Course Title: Microprocessor Interfacing & Assembly Language

Mid Term Examination

Spring Semester- 2021

Instructor Name: Sania Urooj

Faculty of: Computer Science

Program: BSCS

Student Name: M. Mashood Latif

Registration number: BSCS 316/3-1/Gh002

Q:1 (a) Explain the difference between assembly and machine language.

Ans

Machine Language

Machine language is made up of 0s and 1s and cannot be read by humans. As a result, only machines can comprehend. So, whichever language you use (assembly, high level) to write code, it is transformed to machine level language so that it can be understood by the machine.

Assembly Language

Assemble Language is a second-generation programming language that is utilized in computer systems and human readable. In assembly language, instead of machine language instructions, a programmer employs symbolic instructions and descriptive labels for data items and memory locations. An assembly language program is written using tight rules and then translated into machine code by an assembler.

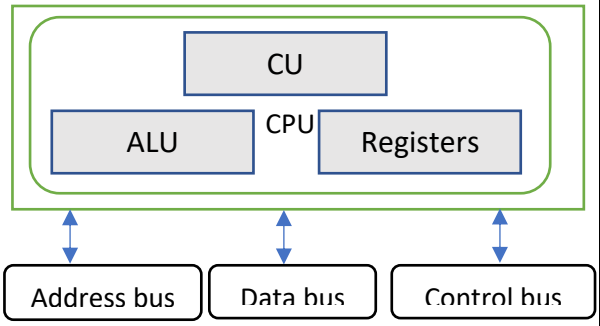
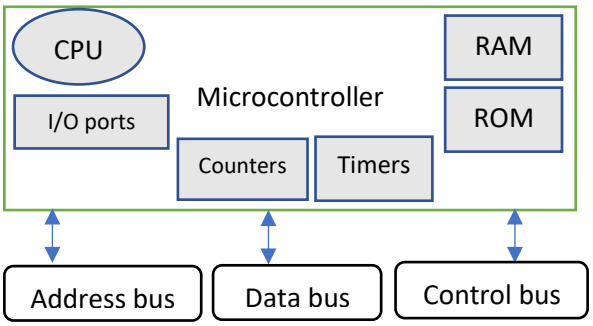
Difference between assembly & machine language

	Assembly Language	Machine Language
Readability	Assembly Language has a high readability since it is written in English, which is easily understood by humans.	Machine Language has far less readability than Assembly Language since it is written in binary code, which a typical person cannot comprehend. However, this is the computer's actual language.
Platform Dependency	Because the Assembly Language is platform-dependent, the majority of programs are currently coded in a third-generation language.	Machine Language differs depending on the platform. Machine language code varies the platform to platform.
Quality	Assembly Language reduces the chances of an error occurring.	Machine Language has a high probability of containing errors.
Modifiable	The improvement of software is required for its survival. To take use of this benefit, the language must be easily adaptable. Furthermore, the Assembly Language is easily modifiable, and its support changes over time.	
Need of Compiler	In the case of Assembly Language, an interpreter, also known as an assembler, is required to translate the code into machine code in the form of bits and bytes.	Machine Language does not require an interpreter or compiler because it is the computer's native language. As a result, a computer does not require a compiler to comprehend its own language.
Memorability	Assembly Language has a high level of memorability because variable names are easier to recall than binary code.	The binary code is extremely hard to remember.

Q:1 (b) What are the major differences between microprocessors and microcontrollers?

