1) Explain the function of the following minimum mode pins

1. DT/R-

**–** Output signal from the processor to control the direction of data flow through the data transceivers

(ii)HLDA

Hold indicates that another master is requesting a local bus "HOLD". To be acknowledged, HOLD must be active HIGH as an acknowledgement in the middle of the T1-clock cycle. Simultaneous with the issue of HLDA, the processor will float the local bus and control lines. After "HOLD" is detected as being Low, the processor will lower the HLDA and when the processor needs to run another cycle, it will again drive the local bus and control lines.

1. M/IO

This pin is used to distinguish a memory access or an I/O accesses. When this pin is Low, it accesses I/O and when high it access memory. M / IO becomes valid in the T4 state preceding a bus cycle and remains valid until the final T4 of the cycle. M/IO floats to 3 - state OFF during local bus "hold acknowledge".

b ) Explain the fetch execute cycle of the 8086

(i) The BIU outputs the contents of the instruction pointer register (IP) onto the address bus, causing the selected byte or word to be read into the BIU.  
  
 (ii) Register IP is incremented by 1 to prepare for the next instruction fetch.  
  
 (iii)Once inside the BIU, the instruction is passed to the queue. This is a first-in, first-out storage register sometimes likened to a "pipeline".  
  
 (iv) Assuming that the queue is initially empty, the EU immediately draws this instruction from the queue and begins execution.  
  
 (v)While the EU is executing this instruction, the BIU proceeds to fetch a new instruction. Depending on the execution time of the first instruction, the BIU may fill the queue with several new instructions before the EU is ready to draw its next instruction.

2) Using an appropriate example explain the following assembly instructions:

(i)ADD

The add instruction adds the contents of the source operand to the destination operand.

Example ;Consider J:= K + M

MOV AX, K

ADD AX, M

MOV J, AX

(ii)SUB

The SUB (subtract

(iii)CMP

The cmp (compare) instruction is identical to the sub instruction with one crucial difference it does not store the difference back into the destination operand

(b) Describe the instruction format of the 8086.

Instructions consists of optional instruction prefixes ,one or two primary opcode bytes possibly an address specifier consisting of the mod R/M byte and the SIB (scale index base) byte ,a displacement if required ,and an immediate data field required.

Smaller encoding field can be defined within the primary opcode or opcodes .

Most instructions that can refer to an operand in memory have an addressing form byte following the primary upcode bytes .adressing forms can include a displacement immediately following either mod R/Mor S/B byte.

If the instruction specifies an immediate operand ,the immediate operand always follows and displacement bytes

3) Distinguish between the following commands

Loop- the instruction decrements the cx register and then branches to the target location if the cx register does not contain zero.

Loopz- will branch to the target address if cx is not zero and the zero flag is set.

Loopne - loop while not equal to zero.) Repeats while cx is not zero and the zero flag is clear

Retn- jump to within the existing code segments

Retf- jump to a different code segment

.

4) Explain the concept of pipelining in 8086. Discuss its advantages and disadvantages.

A technique used in advanced [microprocessors](http://www.webopedia.com/TERM/M/microprocessor.html) where the microprocessor begins [executing](http://www.webopedia.com/TERM/E/execute.html) a second [instruction](http://www.webopedia.com/TERM/I/instruction.html) before the first has been completed. That is, several instructions are in the pipeline simultaneously, each at a different processing stage. The pipeline is divided into segments and each segment can execute its operation concurrently with the other segments. When a segment completes an operation, it passes the result to the next segment in the pipeline and fetches the next operation from the preceding segment. The final results of each instruction emerge at the end of the pipeline in rapid succession.

A) Advantages of pipelining

 (i) The cycle time of the processor is reduced, thus increasing instruction issue-rate in mostcases.2.

(ii) Some combinational circuits such as adders or multipliers can be made faster by adding more circuitry. If pipelining is used instead, it can save circuitry vs. a more complex combinational circuit

B) Disadvantages of pipelining

(i) A non-pipelined processor executes only a single instruction at a time. This prevents branch delays (in effect, every branch is delayed) and problems with serial instructions being executed concurrently. Consequently the design is simpler and cheaper to manufacture.

(ii) The instruction latency in a non-pipelined processor is slightly lower than in a pipelined equivalent. This is because extra flip-flopmust be added to the data path of a pipelined processor.

1. A non-pipelined processor will have a stable instruction bandwidth. The performance of a pipelined processor is much harder to predict and may vary more widely between different programs.

(i)Describe the assembly language development phases

Assembler

The assembler is used to convert the assembly language instruction into machine code.

It is used immediately after writing assembly language program.

The assembly starts by checking the syntax or validity of the structure of each of each instruction in the in the source file.

If any errors are found, the assembly display a report of these errors along with brief explanation of their nature.

However if the program does not contain any error, the assembler produces an object file that has the same name as the original.

Linking

The linker is used to convert the object file to an executable file.

The execution file is the final set of machine code instruction that can be directly executed by the microprocessor.

It is different than the object file in the sense that it is self-contained and re-locatable. An object file may represent one segment of a long program. This segment cannot operate by itself, and must be integrated with other object files representing the rest of the program, in order to produce the final self-contained executable file.

In addition to the executable file, the linker can also gene rate a special file called the

Map file. This file contains information about the start, end, and the length of the stack, code, and data segments. It also lists the entry point of the program.

Executing

The executable file contains the machine language code. It can be loaded in the RAM and be executed by the microprocessor

5. Explain the meaning of the following assembly code

(i)MOV AH, 01h

Input a character.

MOV AH, 01h INT 21h After the interrupt, AL contains the ASCII code of the input character. The character is echoed (displayed on the screen). Use function code 8 instead of 1 for no echo. Input a string .

