**8086**

**Features of 8086**

**The most prominent features of an 8086 microprocessor are as follows −**

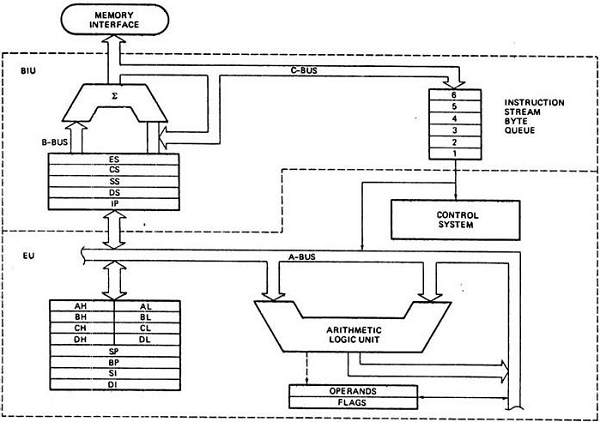
* **It has 256 vectored interrupts.**
* **It consists of 29,000 transistors.**
* **2 Stages of pipelining, i.e. Fetch Stage and Execute Stage, which improves performance.**
* **Fetch stage can prefetch up to 6 bytes of instructions and stores them in the queue.**
* **Execute stage executes these instructions**
* **Instruction queue capable of storing 6 instruction bytes from memory**
* **It was the first 16-bit processor having 16-bit ALU, 16-bit registers, internal data bus, and 16-bit external data bus**
* **Faster processing**
* **It is available in 3 versions based on the frequency of operation −**
  + **8086 → 5MHz**
  + **8086-2 → 8MHz**
  + **(c)8086-1 → 10 MHz**

**Architecture of 8086**

**The following diagram depicts the architecture of a 8086 Microprocessor –**

**8086 Microprocessor is divided into two functional units**

1. **EU (Execution Unit)**
2. **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.**

***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.** | |
| **Conditional Flags** | | |
| **Carry flag** | | **This flag indicates an overflow condition for arithmetic operations** |
| **Auxiliary flag** | | **When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set, i.e. carry given by D3 bit to D4 is AF flag. The processor uses this flag to perform binary to BCD conversion.** |
| **Parity flag** | | **This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1’s, then the Parity Flag is set. For odd number of 1’s, the Parity Flag is reset.** |
| **Zero flag** | | **This flag set to 1 when the result of arithmetic/logical operation is zero else it is set to 0.** |
| **Sign flag** | | **This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.** |
| **Overflow flag** | | **This flag represents the result when the system capacity is exceeded.** |
| **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-a-versa.** |
| **General purpose register** | | |
| **AX register** | | **Used as accumulator register. It is used to store operands for arithmetic operations.** |
| **BX register** | | **Used as base register to store the base address of the memory area of 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.** |
| **Stack pointer register** | | |
| **It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack** | | |

**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 relate to the Internal Bus.**

***Functional parts of 8086 microprocessors.***

|  |  |
| --- | --- |
| **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.** |
| **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.** |
| **Segment Buses** | |
| **CS** | **Code Segment. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.** |
| **DS** | **Data Segment. It consists of data used by the program and is accessed in the data segment by an offset address or the content of other register that holds the offset address.** |
| **SS** | **Stack Segment. It handles memory to store data and addresses during execution.** |
| **ES** | **Extra Segment. ES is additional data segment, which is used by the string to hold the extra destination data.** |
| **Instruction pointer** | |
| **It is a 16-bit register used to hold the address of the next instruction to be executed.**  **8086 was the first 16-bit microprocessor available in 40-pin DIP (Dual Inline Package) chip. Let us now discuss in detail the pin configuration of a 8086 Microprocessor.** | |

