segment register was selected on the bus cycle, according to the following table.

| S4 | S3 | Characteristics |
|---|---|---|
| 0 (LOW) | 0 | Alternate Data (Extra segment) |
| 0 | 1 | Stack |
| 1 (HIGH) | 0 | Code or None |
| 1 | 1 | Data |

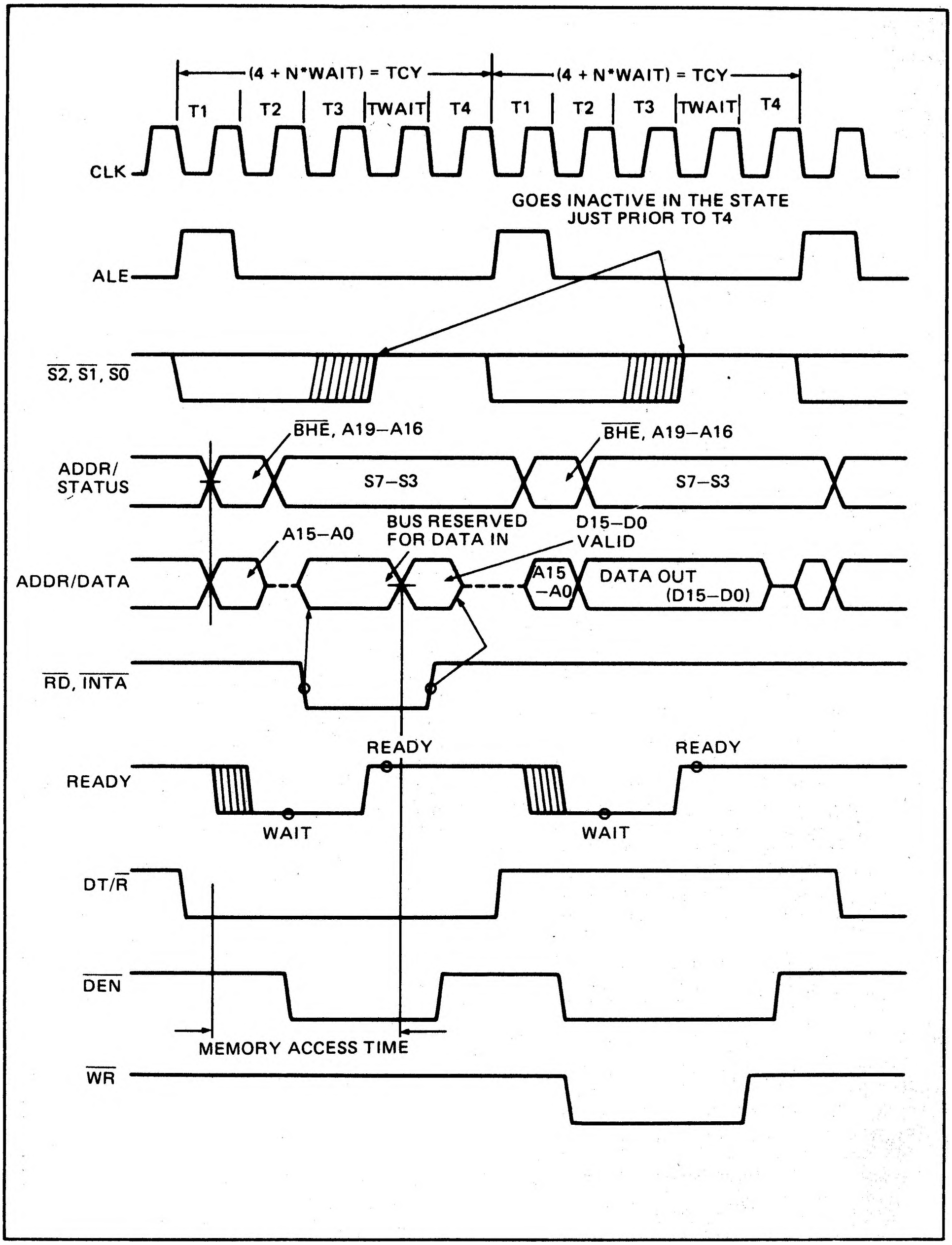
S5 indicates interrupt enable Flag.

