

1. **I8086 Internal Structure. Pin description**
2. **Programming model. Effective Address. Flags**
3. **Memory Space organization. Segment and offset addressing scheme**

The Intel 8086 high performance 16-bit CPU is available in three clock rates: 5, 8 and 10 MHz. The CPU is implemented in N-Channel, depletion load, silicon gate technology (HMOS-III), and packaged in a 40-pin Cerdip or plastic package. The 8086 operates in both single processor and multiple processor configurations to achieve high performance levels.

- **Direct Addressing Capability 1 MByte of Memory**
- **Architecture Designed for Powerful Assembly Language and Efficient High Level Languages**
- **14 Word, by 16-Bit Register Set with Symmetrical Operations**
- **24 Operand Addressing Modes**
- **Bit, Byte, Word, and Block Operations**
- **8 and 16-Bit Signed and Unsigned Arithmetic in Binary or Decimal Including Multiply and Divide**
- **Range of Clock Rates:**
5 MHz for 8086,
8 MHz for 8086-2,
10 MHz for 8086-1
- **MULTIBUS System Compatible Interface**
- **Available in EXPRESS**
— Standard Temperature Range
— Extended Temperature Range
- **Available in 40-Lead Cerdip and Plastic Package**

Architecture of 8086

The following diagram depicts the architecture of a 8086 Microprocessor –

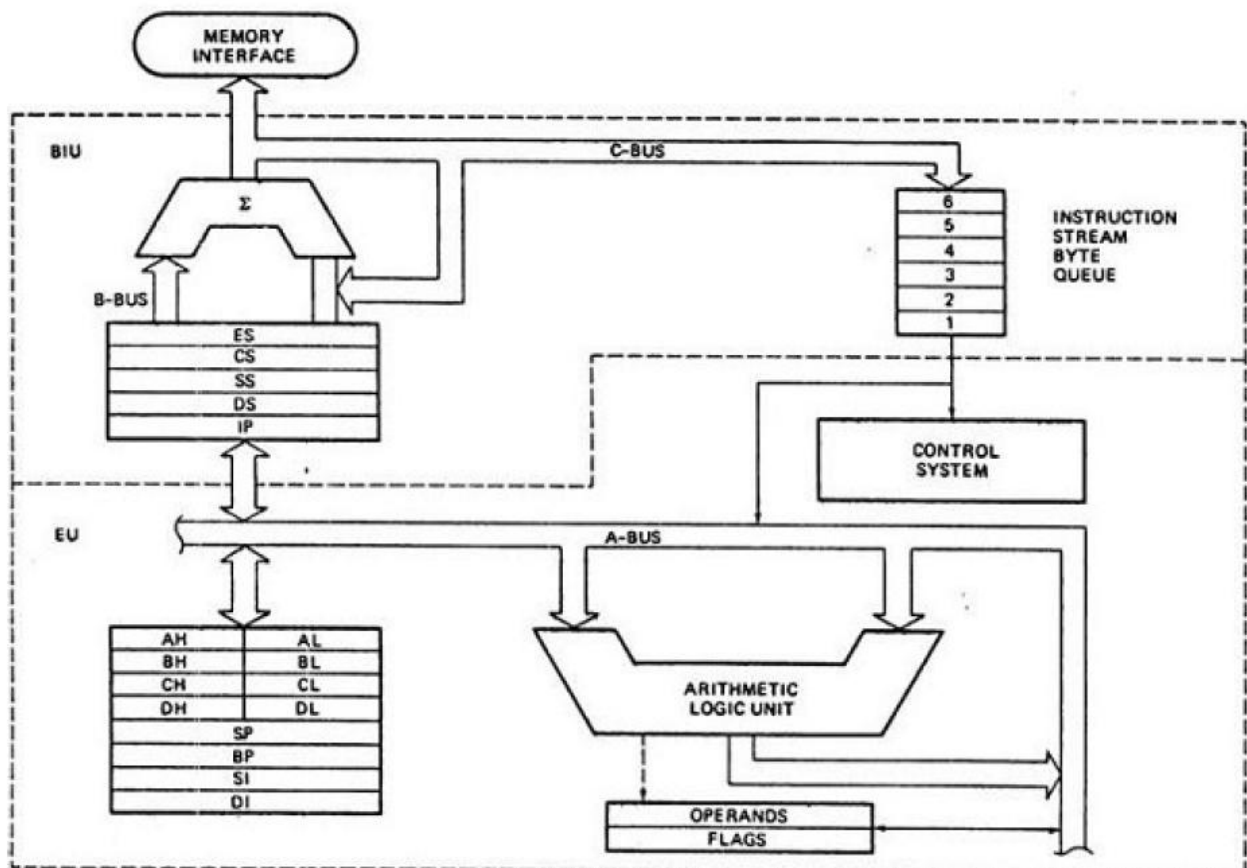


Fig.1 8086 Architecture

8086 Microprocessor is divided into two functional units, i.e., **EU** (Execution Unit) and **BIU** (Bus Interface Unit).

EU (Execution Unit)

Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU. EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU.

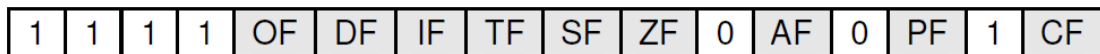
Let us now discuss the functional parts of 8086 microprocessors.

ALU

It handles all arithmetic and logical operations, like +, −, ×, /, OR, AND, NOT operations.

Flag Register

It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups – Conditional Flags and Control Flags.



Flags register

Conditional Flags

It represents the result of the last arithmetic or logical instruction executed. Following is the list of conditional flags –

- Overflow Flag (OF) - set if the result is too large positive number, or is too small negative number to fit into destination operand.
- Direction Flag (DF) - if set then string manipulation instructions will auto-decrement index registers. If cleared then the index registers will be auto-incremented.
- Interrupt-enable Flag (IF) - setting this bit enables maskable interrupts.
- Single-step Flag (TF) - if set then single-step interrupt will occur after the next instruction.
- Sign Flag (SF) - set if the most significant bit of the result is set.
- Zero Flag (ZF) - set if the result is zero.
- Auxiliary carry Flag (AF) - set if there was a carry from or borrow to bits 0-3 in the AL register.
- Parity Flag (PF) - set if parity (the number of "1" bits) in the low-order byte of the result is even.
