International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 6 Issue 6, September-October 2022 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Micro-Processors: Present Technology and Up-gradations Required

Vividh Bansal

Manager, Central Bank of India

ABSTRACT

In this paper we will deal with the current technology being used in microprocessor. We will analyse both the hardware and the software and their interfacing with each other and How to better them to increase speed and reduce size. Assembly language with its constituent syntax in NASM on Linux Operating System will be discussed and we will show how it can be used to be executed with minimum time gap. Assembler will be discussed and user-friendly high level language to be used as fast as machine language will be discussed with recommendations. We will be also comparing and contrasting various micro-processor, their architecture and speed with each other so that to highlight advantages of each over other and also to suggest issues and their respective solutions.

How to cite this paper: Vividh Bansal "Micro-Processors: Present Technology and Up-gradations Required" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-6 | Issue-6, October 2022, pp.1570-1577, URL: www.ijtsrd.com/papers/ijtsrd52123.pdf

IJTSRD52123

Detailed Analysis

Technology has risen by manifolds in recent decade. Everything has changed, changed by whiskers with every technological step up – which has now taken to enormity considering reach, access, ease of usage, availability, networking and most important the spread. 'Payment' has become simple and quick now days. In a matter of seconds you can transfer money electronically and thus 'Digital Banking' has now days gained prominence.

'Technology' especially when we see development of application through a website, it has been made precise and without entering into the technical know-how, it is possible to design these simple applications through interfacing of already developed websites with your business website or application in a way which fulfils the necessity of the organization, firm, or individual concerned. "Technology" has changed manifold so are the expectations. 'Door-Step Service' and 'Anywhere Banking' are the two new buzz words which have gained prominence in the last few years and now international brands have started to penetrate the domestic market and to be precise they have started to create impression through technology in the minds of the consumer. If it is in your mind it is with you within minutes. For example you need to eat something; you go to the application of the food outlet and order and make a payment and you receive the order within minutes.

Loading….

Whenever you start with an application, it starts to load the files on backhand which includes graphic as well as functionality driven files. Capacity to store these files and the speed to process it determines the efficiency attained through miniaturization as well as the material used. Silicon based wafers being used as a 'semiconductor material' in the electronic circuit with printed circuit board design and tested right now through the mock drill test. So the power as well as the material used determines speed of the application.

If any file fails to be loaded, the program malfunctions and correct functionality is not depicted. In most of the software it shows the error and asks to reload or restart the program. Sometimes it even asks to reboot the system. So, in a nutshell, a change in architecture can remove this anomaly of reloading as well as rebooting to remove the error that has crept in due to unloaded program files.

Suggestion: An integrated chip can be designed to work in synch with the main integrated chip and can

temporarily store the files to be loaded and can check through pointers, in case of malfunction, what went wrong and with which file it went wrong? It can do the required correction either by re-loading that temporary file or by running the necessary minimum steps to re-correct the mistake.

How can we minimize Loading time?

Most of the applications are built on C which after compilation runs very fast as their compiled version run at the same speed as assembly programs. Microprocessor is digital integrated circuit, clock driven; multipurpose that accepts binary as input and gives binary as output. So OS should take lesser memory and the size of the RAM should be high probably to let it hold the Operating System code on it and let other programs use the microprocessor to enable better running speed and better response. The main limiting factors of the microprocessor chip that in turn also determines its speed are

1. The number of interconnections can be made on the chip
2. The number of transistors that can be put on the chip
3. The amount of heat that can be dissipated by the device under normal operation

The minimum components of a microprocessor are arithmetic and logical unit (ALU), control unit (CU). ALU performs operations such as 'AND' or 'OR' or perform addition, subtraction and operations like 'AND' or 'OR' etc. Each operation of ALU sets one or more flags in the status register as zero, negative number, overflow or others indicating the result of the last operation. The control unit retrieves the instruction codes from the memory to carry out operations required by ALU. A single operation code might effects data-paths, register and other elements in the processor. With advanced integrated circuit technology now we are having 64-bit words. Instead of carrying out certain operations through software were substituted by the built up of the additional unit in the chip like floating point arithmetic unit to fasten floating point operations were done. Lastly by increasing CPU cache it allowed the chips to become more fast and fast.