**8086 Pin Diagram**

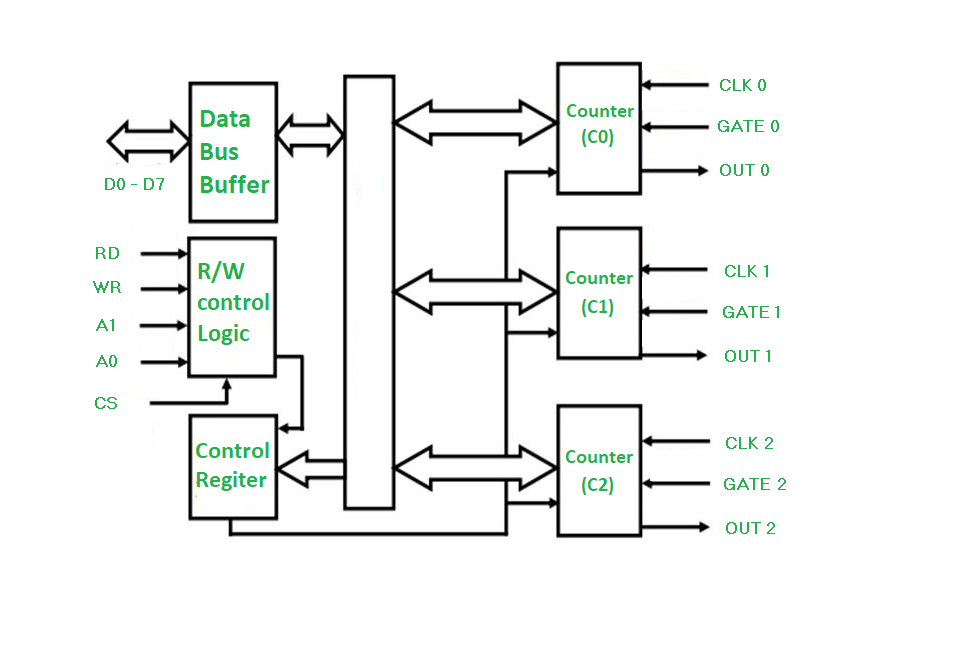
|  |  |  |
| --- | --- | --- |
| **Pin** | **Pin No** | **Function** |
| **VCC** | **40** | **Power Supply** |
| **GND** | **1 + 20** | **Ground** |
| **CLK** | **19** | **Provides timing to the processor for operations** |
| **AD Bus** | **(9-16)** | **Carries low order byte data** |
| **(2-7,39)** | **Carries higher order byte data** |
| **AStat Bus** | **(35-38)** | **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** | **34** | **BHE stands for Bus High Enable. 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.** |
| **Ready** | **22** | **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** | **21** | **Used to restart the execution. It causes the processor to immediately terminate its present activity.** |
| **INTR** | **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** | **17** | **Non-maskable interrupt. It is an edge triggered input, which causes an interrupt request to the microprocessor.** |
| **TEST** | **23** | **This signal is like wait state. When this signal is high, then the processor has to wait for IDLE state, else the execution continues.** |
| **HOLD** | **31** | **This signal indicates to the processor that external devices are requesting to access the address/data buses.** |
| **HLDA** | **30** | **Hold Acknowledgement signal. This signal acknowledges the HOLD signal.** |
| **Read** | **32** | **Used to read signal for Read operation** |
| **WR** | **29** | **Used for writing signal for Write Operation** |
| **M/IO** | **28** | **This signal is used to distinguish between memory and I/O operations.** |
| **DT/R** | **27** | **Data Transmit/Receive signal. It decides the direction of data flow through the trans receiver. When it is high, data is transmitted out and vice-a-versa.** |
| **DEN** | **26** | **Data Enable used to enable Transreceiver 8286. The transreceiver is a device used to separate data from the address/data bus** |
| **ALE** | **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** |
| **MN** | **33** | **Minimum/Maximum. Indicates what mode the processor is to operate in; when it is high, it works in the minimum mode and vice-versa.** |
| **QS1 and**  **QS0** |  | **Queue status signals. These signals provide the status of instruction queue. Their conditions are shown in the following table –**  **00 – No operation**  **01 – First Byte of Opcode from Queue**  **10 – Empty the Queue**  **11 - Subsequent byte from the queue** |
| **S0, S1, S2** | **26,27,28** | **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.** |
| **LOCK** | **29** | **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** |
| **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.** |

**8253-8254**

**Timer is a counter that counts pulses at the regular intervals and hence can indicate the time that has elapsed based on the count. Counter counts pulses that may or may not occur at regular intervals and hence gives the time of the number of times the event has occurred. There are two ICs that can act as the timers and they are as follows:**

|  |  |
| --- | --- |
| **8253** | **8254** |
| **Op Frequency – 0 to 2.6Mhz** | **Op Frequency – 0 to 10Mhz** |
| **NMOS Techmology** | **HMOS Technology** |
| **Read Back Command Unavailable** | **Read Back Command Available** |
| **Reads/Writes of same command not interleaved** | **Reads/Writes of same command interleaved** |

**Features of Programmable interval Timer**

* **Compatible with Intel and other Microprocessors**
* **Compatible with TTL**
* **8 MHZ High speed**
* **Handles input from DC to 10MHz for 82C54**
* **3 Independent 16 Bit Counter**
* **6 Programmable Courier Modes**
* **Binary/BCD Counting**
* **Status Read Back Command**
* **Low Power CHMOS at 10mA**