Ans

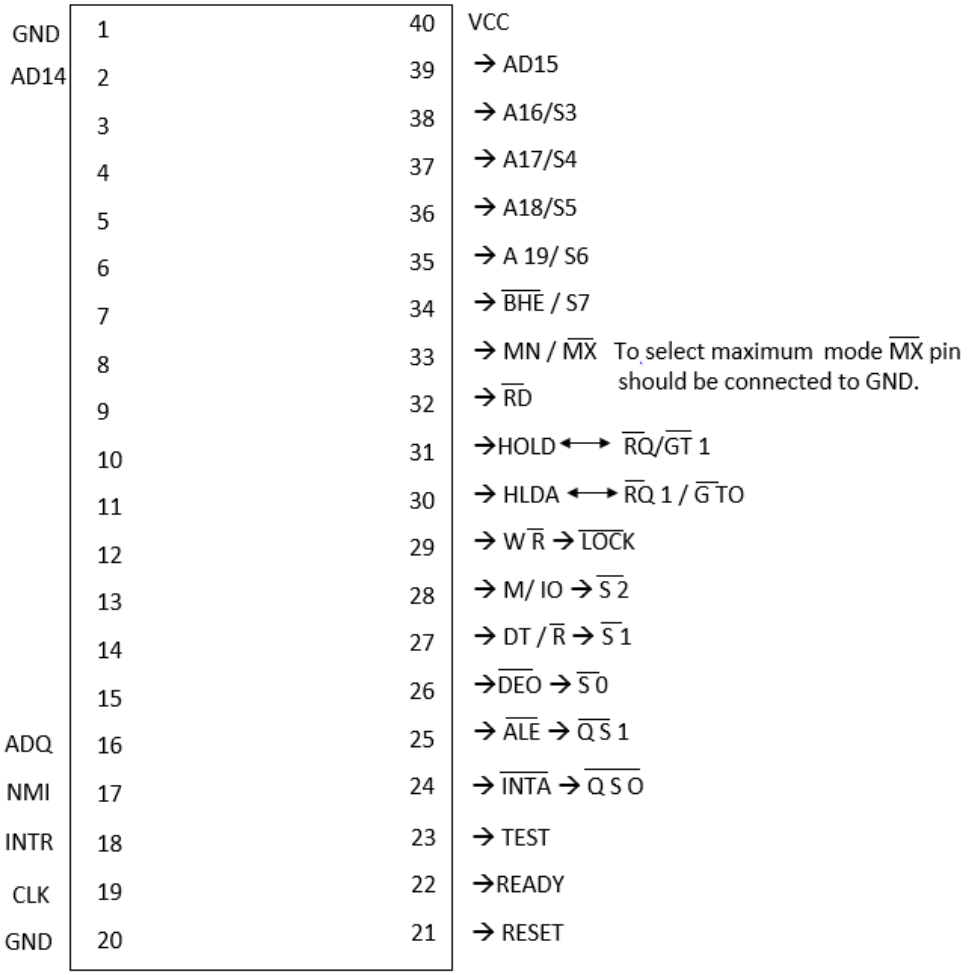
A microprocessor is used for applications that demand intense processing, whereas a microcontroller is utilized to accomplish a specific task.

Microprocessor	Microcontroller
	
The function of a central processing unit (CPU) is assimilated into a single integrated circuit by a microprocessor (IC).	A microcontroller can be thought of as a little computer with a CPU and other components that make it a computer.
Microprocessors are mostly utilized in the development of general-purpose systems ranging in size from tiny to big and complicated systems such as supercomputers.	Microcontrollers are used in devices that are automatically controlled.
Microprocessors are basic components of personal computers.	Embedded systems frequently use microcontrollers.
The microprocessor's computational capacity is extremely high. As a result, it is capable of doing difficult tasks.	When compared to microprocessors, they have a lower computing capacity. Typically used for easier jobs.
A microprocessor-based system is capable of completing a wide range of tasks.	A microcontroller-based system can carry out a single or a small number of functions.
Complex mathematical computations involving floating point can be handled quickly and easily.	They conduct floating point computations with software, which slows down the system.
The main function of a microprocessor is to repeat the instruction cycle.	A microcontroller manages its surroundings based on the output of the instruction cycle, in addition to executing the duties of retrieve, decode, and execute.
A microprocessor must be connected externally to other components such as memory (RAM and ROM) and input/output ports in order to build or develop a system (computer).	A microcontroller's integrated circuit (IC) contains memory (both ROM and RAM) as well as other components such as timers and I/O devices.
A system built with a microprocessor has a high in total cost.	Because all of the components are easily available, the cost of a system designed with a microcontroller is lower.
Because of the external devices, power consumption and dissipation are generally high.	Power consumption is less.
The frequency of the clock is typically in the gigahertz range.	The frequency of a clock is rarely in the Mega Hertz range.
Instruction throughput takes priority over interrupt latency.	Microcontrollers, on the other hand, are built to minimize interrupt latency.
Have a few instructions on bit manipulation.	In microcontrollers, bit manipulation is a strong and widely used feature.
Microprocessors are typically not employed in real-time systems since they are overly reliant on a number of other components.	Because microcontrollers are single-programmed, self-contained, and task-oriented devices, they are employed to handle real-time tasks.

Q:2 (a) Provide the name of pins that are used as control signals and also write down their functions.

Ans

The first 16-bit microprocessor available in a 40-pin DIP (Dual Inline Package) chip was the 8086.



Power supply and frequency signals

It operates using a 5V DC supply at VCC pin 40 and ground at VSS pins 1 and 20.

Clock signal

Pin-19 is used to give the clock signal. It gives the CPU timing for operations. Its frequency varies according on the version, with 5MHz, 8MHz, and 10MHz being the most common.

Address/data bus

AD0-AD15. There are a total of 16 address/data buses. Low-order byte data is carried by AD0-AD7, while higher-order byte data is carried by AD8AD15. It carries 16-bit address during the first clock cycle and 16-bit data after that.