SECTION .data

Buffer DB BUFSIZE ;BUFSIZE is max number of chars to read, <= 255

RESB BUFSIZE + 1

SECTION .text

MOV DX, Buffer

MOV AH, 0Ah

INT 21h

After the interrupt, BYTE [Buffer + 1] will contain the number of characters read, and the characters themselves will start at Buffer + 2 . The characters will be terminated by a carriage return (ASCII code 13), although this will not be included in the count. Output a character . MOV DL, ... -MOV AH, 02h

INT 21h

Load the desired character into DL, then call the interrupt with function code 2 in AH. Output a string .

MOV DX, ...MOV AH, 09h INT 21h

Load the address of a '$'-terminated string into DX, then call the interrupt with function code 9 in

AH. Exit .

MOV AL, ...

MOV AH, 4Ch

INT 21h

Load the return code (0 for normal exit, non-zero for error) into AL, then call the interrupt with function code 4Ch in AH. This is the proper DOS exit routine; however, if you are running your program with DEBUG, this will exit DEBUG. An alternative is to go behind DOS's back and use the BIOS routine accessed by INT 20h, which will return control to the DEBUG prompt when executed.

ii) Int 21h

Int21h

int 21h means, call the interrupt handler 0x21 which is the DOS Function dispatcher. the "mov ah,01h"

is seeting AH with 0x01, which is the Keyboard Input with Echo handler in the interrupt.

6.) Differentiate between the following types of interrupts

(i)Hardware and software interrupt

Hardware interrupts- These interrupts Hardware and software are generated by external devices (input/output devices) or occurrence of a hardware failure while software interrupts are interrupts are usually associated with the software instruction.

(ii)Vectored and Non Vectored interrupt

Vectored interrupts are interrupts that supply the CPU with information, the interrupt vector, which is used to generate the address of the handler routine for the interrupt while non-vectored interrupts are interrupts that transfer control directly to a single interrupt service routine, regardless of the interrupt source.

(iii) Maskable and Non Maskable interrupts

Maskable interrupts are Interrupt that can be disabled or ignored by the instructions of CPU are called as Maskable Interrupt while Non maskable are interrupt that cannot be disabled or ignored by the instructions of CPU.

7.Distinguish between:

i)Compiler and assembler program

Assembler is a program that converts source-code programs from assembly language into machine language while a complier is a program a complier s a computer program that reads a program written in one language, which is called the source language, and translates it in to another language, which is called the target language.

ii)Macro and procedure

is a Procedure is  is a collection of instructions to which we can direct the flow of a program while macro group of repetitive instructions in a program which are   
codified only once and can be used as many times as necessary.

iii)External and internal interrupts

External interrupts are interrupts that are generated by external devices while internal interrupts are interrupts that are generated internally to the processor, generally on the occurrence of an error condition.

8.Differentiate between the following instructions:

i)RCL and RCR

RCL is (Rotate through Carry Left), it rotates bits to the left, through the carry flag, and back into bit zero on the right. **RCR** (rotate through carry right),It shifts its bits right through the carry flag and back into the H.O. bit.

ii.ADD and ADC

ADD- The add instruction adds the contents of the source operand to the destination operand.

While ADC (add with carry) is similar to mov. Like mov, there are special

forms for the ax register that are more efficient

ii.ROL and ROR

ROR shifts bit zero into the H.O. bit

while ROL instruction rotates its operand to the left

the specified number of bits.

9.Using example explain the function of the following assembly directives

i)ORG-(Originate) Tells the assembler to set the location counter value

ii)PROC-(Procedure) is used to identify the start of a procedure.

iii)EQU-(Equate) is used to give a name to some value or to a symbol.

10) Explain the meaning of the following string commands in assembly language

i) SCAS

The SCAS instruction is used for searching a particular character or set of characters in a string. The data item to be searched should be in AL (for SCASB), AX (for SCASW) or EAX (forSCASD) registers. The string to be searched should be in memory and pointed by the ES:DI (or EDI) register.

ii) INS

ins

ins/outs : it means to Transfers a byte, word or double word of data from/to an I/O device into/out of the extra

/data segment + offset edi / esi ,respectively.The I/O address is stored in the edx register.

iii) LODS

The LODS Instruction

The LODS instruction is unique among the string instructions. You will probably never use a repeat prefix with this instruction. The LODS instruction copies the byte, word, or double word pointed at by ESI into the AL, AX, or EAX register, after which it increments or decrements the ESI register by one, two, or four. Repeating this

instruction via the repeat prefix would serve no purpose whatsoever since the accumulator register will be overwritten each time the LODS instruction repeats. At the end of the repeat operation, the accumulator will contain the last value read from memory. Instead, use the LODS instruction to fetch bytes (LODSB), words (LODSW), or double words (LODSD) from memory for further processing. By using the STOS instruction, you can synthesize powerful string operations. Like the STOS instruction, the LODS instructions take four forms:

lodsb();

lodsw();

lodsd();

rep.lodsb();

rep.lodsw();

rep.lodsd();

As mentioned earlier, you'll rarely, if ever, use the REP prefixes with these instructions 4. The 80x86 increments or decrements ESI by one, two, or four depending on the direction flag and whether you're using the LODSB, LODSW, or LODSD instruction.

(iv) Explain two separate units of the 8086 CPU

The 8086 CPU is organized as two separate processors;

(i)Bus Interface Unit (BIU)

The BIU provides H/W functions, including generation of the memory and I/O addresses for the transfer of data between the outside world -outside the CPU, that is- and the EU.

(ii)Execution Unit (EU).

The EU receives program instruction codes and data from the BIU, executes these instructions, and store the results in the general registers. By passing the data back to the BIU, data can also be stored in a memory location or written to an output device. Note that the EU has no connection to the system buses. It receives and outputs all its data thru the BIU.