- Carry Flag (CF) - set if there was a carry from or borrow to the most significant bit during last result calculation.

Control Flags

Control flags controls the operations of the execution unit. Following is the list of control flags –

- **Trap flag** – It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
- **Interrupt flag** – It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.
- **Direction flag** – It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-versa.

General purpose register

There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16bit data. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.

- **AX register** – It is also known as accumulator register. It is used to store operands for arithmetic operations.
- **BX register** – It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
- **CX register** – It is referred to as counter. It is used in loop instruction to store the loop counter.
- **DX register** – This register is used to hold I/O port address for I/O instruction.

BIU (Bus Interface Unit)

BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.

It has the following functional parts –

- **Instruction queue** – BIU contains the instruction queue. BIU gets upto 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed.
- Fetching the next instruction while the current instruction executes is called **pipelining**.
- **Segment register** – BIU has 4 segment buses, i.e. CS, DS, SS& ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations. It also contains 1 pointer register IP, which holds the address of the next instruction to executed by the EU.

Most of the registers contain data/instruction offsets within 64 KB memory segment. There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located the processor uses four segment registers:

Code segment (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions.

Stack segment (SS) is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction.

Data segment (DS) is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions.

Extra segment (ES) is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions.

It is possible to change default segments used by general and index registers by prefixing instructions with a CS, SS, DS or ES prefix.

All general registers of the 8088 microprocessor can be used for arithmetic and logic operations. The general registers are:

Accumulator register consists of 2 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX. AL in this case contains the low-order byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations and string manipulation.

Base register consists of 2 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX. BL in this case contains the low-order byte of the word, and BH contains the high-order byte. BX register usually contains a data pointer used for based, based indexed or register indirect addressing.