Address/status bus

A16-A19/S3-S6. The four address/status buses are as follows. It carries 4-bit address during the first clock cycle and status signals later.

S7/BHE

Bus High Enable is the abbreviation for BHE. It's located at pin 34 and is used to notify data flow via data bus D8-D15. During the first clock cycle, this signal is low; after that, it is active.

Read($\overline{RD}$)

It's on pin32, and it's used to read the signal for the Read function.

Ready

Pin 22 is where you'll find it. It's a signal from I/O devices indicating that data has been sent. It's a high-intensity signal that's active. When this value is high, it means the device is ready to send data. When it is low, it means you're in a waiting mode.

RESET

It's on pin 21 and it's utilised to restart the execution. It enables the processor's current activity to be terminated immediately. To RESET the microprocessor, this signal is active high for the first four clock cycles.

INTR

It can be found at pin 18. It's an interrupt request signal that's sampled on each instruction's last clock cycle to see if the CPU considers it an interrupt or not.

NMI

It is available at pin 17 and stands for non-maskable interrupt. It's an edge-triggered input that sends a Microprocessor interrupt request.

TEST

This signal, which is similar to a wait state, is available at pin 23. When this signal is high, the processor must wait for the IDLE state to be reached; otherwise, the execution will proceed.

MN/MX

It is available at pin 33 and stands for Minimum/Maximum. It specifies the processor's operating mode; when set to high, it operates in the minimal mode, and vice versa.

INTA

It's an interrupt acknowledgement signal, and pin 24 is where you'll find it. The interrupt is acknowledged by the microprocessor when it receives this signal.

ALE

It is available at pin 25 and stands for address enable latch. Every time the processor performs an operation, a positive pulse is generated. This signal indicates whether the address/data lines have a valid address.

DEN

It's available at pin 26 and stands for Data Enable. It's used to make Transreceiver 8286 functional. The data is separated from the address/data bus by the transreceiver.

DT/R

It is available at pin 27 and stands for Data Transmit/Receive signal. It determines the data flow direction through the transreceiver. Data is transmitted out when it is high and vice versa when it is low.

M/IO

This signal is used to differentiate memory operations from I/O operations. It signals an I/O operation when it is high and a memory operation when it is low. It can be found at pin 28.

WR

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

HLDA

It is present at pin 30 and stands for Hold Acknowledgement signal. The HOLD signal is acknowledged by this signal.

HOLD

External devices are demanding access to the address/data buses, as indicated by this signal. Pin 31 is where you will find it.

QS1 and QS0

These are queue status signals, which may be found on pins 24 and 25. These signals indicate the current state of the instruction queue. The following table summarises their conditions.

QS_0	QS_1	*Status*
0	0	No operation

0	1	First byte of opcode from the queue
1	0	Empty the queue

S0, S1, S2

These are the status signals that the Bus Controller 8288 uses to create memory and I/O control signals based on the current state of operation. These can be found at pins 26, 27, and 28. The table below shows their current state.

S_2	S_1	S_0	*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

This signal advises to other processors that they should not ask the CPU to leave the system bus when it is active. It is available at pin 29 and is triggered by using the LOCK prefix on any instruction.

RQ/GT1 and RQ/GT0

The other processors utilise these Request/Grant signals to ask the CPU to release the system bus. When the CPU receives the signal, it sends an acknowledgement. RQ/GT0 takes precedence over RQ/GT1.

Q:2 (b) What is the purpose of MN/Mx pin? Explain.

Ans

MN/Mx

This pin shows the processor's operating mode. The 8086 generates all bus control signals in minimal mode. To create all of the bus control signals in maximum mode, the three status signals must be decoded.

MN/MX is an input pin that is used to select which mode to use. The 8086 works in minimal mode when MN/MX is high. The 8086 is set up in this mode to enable a modest single-processor system with a few peripherals connected to the system bus. When MN/MX is low, the 8086 is set up to work with a multiprocessor system.