The very fewer points that have not been done in detail and can be taken up for future can be

1. The material being used for the printed circuit board. Right now Silicon is being used but certain other elements can be used.
2. Heat dissipation and miniaturization can be two more points on which research can be done to attain better results.
3. Instead of binary to communicate, usage of other machine/assembly language based on the concept of electric and magnetic fields or to an extant usage of quark particles or other for the same.
4. Alternate processes to execute the code in a way that fulfils the basic requirements of a display with necessary optional tasks embedded in the supporting software kept in ROM or RAM as it is suitable.
5. Usage of microprocessor in a way which allows the running program to be differentiated into distinct parts each with a label stored in IC's memory in place of the whole program and at the time of execution minimum code is reloaded so the processor is able to check the correctness of the code as well as run at higher speed and takes minimum amount of time. A concept of parallel processing is established here.
6. Self-Creation in microprocessor: A material as a substitute, to the silicon, be used which is able to re-create and rejuvenate itself after incidents through Artificial Intelligence which helps in keeping the system up to-date. Certain materials through the robotic technology can be used which are activated when system malfunctions, the robotic technology parts can come out through their pre-defined sockets, do the needful task and then get back to their designated sockets. Such system can be used to repair and refurbish the circuit and it can also be used for large machines and on a large scale for machines. These way old machines can get replaced and also modernized to the latest technology.

Our microchip-technology stands on the semiconductor's which are being used on a property that the resistance of the chip falls and subsequently the current through the chip increases as the doping is increased. We can get a design where the current is amplified, constant current sources, constant voltage sources and speed can be varied with the applied voltage, doping etc.

Heat dissipation reduction technique: Concept of 'heat sink' comes out here where Air velocity, choice of material, protrusion design, and surface treatment are the important factors that define heat sink. With more and more surface area to be in contact with the cooling medium like air we have the die temperature of the integrated circuit. A heat sink is made up of aluminium or copper. Thermal adhesive or thermal paste improves the heat sink's performance by filling gap between heat spreader and heat sink. Each semiconductor is labelled as a thermal resistance and is defined as average temperature rise per unit increase of temperature. Aluminium and copper are two random alloys that are being continuously used as heat-sink. The thermal resistance of Aluminium is around 166-201 W/(m-k) whereas Copper has double

the thermal conductivity i.e. around 400 W/(m-k). Fin efficiency can be used to calculate the thermal efficiency of the device which is equal to a fin which is a flat plate with heat flowing in one end and being dissipated from the other end as onto the surrounding medium or fluid. The dissipation will be maximum onto the base and will gradually decrease till the front end.

Fin efficiency is defined as actual heat transferred and the heat transferred considering the plate to be isothermal

$\eth_f = (\tanh(mL_c))/mL_c$

$mL_c=(2h_f/Kt_f)^{1/2}L_f$

where h_f is the convection coefficient and its value in air is about 10-100 in air and in water it is about 500-10000. K is thermal conductivity of material of Fin. L_f is the Fin height (in meter) and t_f is the Fin thickness (in meters).

Three types of Fin are widely used which are listed as below:

1. Pin-Fin: There are pins which can be elliptical, square or cylindrical.
2. Straight-Pin: Straight sheets perpendicular to the base is drawn
3. Flared-Pin: When the sheets are drawn which are not parallel to the base we end up in flared pin.

We have to see that heat dissipation takes by three ways which are conduction, convection and radiation. Since we are dealing with low temperatures of the order of 0 to 100 degree Celsius temperature the factor of radiation is negligible and thus neglected. In case of material with no fin shaped plate being used with little or no flow the emission by radiation increases. Here we use the fact that the absorption and reflection are both the features of surface dependent on the frequency related parameters where the black surface is the best absorber and shiny surface being best reflector.

Programming: Turning assembly language into machine language is the function of the assembler and transforming machine language into assembly language is the function of disassemble. Usage of pseudo-instructions and macro's in order to accommodate those instructions which are not present directly in the assembly language is used for designing complex code and sequences. Since in the assembler environment we have the instruction set defining pseudo-instructions and the macro languages we can transmit the full code into object code in the machine code environment but the reverse is not true where we can disassemble the object code only into the assembly language codes which were fully transformed into machine instructions. In nutshell we lose pseudo-instructions and macro instructions when we do the reverse to transform machine code to the assembly language code.

Basic elements of Language design:

1. Op-code mnemonics: A mnemonic is a name of a single executable machine language instruction (an op-code), and there is at least one op-code defined for each machine language instruction. Each instruction has an op code plus zero or more operands. Operands can be immediate; data stored in registers specified or implied, address of data stored in the memory or elsewhere. Extended mnemonics are often used with specific op-code with a specific operand.
2. Data definitions: These are the set of instructions which are used to define data elements to hold data and variables. They define the type of data, the length and the alignment of data. They also determine whether the data is available to outside or not, also known as pseudo-ops.
3. Assembly directives: Assembly directives are also called pseudo-op-codes, pseudo-operations, or pseudo-ops are commands given to the assembler to perform instructions other than assembling instructions. Directives decide how assembler operates and affects the object code, the listing file, symbol table and the values of internal assembler parameters. Sometimes pseudo-op-codes is reserved for directives that generate object code, such as those that generate data.