Count register consists of 2 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX. When combined, CL register contains the low-order byte of the word, and CH contains the high-order byte. Count register can be used as a counter in string manipulation and shift/rotate instructions.

Data register consists of 2 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX. When combined, DL register contains the low-order byte of the word, and DH contains the high-order byte. Data register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

The following registers are both general and index registers:

Stack Pointer (SP) is a 16-bit register pointing to program stack.

Base Pointer (BP) is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.

Source Index (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions.

Destination Index (DI) is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.

Other registers:

Instruction Pointer (IP) is a 16-bit register.

Effective address. Memory organization and addressing

Program, data and stack memories occupy the same memory space. The total addressable memory size is 1MB. As the most of the processor instructions use 16-bit pointers the processor can effectively address only 64 KB of memory. To access memory outside of 64 KB the CPU uses special segment registers to specify where the code, stack and data 64 KB segments are positioned within 1 MB of memory (see the "Registers" section below).

16-bit pointers and data are stored as:

address: low-order byte

address+1: high-order byte

32-bit addresses are stored in "segment:offset" format as:

address+0: low-order byte of segment

address+1: high-order byte of segment

address+2: low-order byte of offset

address+3: high-order byte of offset

Effective /Physical memory address pointed by segment:offset pair is calculated as:

address = (<segment> * 16) + <offset>

Program memory - program can be located anywhere in memory. Jump and call instructions can be used for short jumps within currently selected 64 KB code segment, as well as for far jumps anywhere within 1 MB of memory. All conditional jump instructions can be used to jump within approximately +127 - -127 bytes from current instruction.

Data memory - the processor can access data in any one out of 4 available segments, which limits the size of accessible memory to 256 KB (if all four segments point to different 64 KB blocks). Accessing data from the Data, Code, Stack or Extra segments can be usually done by prefixing instructions with the DS:, CS:, SS: or ES: (some registers and instructions by default may use the ES or SS segments instead of DS segment).

Word data can be located at odd or even byte boundaries. The processor uses two memory accesses to read 16-bit words.

Stack memory can be placed anywhere in memory.

Reserved locations:

- 0000h - 03FFh are reserved for interrupt vectors. Each interrupt vector is a 32-bit pointer in format segment:offset.
- FFFF0h - FFFFFh - after RESET the processor always starts program execution at the FFFF0h address.