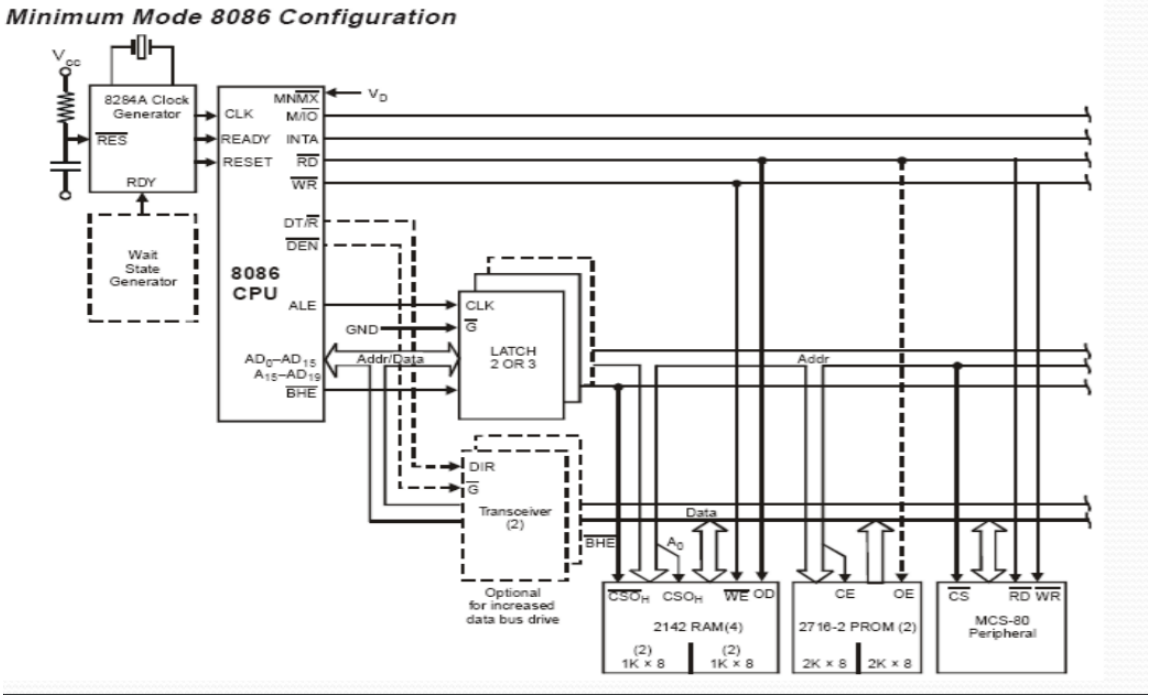
Q:2 (b) Define the minimum and maximum mode of 8086 microprocessor.

Ans

Minimum Mode 8086 System

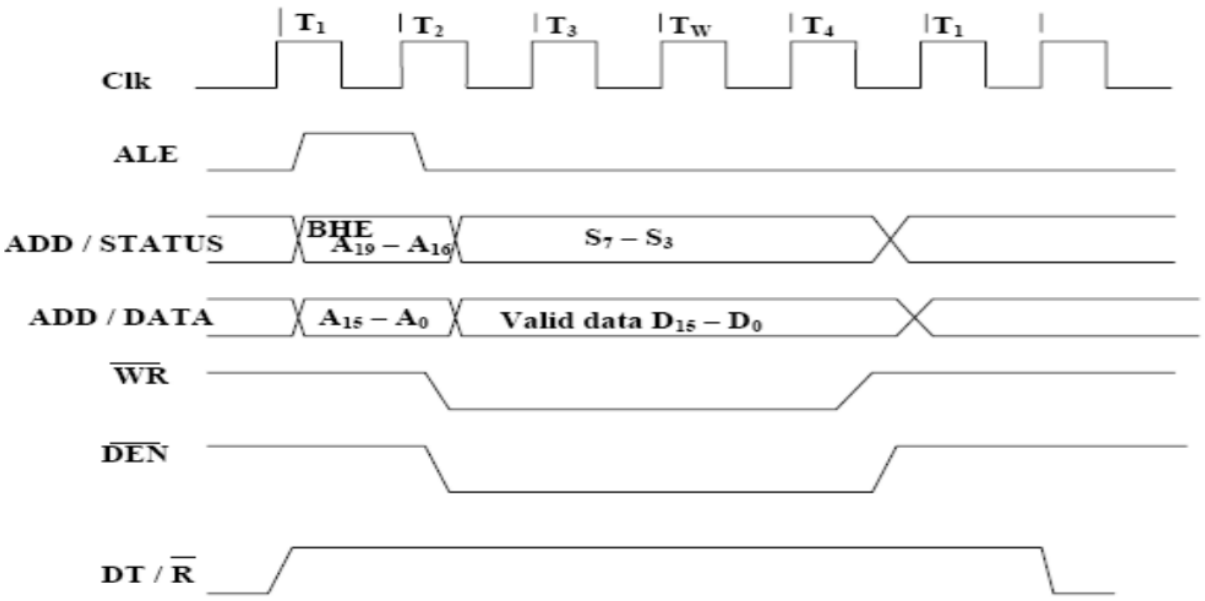
- The 8086 microprocessor is set to minimal mode by connecting the MN/MX pin to logic 1.
- In this mode, the microprocessor chip sends out all of the control signals. In the minimum mode system, there is only one microprocessor.
- Latches, transreceivers, a clock generator, memory, and I/O devices make up the rest of the system.
- Latches are D-type flip-flops with buffered outputs, such as the 74LS373 or 8282. They are controlled by the ALE signal generated by the 8086 and are used to separate the valid address from the multiplexed address/data signals.