**8254 Block diagrams**

**8254 is a Programmable Interval Timer/Counter designed for use with microcomputer systems.**

**The characteristics are**

1. **General Purpose**
2. **Multi Timing Element treated as an array of I/O Ports in the Software**
3. **Generation of Accurate time delays under Software Control**
4. **Programmer configures the 8254 to match his requirements and programs one counter by desirable delay.**
5. **Software overhead is minimal and variable length delays can be accommodated**

**Data Bus Buffer – 3-State Bidirectional, 8 Bit Buffer used to interface 8254 to System Bus.**

**Read/Write Logic –**

**Accepts input from system bus and generates control signals for blocks of 8254**

**A low on RD tells 82C54 that the CPU is reading one of the counters**

**A low on WR tells 82C54 that the CPU is writing one of the counters**

**Both are under the control of the CS and cannot be activated when the CS is low**

**Operation Modes 8254**

**Operation Modes of 8254 are as follows:**

1. **MODE 0 - Interrupt on Terminal Count**
2. **MODE 1 - Monostable Multivibrator**
3. **MODE 2 - Rate/Pulse Generator**
4. **MODE 3 - Square Wave Generator**
5. **MODE 4 - Software Triggered Strobe**
6. **MODE 5 - Hardware Triggered Strobe**

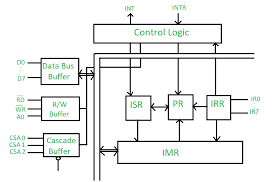
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Modes | MODE 0 | MODE 1 | MODE 2 | MODE 3 | MODE 4 | MODE 5 |
| Output Pin | **Low** | **High** | **High** | **High** | **High** | **High** |
| Count Value | **Loaded** | **Loaded** | **Loaded** | **Loaded** | **Loaded** | **Loaded** |
| Count  Enable | **Gate Pin**  **High** | **Gate Pin**  **Risen Edge** | **Gate Pin**  **High** | **Gate Pin**  **High** | **Gate Pin**  **High** | **Gate Pin**  **Risen Edge** |
| During Counting out remains | **Low** | **Low** | **High** | **½ Low**  **½ High** | **Low** | **High** |
| On Terminal Count | **Out is High and remains high** | **Out is High and remains high** | **One cycle before counting it goes low** | **Count is reloaded and process repeats** | **One cycle after counting it goes high** | **One cycle after counting it goes high** |
| During Counting GATE made LOW | **Disables counting and restarts when made HIGH** | **Doesn’t Effect Counting** | **Disables counting and restarts when made HIGH** | **Disables counting and restarts when made HIGH** | **Disables counting and restarts when made HIGH** | **Counting Triggered and restarts again** |

***8259 PROGRAMMABLE INTERRUP CONTROLLER***

**When we need multiple sources, we need to use an external device called as Priority Interrupt Controller (PIC). PIC helps to increase the Interrupt Handling Capacity of the Microprocessor. 8259A is the commonly used PIC which can easily help in handling Non-Vectored Interrupt of 8086-INTR.**

**FEATURES OF 8259**

1. **Programmable Interrupt used to work with 8086 and 8085.**
2. **Can Handle Edge as well as Level Triggered Interrupt**
3. **Flexible Priority Structure**
4. **Interrupts Can be Masked Individually**
5. **Vector Address of Interrupt is Programmable**
6. **Single 8259 can handle 8 Interrupts while cascaded configuration of 1 Master 8259. Similarly, 8 Slave 8259 can handle 64 Interrupts**

****

|  |  |
| --- | --- |
| **BLOCK** | **INFORMATION** |
| **Data Bus Buffer** | * **Bi-Directional Buffer used to interface the internal data bus of 8259 with any external data bus.** * **Used to send read and control signals** * **Used to read interrupt type from the 8259** |
| **Read Write Logic** | * **Used to accept RD,WR,Ao,CS Signal** * **Used to control Data flow on Data Bus** * **Holds Initialization Command Words (ICWs) and Operational Commands Words (OCWs)** |
| **Comparator or Cascade Buffer** | * **Used in cascade mode operation** * **CAS2, CAS1, CAS0:  Output for Slave and Input for Master.**   **Using these 3 Lines there are 8 Possible Interrupts (23=8)**  **The Master Sends Address of the Slave on these lines**  **The Slaves read the address on these lines.**   * **SP/EN:   EN – Used to Enable the Buffer in Buffered Mode**   **SP – Functions as Output Line in Non-Buffered Mode** |
| **Interrupt Request Register** | * **IRR – 8 Bit Register having 1-Bit for each interrupt lines IR7-IR0** * **Interrupt request occurs and the corresponding bit is set in the IRR** |
| **Interrupt Mask Register** | * **IMR- 8 Bit Register storing Masking Information of Interrupt IR7-IR0** * **This is written by the Programmer** |
| **In Service Register** | * **InSR - 8 Bit Register storing info about Interrupt Request being serviced** |
| **Priority Resolved** | * **Examines IRR,IMR,InSR and examines which interrupt has the highest priority of them all and send it to the micro-processer** |
| **Control Logic** | * **INT Output Signal connected to the INTR of Microprocessor** * **Also has the INTA input connected to the INTA of the Microprocessor** * **Also used to control the remaining blocks** |