8086 pin Diagram

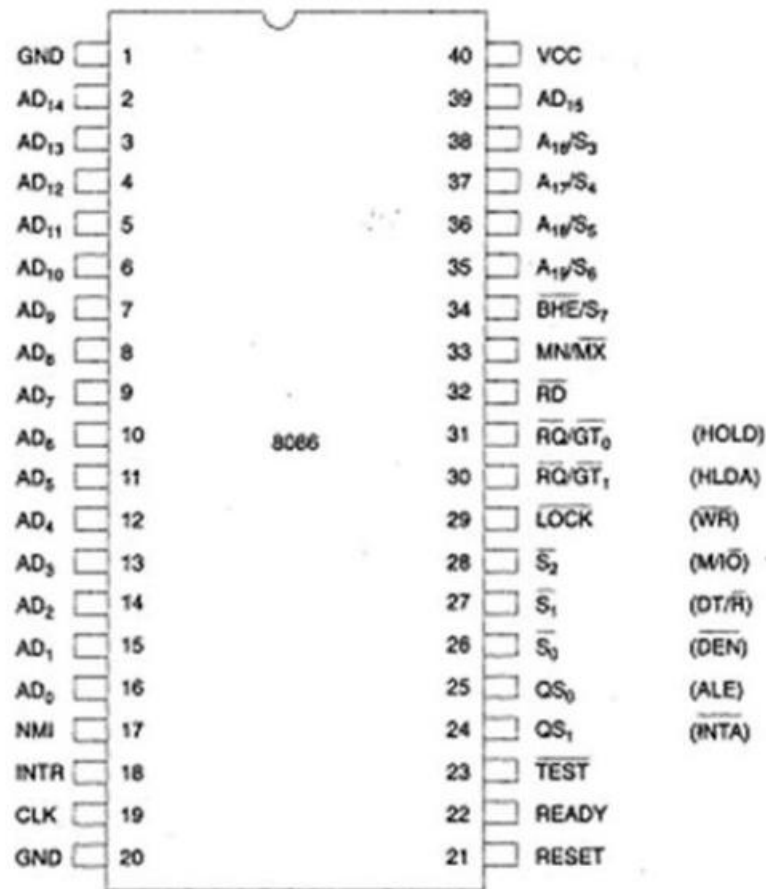


Fig.2 8086 Pinout

Power supply and frequency signals

It uses 5V DC supply at VCC pin 40, and uses ground at VSS pin 1 and 20 for its operation.

Clock signal

Clock signal is provided through Pin-19. It provides timing to the processor for operations. Its frequency is different for different versions, i.e. 5MHz, 8MHz and 10MHz.

Address/data bus

AD0-AD15. These are 16 address/data bus. AD0-AD7 carries low order byte data and AD8-AD15 carries higher order byte data. During the first clock cycle, it carries 16-bit address and after that it carries 16-bit data.

Address/status bus

A16-A19/S3-S6. These are the 4 address/status buses. During the first clock cycle, it carries 4-bit address and later it carries status signals.

S7/BHE

BHE stands for Bus High Enable. It is available at pin 34 and used to indicate the transfer of data using data bus D8-D15. This signal is low during the first clock cycle, thereafter it is active.

Read(RD)

It is available at pin 32 and is used to read signal for Read operation.

Ready

It is available at pin 22. It is an acknowledgement signal from I/O devices that data is transferred. It is an active high signal. When it is high, it indicates that the device is ready to transfer data. When it is low, it indicates wait state.

RESET

It is available at pin 21 and is used to restart the execution. It causes the processor to immediately terminate its present activity. This signal is active high for the first 4 clock cycles to RESET the microprocessor.

INTR

It is available at pin 18. It is an interrupt request signal, which is sampled during the last clock cycle of each instruction to determine if the processor considered this as an interrupt or not.

NMI

It stands for non-maskable interrupt and is available at pin 17. It is an edge triggered input, which causes an interrupt request to the microprocessor.

TEST

This signal is like wait state and is available at pin 23. When this signal is high, then the processor has to wait for IDLE state, else the execution continues.

MN/MX

It stands for Minimum/Maximum and is available at pin 33. It indicates what mode the processor is to operate in; when it is high, it works in the minimum mode and vice-versa.

INTA

It is an interrupt acknowledgement signal and is available at pin 24. When the microprocessor receives this signal, it acknowledges the interrupt.

ALE

It stands for address enable latch and is available at pin 25. A positive pulse is generated each time the processor begins any operation. This signal indicates the availability of a valid address on the address/data lines.

DEN

It stands for Data Enable and is available at pin 26. It is used to enable Transceiver 8286. The transceiver is a device used to separate data from the address/data bus.

DT/R

It stands for Data Transmit/Receive signal and is available at pin 27. It decides the direction of data flow through the transreceiver. When it is high, data is transmitted out and vice-a-versa.

M/IO

This signal is used to distinguish between memory and I/O operations. When it is high, it indicates I/O operation and when it is low indicates the memory operation. It is available at pin 28.

WR

It stands for write signal and is available at pin 29. It is used to write the data into the memory or the output device depending on the status of M/IO signal.

HLDA

It stands for Hold Acknowledgement signal and is available at pin 30. This signal acknowledges the HOLD signal.

HOLD

This signal indicates to the processor that external devices are requesting to access the address/data buses. It is available at pin 31.