**I/O ADDRESSING**

The MSM80C86A has 64 Kbyte of I/O or as 32 Kwords I/O. When the CPU accesses an I/O device, addresses A0 ~ A15 are in the same format as a memory address, and A16 ~ A19 are low.

The I/O ports addresses are same as memory, so it is necessary to be careful when using 8-bit peripherals.

**Basic System Timing**



# EXTERNAL INTERFACE

**RESET**

CPU Initalization is executed by the RESET pin. The MSM80C86A's RESET High signal is required for greater than 4 clock cycles.

The Rising edge of RESET terminates present operation immediately. The Falling edge of RESET triggers an internal reset sequence for approximately 10 clock cycles. Afer the internal reset sequence is finished normal operation occurs from absolute location FFFFOH.

**INTERRUPT OPERATIONS**

Interrupt operation is classified as software or hardware, and hardware interrupt is classified as non-maskable or maskable.

An interrupt causes a new program location defined on the interrupt pointer table, according to the interrupt type. Absolute locations 00000H through 003FFH are reserved for the interrupt pointer table. The interrupt pointer table consists of 256-elements. Each element is 4 bytes in size and corresponds to an 8 bit type number which is sent from an interrupt interrupt request device during the interrupt acknowledge cycle.

**NON-MASKABLE INTERRUPT (NMI)**

The MSM80C86A has a Non-maskable Interrupt (NMI) which is of higher priority than the maskable interrupt request (INTR).

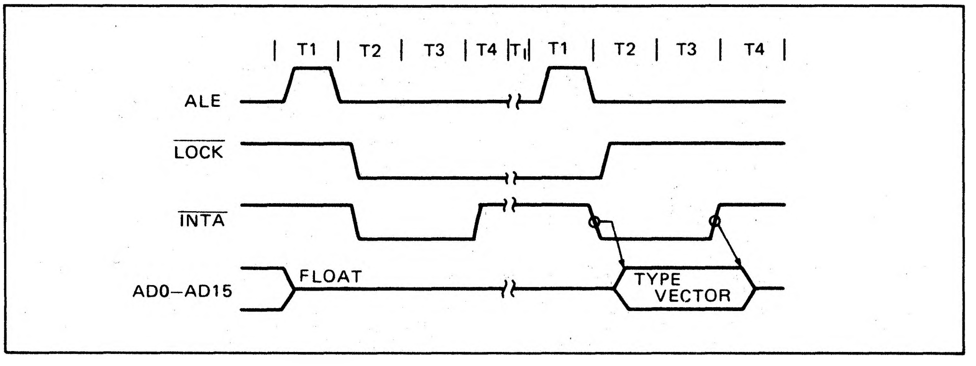
The NMI request pulse width needs a minimum of 2 clock cycles. The NMI will be serviced at the end of the current instruction or between string manipulations.

**MASKABLE INTERRUPT (INTR)**

The MSM80C86A provides another interrupt request (INTR) which can be masked by software. INTR is level triggered, so it must be held until the interrupt request is acknowledged.

INTR will be serviced at the end of the aurrent instruction or between string manipulations.

**Interrupt Acknowledge Sequence**




**INTERRUPT ACKNOWLEDGE**

During the interrupt acknowledge sequence, further interrupts are disabled. The interrupt enable bit is reset by any interrupt, after which the Flag register is automatically pushed onto the stack. During the acknowledge sequence, the CPU emits the lock signal from T2 of the first bus cycle to T2 of the second bus cycle. At second bus cyclees, byte is fetched from the external device as a vector which identified the type of interrupt. This vector is multiplied by four and used as a interrupt pointer address. (INTR only)

The Interrupt Return (IRET) instruction includes a Flag pop operation which returns the original interrupt enable bit when it restores the Flag.

**HALT**

When a Halt instruction is executed, the CPU enters the Halt state. An interrupt request or RESET will force the MSM80C86A out of the Halt state.

**SYSTEM TIMING – MINIMUM MODE**

A bus cycle begins T1 with an ALE signal. The trailing edge of ALE is used to latch the address. From T1 to T4 the M/$\overline{IO}$ signal indicates a memory or I/O operation. From T2 to T4, the address data bus changes the address bus to data bus.

The read ($\overline{RD}$), write ($\overline{WR}$) and interrupt acknowledge ($\overline{INTA}$) signals causes the addressed device to enable data bus. These signal becomes active at the beginning of T2 and inactive at the beginning of T4.