Minimum Mode Configuration For 8086



- Transreceivers are bidirectional buffers, which are also known as data amplifiers. They must distinguish between genuine data and time multiplexed address/data signals. Two signals, DEN and DT/R, are used to control them.
- The DEN signal specifies whether data is coming from or going to the CPU.
- The system includes monitor memory as well as user programme storage. EPROMs are typically used for monitor storage, while RAM is typically utilised for user programme storage. I/O devices can be found in a system.
- The fetch and read cycles for opcodes are comparable. As a result, the timing diagram can be divided into two sections: the first is for the read cycle, and the second is for the write cycle.
- T1 starts the read cycle by asserting the address latch enable (ALE) signal as well as the M/IO signal. The valid address is latched on the local bus at the signal's negative going edge.
- The BHE and A0 signals control whether low, high, or both bytes are addressed. The M/IO signal represents a memory or I/O action from T1 to T4.
- T2 removes the address from the local bus and sends it to the output. After that, the bus is tristated. T2 additionally activates the read (RD) control signal.
- The read (RD) signal causes the address device's data bus drivers to be enabled. The legitimate data is available on the data bus when RD becomes low.
- The READY line will be driven high by the designated device. The addressed device will tristate its bus drivers again when the processor returns the read signal to a high level.
- The assertion of ALE and the emission of the address also start a write cycle.

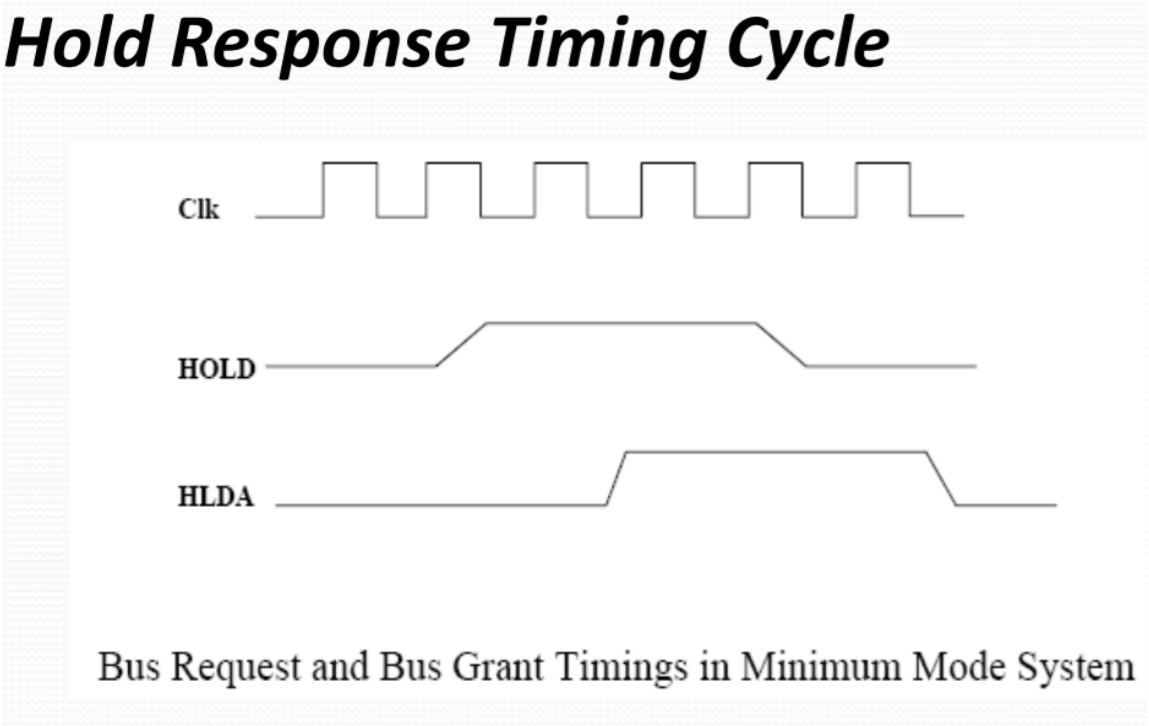
- To signify a memory or I/O operation, the M/IO signal is asserted once more. In T2, the processor transmits the data to be written to the addressed place after providing the address in T1.
- The data is kept on the bus until it reaches the middle of the T4 state. The WR activates at the start of T2 (unlike the RD, which is somewhat delayed in T2 to allow for floating).
- The BHE and A0 signals are used to pick the correct memory byte or bytes or I/O word to read or write.
- As shown in the table below, the M/IO, RD, and WR signals indicate the type of data transport.