**PIN CONFIGURATION of 8259**

|  |  |  |
| --- | --- | --- |
| **PIN** | **NUMBER** | **INFORMATION** |
| **Vcc** | **28** | **+5V Power Supply** |
| **GND** | **14** | **Ground** |
| **CS** | **1** | **Chip Select – Low on this pin enables RD and WR Communication** |
| **WR** | **2** | **Write – Low on this command enables to accept command words from CPU** |
| **RD** | **3** | **Read - Low on this command enables to release status onto data bus** |
| **D7-D0** | **4-11** | **Bidirectional Data Bud for data transfer** |
| **CASo-CAS2** | **12,13,15** | **Cascade Lines to control multiple 8259A structure** |
| **SP/EN** | **16** | **Slave Program or Enable Buffer**  **EN – Used to Enable the Buffer in Buffered Mode**  **SP – Functions as Output Line in Non-Buffered Mode** |
| **INT** | **17** | **Interrupt Pin goes high when a valid interrupt request is asserted and is used to interrupt the CPU** |
| **IR0-IR7** | **18-25** | **Interrupt Request is asserted by pin going from low to high** |
| **INTA** | **26** | **Enable 8259 vector data into data bus by a Interrupt Acknowledgement** |
| **Ao** | **27** | **AO Address Line acts in conjunction of CS, WR,RD, and is used to decipher various command words** |

**END OF INTERRUPT**

**There are 3 different ways of sending END OF INTERRUPT command and they are as follows:**

|  |  |
| --- | --- |
| **Automatic EOI** | **No command needed here**  **In the 3rd INTA cycle corresponding bit in the InSR is reset** |
| **Non-Specific EOI** | **Command is sent to 8259 at the end of service routine**  **This would clear the bit of the currently serviced interrupt in InSR** |
| **Specific EOI** | **Command is sent to 8259 at the end of service routine**  **This would clear the bit of the specified serviced interrupt in InSR** |

**OPERATION MODES**

**There are 2 different ways of Operating Modes of 8259 and they are as following**

|  |  |
| --- | --- |
| **Interrupt Driven** | **8259 interrupts processor with the INT Pin whenever it encounters an interrupt** |
| **Polled Mode** | **Here the INT mode is not used and checks the interrupt request by issuing the poll command. The micro-processor reads contents of 8259 and issues the poll command. During read operation provides polled words and sets the InSR bit of highest active interrupt** |

**PRIORITY MODES**

1. **Fully Nested Modes**
   1. **Default Mode of 8259**
   2. **Fixed Priority Mode**
   3. **IR0 – Highest and IR7 – Lowest**
2. **Special Fully Nested Modes**
   1. **Used for Master 8259 in cascading mode.**
   2. **The priority for the same are same as Fully Nested Modes**
   3. **Consider a large system that uses cascading 8259 where the interrupt level of each interrupt must be considered. An interrupt input to the slave causes the slave to place an interrupt request to the master and on one of master’s Input**
   4. **Interrupts to the same slave will not be recognized and thus disabling further interrupts. This can be prevented using SFNM**
3. **Rotating Priority Mode**
   1. **Automatic Rotation Mode**
      1. **Preferred when multiple interrupts have the same priority**
      2. **After a device requests service, then it gets the lowest priority**
      3. **All other priorities rotate subsequently**
   2. **Specific Rotation Mode**
      1. **Here the user can fix the priorities**
4. **Special Mask Mode**
   1. **8259 disables interrupt requests lower or equal to the interrupt which is currently under service. SMM allows interrupts of all levels except one currently in service**
5. **Poll Mode**
   1. **Here INT line of 8259 is disabled**
   2. **Microprocessor gives Poll Command to 8259 using OCW3 and in return gets a Poll Word and then the service of interrupt takes place**
   3. **Helps when we need expand number of interrupts more than 64**
6. **Buffered Mode**
   1. **SP/EN becomes low during the INTA cycle and this is used to enable the buffer**