**SYSTEM TIMING – MAXIMUM MODE**

At maximum mode, the MSM82C88 Bus Controller is added to system. The CPU sends status information to the Bus Controller. Bus timing signals are generated by Bus Controller. Bus timing is almost the same as in the minimum mode.

## BUS HOLD CIRCUITRY

To avoid high current conditions caused by floating inputs to CMOS devices and to eliminate the need for pull-up/down resistors, "bus-hold" circuitry has been used on MSM80C86A pins 2–16, 26–32, and 34–39 (Figures 6a, 6b). These circuits will maintain the last valid logic state if no driving source is present (i.e. an unconnected pin or a driving source which goes to a high impedance state). To overdrive the "bus hold" circuits, an external driver must be capable of supplying approximately 600 μA minimum sink or source current at valid input voltage levels. Since this "bus hold" circuitry is active and not a "resistive" type element, the associated power supply current is negligible and power dissipation is significantly reduced when compared to the use of passive pull-up resistors.

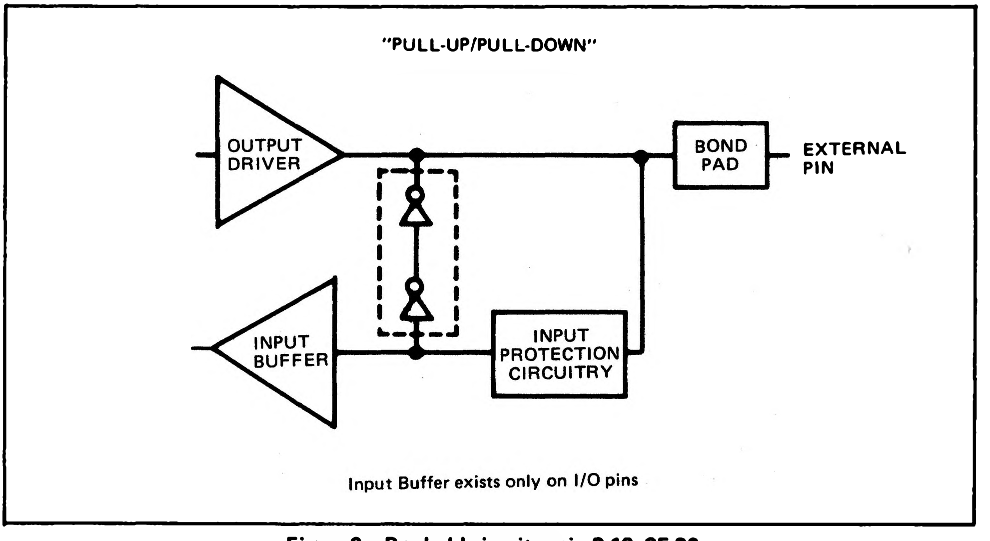



**Figure 6a. Bus hold circuitry pin 2-16, 35-39.**

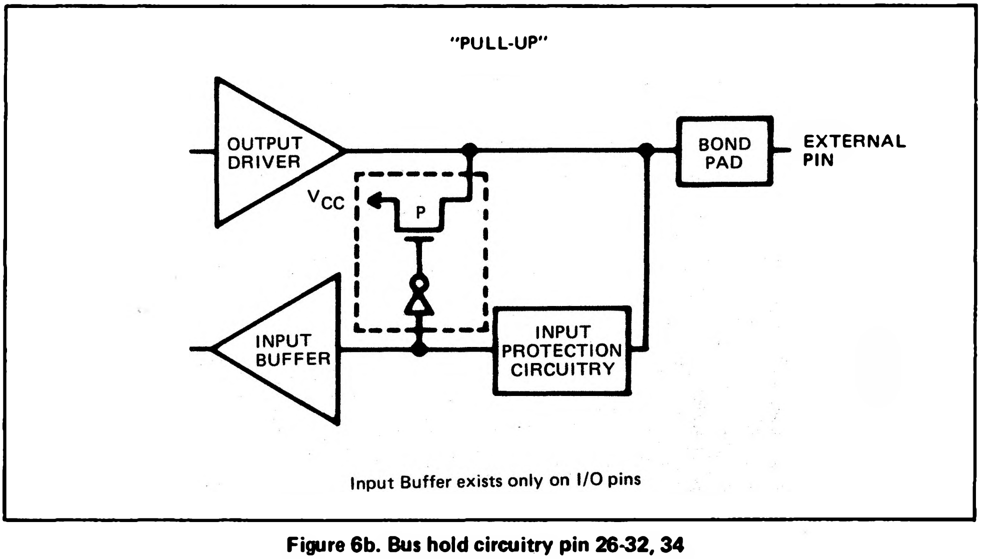



**Figure 6b. Bus hold circuitry pin 26-32, 34**

## DATA TRANSFER

| | | | | |
|---|---|---|---|---|
| MOV = Move: | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 |
| Register/memory to/from register | 1 0 0 0 1 0 d w | mod reg r/m | | |
| Immediate to register/memory | 1 1 0 0 0 1 1 w | mod 0 0 0 r/m | data | data if w = 1 |
