Short Notes Chapter 1 [Lecture 1-4]

Address Bus:

The group of bits that the processor uses to inform the memory about which element to read or write is collectively known as the address bus.

The address bus is unidirectional and address always travels from processor to memory. This is because memory is a dumb device and cannot predict which element the processor at a particular instant of time needs.

Data Bus:

The data bus is used to move the data from the memory to the processor in a read operation and from the processor to the memory in a write operation.

Data moves from both, processor to memory and memory to processor, so the data bus is bidirectional.

Control Bus:

The group consists of miscellaneous independent lines used for control purposes. For example, one line of the bus is used to inform the memory about whether to do the read operation or the write operation. These lines are collectively known as the control bus.

Control bus is special and relatively complex, because different lines comprising it behave differently. Some take information from the processor to a peripheral and some take information from the peripheral to the processor.

Memory Cell:

A memory cell is an n-bit location to store data, normally 8=bit also called a byte. The number of bits in a cell is called the cell width. The two dimensions, cell width and number of cells, define the memory completely. For memory we define two dimensions. The first dimension defines how many parallel bits are there in a single memory cell.

In general, the memory cell cannot be wider than the width of the data bus.

Why is the control bus Bi-directional?

For communication both processor and memory must be synchronized, like a speaker speaks and the listener listens. Synchronization is very important, as a result of our orders either we got the desired cell or we came to know that the memory is locked for the moment. Such information cannot be transferred via the

address or the data bus. For such situations when peripherals want to talk to the processor when the processor wasn't expecting them to speak, special lines in the control bus are used. The control bus carries the intent of the processor that it wants to read or to write. The information in such signals is usually to indicate the incapability of the peripheral to do something for the moment. For these reasons the control bus is a bidirectional bus and can carry information from processor to memory as well as from memory to processor.

What are registers?

There are temporary storage places inside the processor called registers. They are used when we need more than one data element inside the processor at one time. Registers are like a scratch pad ram inside the processor and their operation is very much like normal memory cells. They have precise locations and remember what is placed inside them.

Accumulator

There is a central register in every processor called the accumulator. Traditionally all mathematical and logical operations are performed on the accumulator. The word size of a processor is defined by the width of its accumulator. A 32bit processor has an accumulator of 32 bits.

Index register: [or pointer/base register]

Index register is used in such a situation to hold the address of the current array location. Now the value in the index register cannot be treated as data, but it is the address of data. In general, whenever we need access to a memory location whose address is not known until runtime, we need an index register. Without this register we would have needed to explicitly code each iteration separately.

Thus, index registers do not hold data instead they are used to hold the address of data.

Flag register: [or program status word]

This is a special register in every architecture called the flags register or the program status word. Like the accumulator it is an 8, 16, or 32 bits register but unlike the accumulator it is meaningless as a unit, rather the individual bits carry different meanings. The bits of the accumulator work in parallel as a unit and each bit mean the same thing. The bits of the flags register work independently and individually, and combined its value is meaningless.

Program Counter: [or Instruction Pointer]

A special register exists in every processor called the program counter or the instruction pointer that ensures the ordering of instructions i.e. Instructions run one after another, first, second, third and so on. "The program counter holds the address of the next instruction to be executed." A number is placed in the

memory cell pointed to by this register and that number tells the processor which instruction to execute; for example, 0xEA, 255, or 152.

Op-code:

The op=code or operation code is the binary pattern that represents an instruction.

Or

Short for operational code, it is a number that determines the computer instruction to be executed.

Instruction mnemonics:

A word or acronym used in assembly language to represent a binary machine instruction op code. Different processors have different instruction set and therefore use a different set of mnemonics to represent them.

Assembler:

The dumb translator that will convert the instruction mnemonics back to the original opcodes is a program and is called the assembler.

Instruction Groups:

Data Movement Instructions:

These instructions are used to move data from one place to another. These places can be registers, memory, or even inside peripheral devices. Some examples are:
mov ax, bx
lad 1234

Arithmetic and Logic Instructions

Arithmetic instructions like addition, subtraction, multiplication, division and Logical instructions like logical AND, logical OR, logical XOR, or complement are part of this group.
Some examples are:
and ax, 1234
add bx, 0534
add bx, [1200]

Program Control Instructions:

These are instructions that control the program execution and flow by playing with the instruction pointer and altering its normal behavior to point to the next instruction. Some examples are:

```
cmp ax, 0
jne 1234
```

Special Instructions:

Another group called special instructions works like the special service commandos. They allow changing specific processor behaviors and are used to play with it. They are used rarely but are certainly used in any meaningful program. Some examples are:

```
cli
sti
```

Where cli clears the interrupt flag and sti sets it.

The iAPX88 Registers:

The iAPX88 architecture consists of 14 registers.

CS	SP
DS	BP
SS	SI
ES	DI
IP	AX (AH, AL)
FLAGS	BX (BH, BL)
	CX (CH, CL)
	DX (DH, DL)

General Registers [AX, BX, CX, and DX]

These are the general-purpose registers. X in their name stand for Extended mean 16-bit registers. All of them can be accessed as one 16-bit or two 8-bit registers. All general-purpose registers have high(H) Byte and low(L) Byte which can be separately accessed.