Hold Response sequence

- At the leading edge of each clock pulse, the HOLD pin is tested.
  If it is received active before T4 of the previous cycle or during T1 of the current cycle, the CPU activates HLDA in the next clock cycle, and the bus will be given to another requesting master for subsequent bus cycles.
- The processor does not retake control of the bus until the asking master does not drop the HOLD pin low.

- When the asking master drops the request, the processor drops the HLDA at the trailing edge of the next clock.

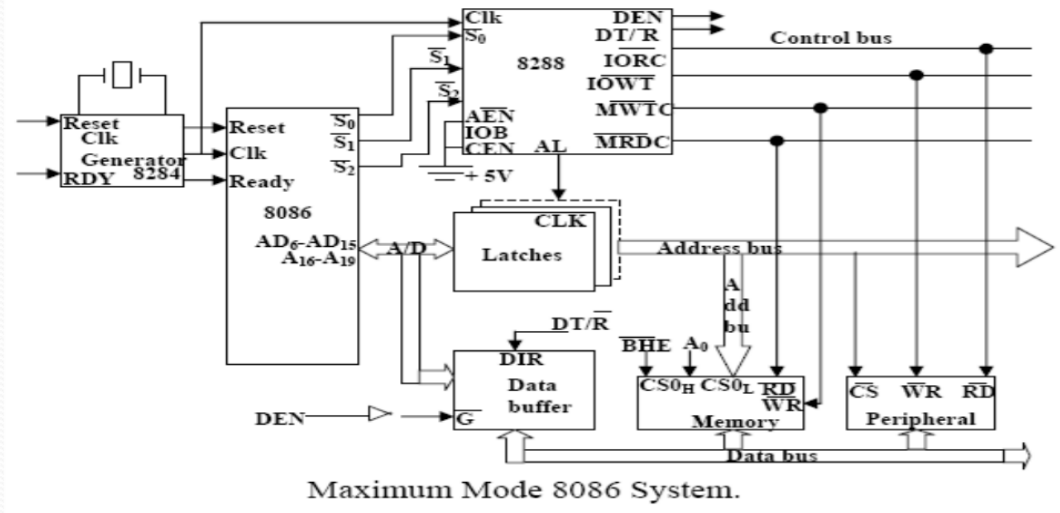


Maximum Mode 8086 System

- The 8086 is operated in maximal mode by connecting the MN/MX pin to ground.
- The processor generates the status signals S2, S1, and S0 in this mode.
  This status information is used by another chip called the bus controller to generate the control signal.
- In the maximal mode, the system configuration may include more than one microprocessor. The system's components are the same as in the minimum mode system.
- The bus controller chip IC8288's primary job is to extract control signals such as RD and WR (for memory and I/O devices), DEN, DT/R, ALE, and so on, from information provided by the processor on the status lines.
- S2, S1, S0, and CLK are input lines on the bus controller chip.
  The CPU controls these 8288 inputs.
- ALE, DEN, DT/R, MRDC, MWTC, AMWC, IORC, IOWC, and AIOWC are the outputs. For multiprocessor systems, the AEN, IOB, and CEN pins are especially useful.
- In general, AEN and IOB are grounded. The CEN pin is commonly connected to a +5V supply. The value of the MCE/PDEN output is determined by the state of the IOB pin.
- INTA is a pin that sends two interrupt acknowledge pulses to the interrupt controller or interrupting device.
- IORC and IOWC are signals for I/O read and write commands, respectively. An IO interface can read or write data from or to the address port using these signals.
- Memory read command and memory write command signals are MRDC and MWTC, respectively, and can be utilised as memory read or write signals.