| Immediate to register | 1 0 1 1 w reg | data | data if w = 1 | |
| Memory to accumulator | 1 0 1 0 0 0 0 w | addr-low | addr-high | |
| Accumulator to memory | 1 0 1 0 0 0 1 w | addr-low | addr-high | |
| Register/memory to segment register | 1 0 0 0 1 1 1 0 | mod 0 reg r/m | | |
| Segment register to register/memory | 1 0 0 0 1 1 0 0 | mod 0 reg r/m | | |
| PUSH = Push: | | | | |
| Register/memory | 1 1 1 1 1 1 1 1 | mod 1 1 0 r/m | | |
| Register | 0 1 0 1 0 reg | | | |
| Segment register | 0 0 0 reg 1 1 0 | | | |
| POP = Pop: | | | | |
| Register/memory | 1 0 0 0 1 1 1 1 | mod 0 0 0 r/m | | |
| Register | 0 1 0 1 1 reg | | | |
| Segment register | 0 0 0 reg 1 1 1 | | | |
| XCHG = Exchange: | | | | |
| Register/memory with register | 1 0 0 0 0 1 1 w | mod reg r/m | | |
| Register with accumulator | 1 0 0 1 0 reg | | | |
| IN = Input from: | | | | |
| Fixed port | 1 1 1 0 0 1 0 w | port | | |
| Variable port | 1 1 1 0 1 1 0 w | | | |
| OUT = Output to: | | | | |
| Fixed port | 1 1 1 0 0 1 1 w | port | | |
| Variable port | 1 1 1 0 1 1 1 w | | | |
| XLAT = Translate byte to AL | 1 1 0 1 0 1 1 1 | | | |
| LEA = Load EA to register | 1 0 0 0 1 1 0 1 | mod reg r/m | | |
| LDS = Load pointer to DS | 1 1 0 0 0 1 0 1 | mod reg r/m | | |
| LES = Load pointer to ES | 1 1 0 0 0 1 0 0 | mod reg r/m | | |
| LAHF = Load AH with flags | 1 0 0 1 1 1 1 1 | | | |
| SAHF = Store AH into flags | 1 0 0 1 1 1 1 0 | | | |
| PUSHF = Push flags | 1 0 0 1 1 1 0 0 | | | |
| POPF = Pop flags | 1 0 0 1 1 1 0 1 | | | |