The name of pseudo-ops starts with a dot to distinguish them from machine operations. The usage of pseudo-ops is following:

A. It helps programmer to assemble the same program differently for different applications by making the assembly of program dependent on the parameters entered by the programmer.
B. It helps in maintaining presentation of program better and helps it to maintain it easier to read and maintain.
C. Reserve storage areas for run-time data and optionally initialize their content to known data.

Assemblers such as NASM provides flexible symbol management allowing management of different namespaces, calculating offsets within data structures, assigning labels referring to literal values. Labels are used to assign constants and variables with values in re-locatable addresses.

1. Macros: They can be pre-defined or programmer defined macros. Macro is defined as a mixture of assembler statements example: assembler statements, directives and templates for assembler

instructions. Macro's once defined name can be used in place of mnemonic.

Example: In C language #define parameters are one of the macros used in the assembly language. Although assembly language is not used by the programmers now but it is used still for access for specialized processor instructions, addressing critical performance tests and direct hardware manipulation. Assembly languages are used in comparison to high level languages for optimization of speeds and size.

We will refer to the fetching data from memory as execution cycle which has following three steps to undertake which are:

1. Fetching the instruction from the memory
2. Decoding or identifying the instruction
3. Executing the instruction

Processor stores data or instruction in the form of byte where lower order byte is stored in lower address and higher order byte is stored. An example can be a hexadecimal number AABB where when the data is transferred from program to memory lower order byte occupies lower memory space i.e. in this case it is BB and the higher order byte i.e. AA occupies higher memory space.

Two types of memory addresses are there

1. Absolute address
2. Segment or offset address

Dependence of assembly language is basically on two things

1. Instruction set
2. Underlying architecture of the processor

There are many assembler programs in the market among which we have:

1. NASM
2. MASM
3. TASM (Borland Turbo assembler)
4. GNU assembler (GAS)

Components of an Assembly program are as follows:

1. Data section: For initializing or declaring variables. Syntax is section.data.
2. Bss section: For declaring variables i.e. Syntax is section.bss
3. Text section: For keeping the actual code. This section begins with the keyword global_data under the heading section.text and tells where the code lies.

Basic syntax of the text section:

```
Section.text
        Global_start
_start:
```

4. Comments: It begins with Semi-colon (;)

Syntax of Assembly language program is as follows:

[Label] mnemonic [operands] [;comments]

The fields in the square brackets are optional. Example: INC COUNT;

ADD AH, BH

Example program can be written as following:-

```
Section.text
        global_start ;must be declared for linker;
_start: ;tells linker entry point
        Mov edx, len ;message length
        Mov ecx, msg ;message to write
        Mov ebx, 1 ;file descriptor (stdout)
        Mov eax, 4 ;system call number (sys_write)
        Int 0x80 ;call kernel
        Mov eax, 1 ;system call number (sys_exit)
        Int 0x80 ;call kernel
Section.data
        Msg db 'Hello World!', 0xa ;string to be printed
        Len equ $ -msg ;length of the string
```

Steps to be done with above file:

Save file as hello.asm,

Assemble file: nasm –f elf hello.asm

Object file hello.o will be created

For creation of executable file by linking object file: ld –m elf_386 –s –o hello hello.o

Execute the program by typing: ./hello

Registers are the internal data segments of memory on the microprocessor. There are 10 32-bit Register and 6 16-bit Register. There are three types of register present:

1. General Registers: They are further divided into the following three groups:

A. Data register: 4 32-bit Registers are used for ALU. These 32-bit Registers are used in three ways:
   a) As full 32-bit Register. These registers are EAX, EBX, ECX, EDX
   b) Lower halves can be used as 16-bit Registers which are AX, BX, CX and DX
   c) Lower and higher halves of the above registers (16-bit) can be used as eight 8-bit registers which are as follows:- AH, AL, BH, BL, CH, CL, DH, DL.

Specific uses of these 4 Data register namely EAX, EBX, ECX and EDX

a) AX is the primary accumulator that ia all the inpot/output as well as arithmetic operations are done by it.
b) BX is the Base register
c) CX is the Count register
d) DX is known as the Data register

B. Pointer register: Pointer registers are 32-bit EIP, ESP and EBP Registers and corresponding 16-bit right portions IP, SP and BP. There are three categories of pointer register

a) Instruction Pointer (IP): 16-bit IP register has the offset address of the next instruction to be executed. Together with the CS register CS:IP gives complete address of the current instruction in the code segment.

b) Base Pointer (BP): 16-bit BP register is used to referencing the parameter variables passed to the program in a sub-routine. Address in SS register is combined with offset in the BP register to locate the parameter.

c) Stack Pointer (SP): 16-bit SP register provides the offset value within the program. SS:SP refers to the current position of data or address in the program stack.