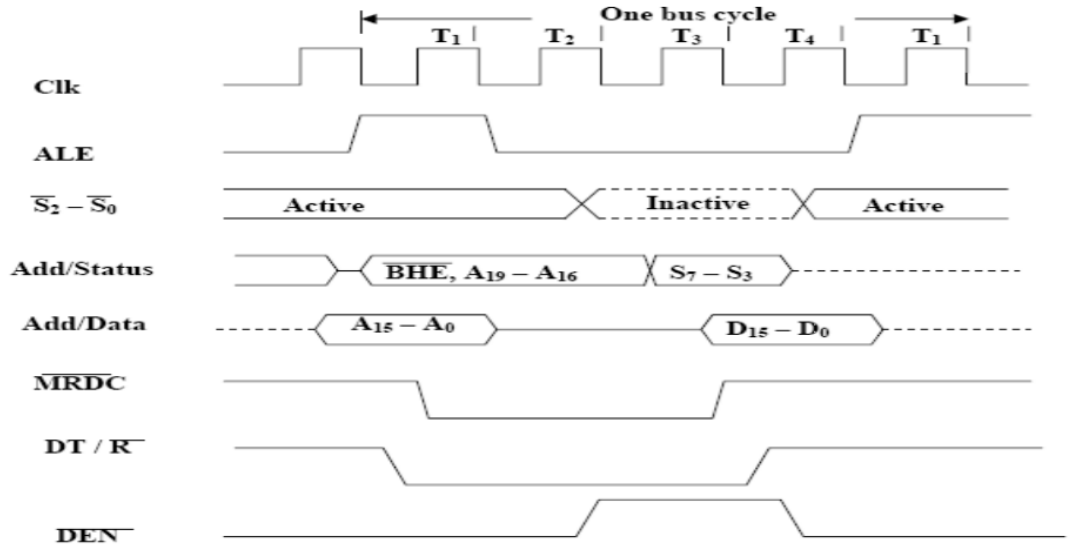
- All of these command signals tell the memory whether or not to accept or send data to or from the bus.
- The sole difference between the lowest and maximum modes in the timing diagram is the status signals used, as well as the accessible control and advanced command signals.

Maximum Mode Configuration For 8086

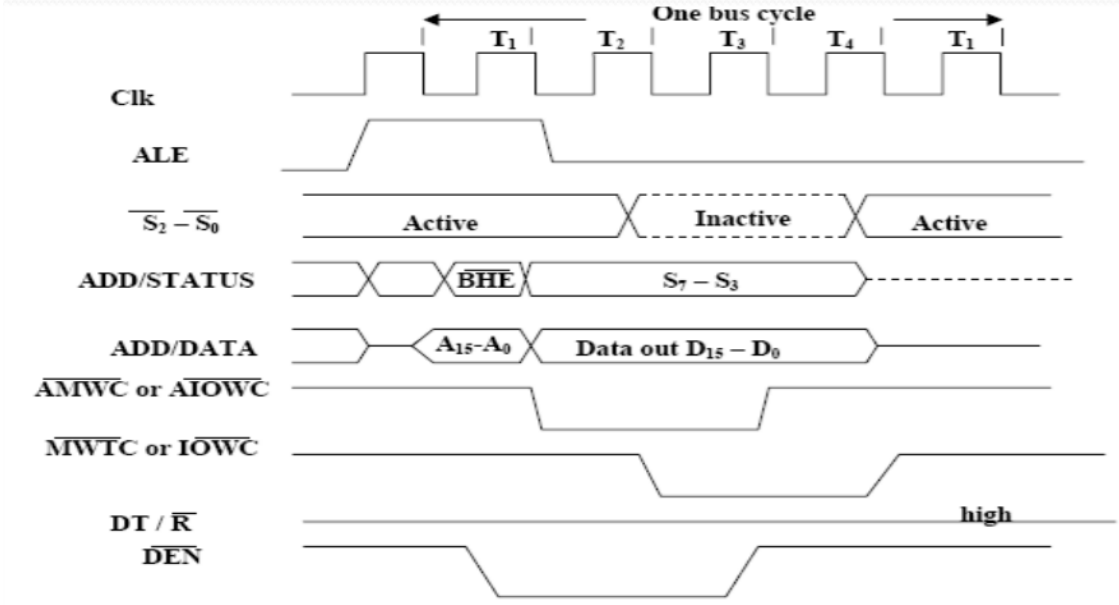


Maximum Mode 8086 System.

- At the start of the bus cycle, R0, S1, and S2 are set. During T1, the 8288 bus controller will generate a pulse similar to the ALE and apply a needed signal to its DT / R pin.
- In T2, 8288 will set DEN=1 to enable transceivers and activate MRDC or IORC for an input. Until T4, these signals are active.
- From T2 to T4, the AMWC or AIOWC is engaged, and from T3 to T4, the MWTC or IOWC is activated.
- S0 to S2 status bits are active till T3 and then become inactive throughout T3 and T4.
- Wait state will be introduced between T3 and T4 if reader input is not activated before T3.



Memory Read Timing in Maximum Mode



Memory Write Timing in Maximum mode.

Q:3 (a) Define the purpose of BIU and Execution Unit.

Ans

Memory segmentation

- The memory of the 8086 is segmented to improve execution and fetching speed.

- It has a 20-bit address bus that can address 1MB of memory and divides it into four 64-kilobyte chunks.
- Within the 1MB memory, the 8086 only works with four 64KB segments.