## ARITHMETIC

| | | | | |
|---|---|---|---|---|
| ADD = Add: | | | | |
| Reg./memory with register to either | 0 0 0 0 0 0 d w | mod reg r/m | | |
| Immediate to register/memory | 1 0 0 0 0 0 s w | mod 0 0 0 r/m | data | data if s:w = 01 |
| Immediate to accumulator | 0 0 0 0 0 1 0 w | data | data if w = 1 | |
| ADC = Add with carry: | | | | |
| Reg./memory with register to either | 0 0 0 1 0 0 d w | mod reg r/m | | |
| Immediate to register/memory | 1 0 0 0 0 0 s w | mod 0 1 0 r/m | data | data if s:w = 01 |
| Immediate to accumulator | 0 0 0 1 0 1 0 w | data | data if w = 1 | |
| INC = Increment: | | | | |
| Register/memory | 1 1 1 1 1 1 1 w | mod 0 0 0 r/m | | |
| Register | 0 1 0 0 0 reg | | | |
| AAA = ASCII adjust for add | 0 0 1 1 0 1 1 1 | | | |
| DAA = Decimal adjust for add | 0 0 1 0 0 1 1 1 | | | |
| SUB = subtract: | | | | |
| Reg./memory and register to either | 0 0 1 0 1 0 d w | mod reg r/m | | |
| Immediate from register/memory | 1 0 0 0 0 0 s w | mod 1 0 1 r/m | data | data if s:w = 01 |
| Immediate from accumulator | 0 0 1 0 1 1 0 w | data | data if w = 1 | |
| SBB = Subtract with borrow: | | | | |
| Reg./memory and register to either | 0 0 0 1 1 0 d w | mod reg r/m | | |
| Immediate from register/memory | 1 0 0 0 0 0 s w | mod 0 1 1 r/m | data | data if s:w = 01 |
| Immediate from accumulator | 0 0 0 1 1 1 0 w | data | data if w = 1 | |
| DEC = Decrement: | | | | |
| Register/memory | 1 1 1 1 1 1 1 w | mod 0 0 1 r/m | | |
| Register | 0 1 0 0 1 reg | | | |
| NEG = Change sign | 1 1 1 1 0 1 1 w | mod 0 1 1 r/m | | |
| CMP = Compare: | | | | |
| Register/memory and register | 0 0 1 1 1 0 d w | mod reg r/m | | |
| Immediate with register/memory | 1 0 0 0 0 0 s w | mod 1 1 1 r/m | data | data if s:w = 01 |
| Immediate with accumulator | 0 0 1 1 1 1 0 w | data | data if w = 1 | |
| AAS = ASCII adjust for subtract | 0 0 1 1 1 1 1 1 | | | |

| | | | | |
|---|---|---|---|---|
| DAS = Decimal adjust for subtract | 0 0 1 0 1 1 1 1 | | | |
| MUL = Multiply (unsigned) | 1 1 1 1 0 1 1 w | mod 1 0 0 r/m | | |
| IMUL = Integer multiply (signed) | 1 1 1 1 0 1 1 w | mod 1 0 1 r/m | | |
| AAM = ASCII adjust for multiply | 1 1 0 1 0 1 0 0 | 0 0 0 0 1 0 1 0 | | |
| DIV = Divide (unsigned) | 1 1 1 1 0 1 1 w | mod 1 1 0 r/m | | |
| IDIV = Integer divide (signed) | 1 1 1 1 0 1 1 w | mod 1 1 1 r/m | | |
| AAD = ASCII adjust for divide | 1 1 0 1 0 1 0 1 | 0 0 0 0 1 0 1 0 | | |
| CBW = Convert byte to word | 1 0 0 1 1 0 0 0 | | | |
| CWD = Convert word to double word | 1 0 0 1 1 0 0 1 | | | |

## LOGIC

| | | | | |
|---|---|---|---|---|
| NOT = Invert | 1 1 1 1 0 1 1 w | mod 0 1 0 r/m | | |
| SHL/SAL = Shift logical/arithmetic left | 1 1 0 1 0 0 v w | mod 1 0 0 r/m | | |
| SHR = Shift logical right | 1 1 0 1 0 0 v w | mod 1 0 1 r/m | | |
| SAR = Shift arithmetic right | 1 1 0 1 0 0 v w | mod 1 1 1 r/m | | |
| ROL = Rotate left | 1 1 0 1 0 0 v w | mod 0 0 0 r/m | | |
| ROR = Rotate right | 1 1 0 1 0 0 v w | mod 0 0 1 r/m | | |
| RCL = Rotate left through carry | 1 1 0 1 0 0 v w | mod 0 1 0 r/m | | |
| RCR = Rotate right through carry | 1 1 0 1 0 0 v w | mod 0 1 1 r/m | | |
| AND = And: | | | | |
| Reg./memory and register to either | 0 0 1 0 0 0 d w | mod reg r/m | | |
| Immediate to register/memory | 1 0 0 0 0 0 0 w | mod 1 0 0 r/m | data | data if w = 1 |
| Immediate to accumulator | 0 0 1 0 0 1 0 w | data | data if w = 1 | |
| TEST = And function to flags, no result: | | | | |
| Register/memory and register | 1 0 0 0 0 1 0 w | mod reg r/m | | |
| Immediate data and register/memory | 1 1 1 1 0 1 1 w | mod 0 0 0 r/m | data | data if w = 1 |
| Immediate data and accumulator | 1 0 1 0 1 0 0 w | data | data if w = 1 | |
| OR = Or: | | | | |
| Reg./memory and register to either | 0 0 0 0 1 0 d w | mod reg r/m | | |
| Immediate to register/memory | 1 0 0 0 0 0 0 w | mod 0 0 1 r/m | data | data if w = 1 |
| Immediate to accumulator | 0 0 0 0 1 1 0 w | data | data if w = 1 | |
| XOR = Exclusive or: | | | | |
| Reg./memory and register to either | 0 0 1 1 0 0 d w | mod reg r/m | | |
| Immediate to register/memory | 1 0 0 0 0 0 0 w | mod 1 1 0 r/m | data | data if w = 1 |
| Immediate to accumulator | 0 0 1 1 0 1 0 w | data | data if w = 1 | |