QS1 and QS0

These are queue status signals and are available at pin 24 and 25. These signals provide the status of instruction queue. Their conditions are shown in the following table –

QS0	QS1	Status
0	0	No operation
0	1	First byte of opcode from the queue
1	0	Empty the queue
1	1	Subsequent byte from the queue

S0, S1, S2

These are the status signals that provide the status of operation, which is used by the Bus Controller 8288 to generate memory & I/O control signals. These are available at pin 26, 27, and 28. Following is the table showing their status –

S2	S1	S0	Status
0	0	0	Interrupt acknowledgement
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

LOCK

When this signal is active, it indicates to the other processors not to ask the CPU to leave the system bus. It is activated using the LOCK prefix on any instruction and is available at pin 29.

RQ/GT1 and RQ/GT0

These are the Request/Grant signals used by the other processors requesting the CPU to release the system bus. When the signal is received by CPU, then it sends acknowledgment. RQ/GT0 has a higher priority than RQ/GT1.

Memory Space organization. Segment and offset addressing scheme

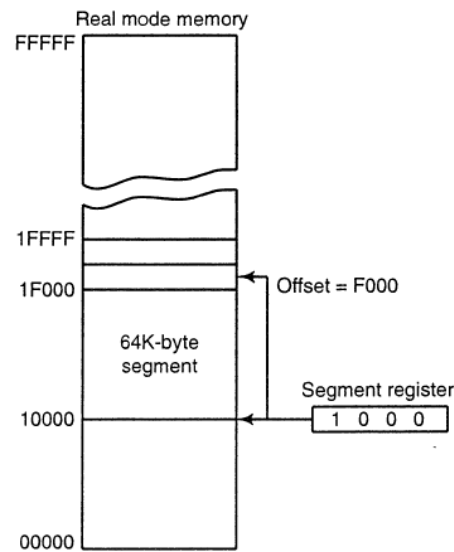


Fig. 2-3 Segment and offset addressing scheme

Segments and Offsets

A combination of a segment address and an offset address access a memory location in the real mode. All real mode memory addresses consist of a segment address plus an offset address. The **segment address**, located within one of the segment registers, defines the beginning address of any 64K-byte memory segment. The **offset address** selects any location within the 64K-byte memory segment. Figure 2-3 shows how the segment plus offset addressing scheme selects a memory location. This illustration shows a memory segment that begins at location 10000H and ends at location 1FFFFH—64K bytes in length. It also shows how an offset, sometimes called a displacement, of F000H selects location 1F000H in the memory system. Note that the offset or displacement is the distance above the start of the segment.

The segment register in Figure 2-3 contains a 1000H, yet it addresses a starting segment at location 10000H. In the real mode, each segment register is internally appended with a 0H on its rightmost end. This forms a 20-bit memory address, allowing it to access the start of a segment at any 16-byte boundary within the first 1M byte of memory. This is required in the microprocessor to generate a 20-bit memory address. For example, if a segment register contains a 1200H, it addresses a 64K-byte memory segment beginning at location 12000H. Likewise, if a segment register contains a 1201H, it addresses a memory segment beginning at location 12010H. Because of the internally appended 0H, real mode segments can only begin at a 16-byte boundary in the memory system. This 16-byte boundary is often called a **paragraph**.

Because a real mode segment of memory is 64K in length, once the beginning address is known, the **ending address** is found by adding FFFFH. For example, if a segment register contains 3000H, the first address of the segment is 30000H and the last address is 30000H + FFFFH or 3FFFFH. Table 2-1 shows several examples of segment register contents and the starting and ending addresses of the memory segments selected by each segment address.

The offset address is added to the start of the segment to address a memory location in the memory segment. For example, if the segment address is 1000H and the offset address is 2000H, the microprocessor addresses memory location 12000H. The segment and offset address is sometimes written as 1000:2000 for a segment address of 1000H with an offset of 2000H.