AX:

The A in AX stands for Accumulator. The two registers AH and AL are part of the big whole AX. Any change in AH or AL is reflected in AX as well. AX is a composite or extended register formed by gluing together the two parts AH and AL.

BX:

The B of BX stands for Base because of its role in memory addressing. BX is divided into two registers of 8-bit which are BH and BL.

CX:

The C of CX stands for Counter as there are certain instructions that work with an automatic count in the CX register. It also has two 8-bit registers as CH (High byte) and CL (Low Byte).

DX:

The D of DX stands for Destination as it acts as the destination in I/O operations. It has two parts DH and DL.

Index Registers [SI and DI]:

SI and DI stand for source index and destination index respectively. These are the index registers of the Intel architecture which hold address of data and used in memory access. Being an open and flexible architecture, Intel allows many mathematical and logical operations on these registers as well like the general registers. The source and destination are named because of their implied functionality as the source or the destination in a special class of instructions called the string instructions. However, their use is not at all restricted to string instructions. SI and DI are 16-bit and cannot be used as 8-bit register pairs like AX, BX, CX, and DX.

Instruction Pointer [IP]:

This is the special register containing the address of the next instruction to be executed. No mathematics or memory access can be done through this register. It is out of our direct control and is automatically used. Playing with it is dangerous and needs special care. Program control instructions change the IP register.

Stack Pointer [SP]:

It is a memory pointer and is used indirectly by a set of instructions.

Base Pointer [BP]:

It is also a memory pointer containing the address in a special area of memory called the stack.

Flags Registers:

The flags register is not meaningful as a unit rather it is bit wise significant and accordingly each bit is named separately.

C	**Carry**	When two 16-bit numbers are added the answer can be 17 bits long or when two 8-bit numbers are added the answer can be 9 bits long. This extra bit that won't fit in the target register is placed in the carry flag where it can be used and tested.
P	**Parity**	Parity is the number of "one" bit in a binary number. Parity is either odd or even. This information is normally used in communications to verify the integrity of data sent from the sender to the receiver.
A	**Auxiliary Carry**	A number in base 16 is called a hex number and can be represented by 4 bits. The collection of 4 bits is called a nibble. During addition or subtraction if a carry goes from one nibble to the next this flag is set. Carry flag is for the carry from the whole addition while auxiliary carry is the carry from the first nibble to the second.
Z	**Zero Flag**	The Zero flag is set if the last mathematical or logical instruction has produced a zero in its destination.

S	**Sign Flag**	A signed number is represented in its two's complement form in the computer. The most significant bit (MSB) of a negative number in this representation is 1 and for a positive number it is zero. The sign bit of the last mathematical or logical operation's destination is copied into the sign flag.
T	**Trap Flag**	The trap flag has a special role in debugging
I	**Interrupt Flag**	It tells whether the processor can be interrupted from outside or not. Sometimes the programmer doesn't want a particular task to be interrupted so the Interrupt flag can be zeroed for this time. The programmer rather than the processor sets this flag since the programmer knows when interruption is okay and when it is not. Interruption can be disabled or enabled by making this bit zero or one, respectively, using special instructions.
D	**Direction Flag**	Specifically related to string instructions, this flag tells whether the current operation has to be done from bottom to top of the block (D=0) or from top to bottom of the block (D=1).
O	**Overflow Flag**	The overflow flag is set during signed arithmetic, e.g. addition or subtraction, when the sign of the destination changes unexpectedly. The actual process sets the overflow flag whenever the carry into the MSB is different from the carry out of the MSB

Big Endian:

The order can be most significant, less significant, lesser significant, and least significant called the big-endian order used by Motorola.

Little Endian:

The order can be least significant, more significant, more significant, and most significant called the little-endian order and is used by Intel.

Segmented Memory Model:

8080 and 8085 could access a total memory of 64K using the 16 lines of their address bus. When designing iAPX88 the Intel designers wanted to remain compatible with 8080 and 8085 however 64K was too small to continue with, for their new processor. To get the best of both worlds they introduced the segmented memory model in 8088.

Function of Segmented memory model:

The segmented memory model allows multiple functional windows into the main memory, a code window, a data window etc. The processor sees code from the code window and data from the data window. The size of one window is restricted to 64K.

There are four segment registers named CS, DS, SS, ES:

CS holds the zero or the base of code.
DS holds the zero of data.
Or we can say CS tells how high code from the floor is
DS tells how high data from the floor is
SS tells how high the stack is.

One **extra segment ES** can be used if we need to access two distant areas of memory at the same time that both cannot be seen through the same window. ES also has special role in string instructions. ES is used as an extra data segment and cannot be used as an extra code or stack segment.

Address Wraparound:

In physical address calculation a carry if generated is dropped without being stored anywhere, for example BX=0100, DS=FFF0 and the access under consideration is [bx+0x0100]. The effective address will be 0200 and the physical address will be 100100. This is a 21-bit answer and cannot be sent on the address bus which is 20 bits wide. The carry is dropped and just like the segment wraparound our physical memory has wrapped around at its very top.

The segment, offset pair is called a logical address, while the 20bit address is a physical address which is the real thing. Logical addressing is a mechanism to access the physical memory