## STRING MANIPULATION

| | | | | |
|---|---|---|---|---|
| REP = Repeat | 1 1 1 1 0 0 1 z | | | |
| MOVS = Move byte/word | 1 0 1 0 0 1 0 w | | | |
| CMPS = Compare byte/word | 1 0 1 0 0 1 1 w | | | |
| SCAS = Scan byte/word | 1 0 1 0 1 1 1 w | | | |
| LODS = Load byte/word to AL/AX | 1 0 1 0 1 1 0 w | | | |
| STOS = Store byte/word from AL/AX | 1 0 1 0 1 0 1 w | | | |

| | | | | |
|---|---|---|---|---|
| **CJMP = Conditional JMP** | | | | |
| JE/JZ = Jump on equal/zero | 0 1 1 1 0 1 0 0 | disp | | |
| JZ/JNGE = Jump on less/not greater or equal | 0 1 1 1 1 1 0 0 | disp | | |
| JLE/JNG = Jump on less or equal/not greater | 0 1 1 1 1 1 1 0 | disp | | |
| JB/JNAE = Jump on below/not above or equal | 0 1 1 1 0 0 1 0 | disp | | |
| JBE/JNA = Jump on below or equal/not above | 0 1 1 1 0 1 1 0 | disp | | |
| JP/JPE = Jump on parity/parity even | 0 1 1 1 1 0 1 0 | disp | | |
| JO = Jump on over flow | 0 1 1 1 0 0 0 0 | disp | | |
| JS = Jump on sign | 0 1 1 1 1 0 0 0 | disp | | |
| JNE/JNZ = Jump on not equal/not zero | 0 1 1 1 0 1 0 1 | disp | | |
| JNL/JGE = Jump on not less/greater or equal | 0 1 1 1 1 1 0 1 | disp | | |
| JNLE/JG = Jump on not less or equal/greater | 0 1 1 1 1 1 1 1 | disp | | |
| JNB/JAE = Jump on not below/above or equal | 0 1 1 1 0 0 1 1 | disp | | |
| JNBE/JA = Jump on not below or equal/above | 0 1 1 1 0 1 1 1 | disp | | |
| JNP/JPO = Jump on not parity/parity odd | 0 1 1 1 1 0 1 1 | disp | | |
| JNO = Jump on not overflow | 0 1 1 1 0 0 0 1 | disp | | |
| JNS = Jump on not sign | 0 1 1 1 1 0 0 1 | disp | | |
| LOOP = Loop CX times | 1 1 1 0 0 0 1 0 | disp | | |
| LOOPZ/LOOPE = Loop while zero/equal | 1 1 1 0 0 0 0 1 | disp | | |
| LOOPNZ/LOOPNE = Loop while not zero/equal | 1 1 1 0 0 0 0 0 | disp | | |
| JCXZ = Jump on CX zero | 1 1 1 0 0 0 1 1 | disp | | |
| INT = Interrupt: | | | | |
| Type specified | 1 1 0 0 1 1 0 1 | type | | |
| Type 3 | 1 1 0 0 1 1 0 0 | | | |
| INTO = Interrupt on overflow | 1 1 0 0 1 1 1 0 | | | |
| IRET = Interrupt return | 1 1 0 0 1 1 1 1 | | | |