The Bus Interface Unit (BIU) and the Execution Unit (EU) make up the Intel 8086's internal design (EU). These are explained in the next paragraphs.

(BIU)The Bus Interface Unit

It uses the System Bus to connect the 8086 to external memory and I/O devices. To transmit data between memory and I/O devices, it executes various machine cycles such as memory read, I/O read, and so on.

The BIU is responsible for the following tasks:

- It generates the physical address for memory access, which is 20 bits long.
- It accesses the memory to retrieve instructions.
- It transports data between the memory and the I/O.
- The 6 byte prefetch instruction queue is maintained (supports pipelining).

The 4 Segment registers, the Instruction Pointer, a prefetch queue, and an Address Generation Circuit are all found in BIU.

IP (Instruction Pointer):

- It's a register with 16 bits. It stores the offset of the Code Segment's following instructions.
- After each instruction byte is fetched, IP is increased.
- When a branch instruction occurs, IP receives a new value.
- The 20 bit physical address of the Code Segment is obtained by multiplying CS by 10H.
- The following instruction's address is determined as CS x 10H + IP.

Example:

- CS = 4321H IP = 1000H
- then CS x 10H = 43210H + offset = 44210H

This is the instruction's address.

Code Segment register:

The base address for the Code Segment is stored in CS. The Code Segment stores all programmes, which can be accessed via IP.

Data Segment register:

The Data Segment's base address is stored in DS.

Stack Segment register:

The Stack Segment's base address is stored in SS.

Extra Segment register:

The Extra Segment's base address is stored in ES.

Address Generation Circuit:

A Physical Address Generation Circuit is included in the BIU.

It uses the algorithm to construct the 20-bit physical address using Segment and Offset addresses:

Physical Address

= Segment Address x 10H + Offset Address

6 Byte Pre-fetch Queue:

- It's a six-byte queue (FIFO).
- Pipelining is the process of fetching the next instruction (via BIU from CS) while executing the current one.
- When a branch instruction occurs, this variable is flushed.

(EU)The Execution Unit:

General purpose registers, the ALU, Special purpose registers, Instruction Register and Instruction Decoder, and the Flag/Status Register are the key components of the EU.

- The ALU fetches instructions from the Queue in BIU, decodes them, and performs arithmetic and logic operations.
- Sends control signals to the CPU for internal data transfer processes.
- Request signals are sent to the BIU in order to gain access to the external module.
- It works with T-states (clock cycles) rather than machine cycles.

AX, BX, CX, and DX are four 16-bit general-purpose registers on the 8086. During the execution, keep track of interim values. There are two 8-bit components in each of these (higher and lower).

- **AX register:**

During multiplication and division operations, it keeps operands and results. During String operations, it can also be used as an accumulator.

- **BX register:**

  In indirect addressing modes, it stores the memory address (offset address).

- **CX register:**

  It keeps track of commands such as loop, rotate, shift, and string.

- **DX register:**

  It's used with AX to retain 32-bit numbers during division and multiplication.

Arithmetic Logic Unit (16 bit):

Performs **8 and 16 bit** arithmetic and logic operations.

Special purpose registers (16-bit):

- **Stack Pointer:**

  Stack top gets points. Stack is in the Stack Segment and is utilised in commands like PUSH, POP, CALL, RET, and so on.

- **Base Pointer:**

  Any offset address in the stack segment can be stored in BP. It's utilised to go to different parts of the stack at random times.

- **Source Index:**

  During string operations, it keeps track of the offset address in Data Segment.

- **Destination Index:**

  It holds offset address in Extra Segment during string operations.

Instruction Register and Instruction Decoder:

The EU fetches an opcode from the queue into the instruction register. The instruction decoder decodes it and sends the information to the control circuit for execution.

Flag/Status register (16 bits):

It has 9 flags that help change or recognize the state of the microprocessor.

6 Status flags:

1. carry flag(CF)
2. parity flag(PF)
3. auxiliary carry flag(AF)
4. zero flag(Z)
5. sign flag(S)
6. overflow flag (O)

Status flags are updated after every arithmetic and logic operation.

3 Control flags:

1. trap flag(TF)
2. interrupt flag(IF)
3. direction flag(DF)

These flags can be set or reset using control instructions like CLC, STC, CLD, STD, CLI, STI, etc.

The Control flags are used to control certain operations.

Q3 (b) What's the purpose of BHE pin of 8086 microprocessor?

Ans

BHE (Active Low)/S_7 (Output)

Bus High Enable/Status

It is used to enable data onto the most significant half of data bus, D_8-D_{15}. 8-bit device connected to upper half of the data bus use BHE (Active Low) signal. It is multiplexed with status signal S_7.