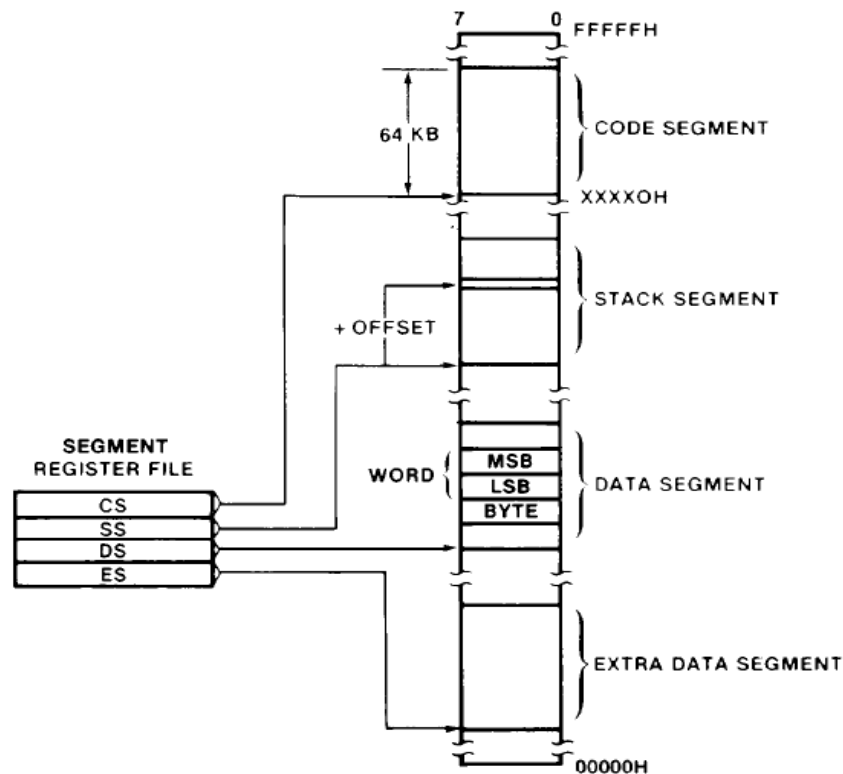


Fig.3 8086 Memory organization

FUNCTIONAL DESCRIPTION

Memory Organization

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries (See Figure 3).

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the ad-

ressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16-bit) operands can be located on even or odd address boundaries. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU will automatically execute two fetch or write cycles for 16-bit operands.

Memory Reference Used	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 override.

Certain locations in memory are reserved for specific CPU operations (See Figure 4). Locations from addresses FFFF0H through FFFFFH are reserved for operations including a jump to the initial system initialization routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be located. Locations 00000H through 003FFH are reserved for interrupt operations. Four-byte pointers consisting of a 16-bit segment address and a 16-bit offset address direct program flow to one of the 256 possible interrupt service routines. The pointer elements are assumed to have been stored at their respective places in reserved memory prior to the occurrence of interrupts.

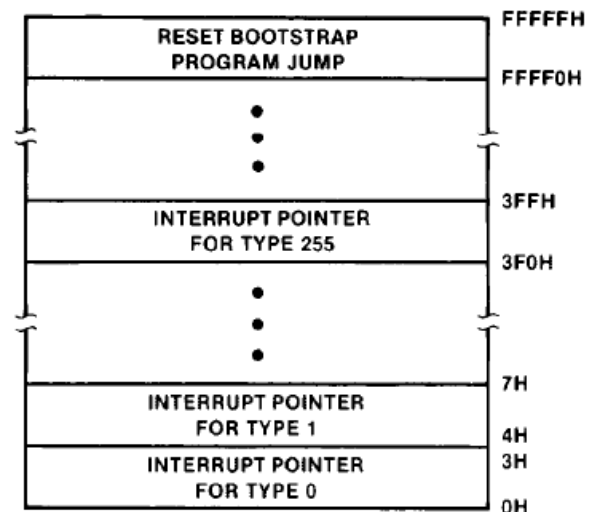


Fig.4 Reserved memory locations

Certain locations in memory are reserved for specific CPU operations (see Figure 3b). Locations from address FFFF0H through FFFFFH are reserved for operations including a jump to the initial program loading routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be. Locations 00000H through 003FFH are reserved for interrupt operations. Each of the 256 possible interrupt types has its service routine pointed to by a 4-byte pointer element consisting of a 16-bit segment address and a 16-bit offset address. The pointer elements are assumed to have been stored at the respective places in reserved memory prior to occurrence of interrupts.