## PROCESSOR CONTROL

| | | | | |
|---|---|---|---|---|
| CLC = Clear carry | 1 1 1 1 1 0 0 0 | | | |
| CMC = Complement carry | 1 1 1 1 0 1 0 1 | | | |
| STC = Set carry | 1 1 1 1 1 0 0 1 | | | |
| CLD = Clear direction | 1 1 1 1 1 1 0 0 | | | |
| STD = Set direction | 1 1 1 1 1 1 0 1 | | | |
| CLI = Clear interrupt | 1 1 1 1 1 0 1 0 | | | |
| STI = Set interrupt | 1 1 1 1 1 0 1 1 | | | |
| HLT = Halt | 1 1 1 1 0 1 0 0 | | | |
| WAIT = Wait | 1 0 0 1 1 0 1 1 | | | |
| ESC = Escape (to external device) | 1 1 0 1 1 x x x | mod x x x r/m | | |
| LOCK = Bus lock prefix | 1 1 1 1 0 0 0 0 | | | |

## CONTROL TRANSFER

| CALL = Call: | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 |
|---|---|---|---|---|
| Direct within segment | 1 1 1 0 1 0 0 0 | disp-low | disp-high | |
| Indirect within segment | 1 1 1 1 1 1 1 1 | mod 0 1 0 r/m | | |
| Direct intersegment | 1 0 0 1 1 0 1 0 | offset-low | offset-high | |
| | | seg-low | seg-high | |
| Indirect intersegment | 1 1 1 1 1 1 1 1 | mod 0 1 1 r/m | | |
| JMP = Unconditional Jump: | | | | |
| Direct within segment | 1 1 1 0 1 0 0 1 | disp-low | disp-high | |
| Direct within segment-short | 1 1 1 0 1 0 1 1 | disp | | |
| Indirect within segment | 1 1 1 1 1 1 1 1 | mod 1 0 0 r/m | | |
| Direct intersegment | 1 1 1 0 1 0 1 0 | offset-low | offset-high | |
| | | seg-low | seg-high | |
| Indirect intersegment | 1 1 1 1 1 1 1 1 | mod 1 0 1 r/m | | |
| RET = Return from CALL: | | | | |
| Within segment | 1 1 0 0 0 0 1 1 | | | |
| Within seg. adding immediate to SP | 1 1 0 0 0 0 1 0 | data-low | data-high | |
| Intersegment | 1 1 0 0 1 0 1 1 | | | |
| Intersegment adding immediate to SP | 1 1 0 0 1 0 1 0 | data-low | data-high | |

**Footnotes:**

AL = 8-bit accumulator
AX = 18-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value
Greater = more positive
Less = less positive (more negative) signed value
If d = 1 then "to" reg: If d = 0 then "from" reg.
If w = 1 then word instruction: If w = 0 then byte instruction

If mod = 11 then r/m is treated as a REG field
If mod = 00 then DISP = 0*, disp-low and disp-high are absent
If mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
If mod = 10 then DISP = disp-high: disp-low

If r/m = 000 then EA = (BX) + (SI) + DISP
If r/m = 001 then EA = (BX) + (DI) + DISP
If r/m = 010 then EA = (BP) + (SI) + DISP
If r/m = 011 then EA = (BP) + (DI) + DISP
If r/m = 100 then EA = (SI) + DISP
If r/m = 101 then EA = (DI) + DISP
If r/m = 110 then EA = (BP) + DISP*
If r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)
* except if mod = 00 and r/m = 110 then EA-disp-high: disp-low

If s:w = 01 then 16 bits of immediate data form the operand
If s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand
If v = 0 then "count" = 1: If v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparison with ZF FLAG

SEGMENT OVERRIDE PREFIX

001 reg 110

REG is assigned according to the following table:

| 16-Bit | (w = 1) | 8-Bit | (w = 0) | Segment |
|---|---|---|---|---|
| 000 | AX | 000 | AL | 00 ES |
| 001 | CX | 001 | CL | 01 CS |
| 010 | DX | 010 | DL | 10 SS |
| 011 | BX | 011 | BL | 11 DS |
| 100 | SP | 100 | AH | |
| 101 | BP | 101 | CH | |
| 110 | SI | 110 | DH | |
| 111 | DI | 111 | BH | |

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = x:x:x:x:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)