C. Index register: Index registers which are 32-bit registers namely ESI and EDI. SI and DI are used for indexed addressing and sometimes for addition and subtraction.

a) Source Index (SI): Source index for the string operations
b) Destination Index (DI): Used as a destination index for string operations

2. Control Registers: 32-bit instruction pointer register and the 32-bits flag register combined are called control registers. The major uses are comparison, conditional sentences and mathematical calculations. The common control registers are following:

a) Overflow Flag (OF): It indicates the overflow of higher order bit after a signed mathematical operation.
b) Direction Flag (DF): Comparison of string data from left or right. When it is one direction is from right to left and when it is 0 directions is from left to right.
c) Interrupt Flag (IF): External interrupts to be ignored or accessed. It disables external interrupt when set to zero and enables it when set to one.
d) Trap Flag (TF): It allows setting the flow of operation with single step at a time. For example DEBUG program sets the trap flag 1 so it can take through the whole program.
e) Sign Flag (SF): It results in the sign of the arithmetic operation. It is set according to the sign of the data item following the arithmetic operation. It is indicated by the higher order of the leftmost bit. A positive result sets SF to zero and negative to one.
f) Zero Flag (ZF): It shows the result of arithmetic or comparison operation. A non-zero result sets it to 0 and zero result sets it to 1.
g) Auxiliary Carry Flag (AF): It is set during arithmetic operation. It is set when 1-byte operation sets carry from bit-3 to bit-4.
h) Parity Flag (PF): It indicates the total number of 1-bits in an arithmetic operation. An even number of 1-bits set parity flag to zero and an odd number of 1-bits set parity flag to 1.
i) Carry Flag (CF): It contains the carry of 0 or 1 from a higher order bit (leftmost) after an arithmetic operation. It also stores the content of last bit or shift or rotate operation.

3. Segment Register: Specific areas in a program containing data, code and data.

a) Code Segment: All the instructions to be executed are stored here. CS register store the address of the beginning of the program.
b) Data Segment: Contains data, constants and work areas. 16-bit DS register stores the address of the beginning of the data segment.
c) Stack Segment: Data and return addresses of the subroutines and the procedures. SS register stores the starting address of the stack.

Design of an assembler:

Seeing how machine language operates as above in an assembly language which is friendly with the machine language and is almost similar to it with certain names given to the variables and the storage spaces with the execution of the statement one by one of each instruction like in machine language. We can, even in the case of high level language, make available the compiled version of machine executable instructions available to the end-user which can be referenced in the program through simple names or through conditional sentences. Like in JAVA, JVM compiles the data into the executable code through the compiler and is available but it has to be run separately or as an executable file like a JAR file. So in specific any one of the JAVA or C code when compiled should be made platform as well as machine independent. The following step will help in the compilation as well as linking it to the other machines especially server. Server should be created and with basic programs to be compiled and run on start,

certain other programs should also be allowed to run on request or approval of the user. They should be stored in memory other than ROM which is lesser volatile than RAM. On user approval it should be allowed to run on the system concerned as a system file which will help to keep the compiled code to be available for future necessary actions.

Design of an assembler is a must based on the Artificial Intelligence (AI) it should allow the program to run and compile certain fixed designated system files with an option to add to its repository more files at the discretion of the user concerned.

Interfacing of the Mother board:

Comparison of Micro-Processor:

NAME	YEAR	TRANSISTORS	DATA WIDTH	CLOCK SPEED
8080	1974	6,000	8 bits	2 MHz
8085	1976	6,500	8 bits	5 MHz
8086	1978	29,000	16 bits	5 MHz
8088	1979	29,000	8 bits	5 MHz
80286	1982	134,000	16 bits	6 MHz
80386	1985	275,000	32 bits	16 MHz
80486	1989	1,200,000	32 bits	25 MHz
PENTIUM	1993	3,100,000	32/64 bits	60 MHz
PENTIUM II	1997	7,500,000	64 bits	233 MHz
PENTIUM III	1999	9,500,000	64 bits	450 MHz
PENTIUM IV	2000	42,000,000	64 bits	1.5 GHz

Evolution of Microprocessor

('Evolution of Microprocessor', 2022)

We will be comparing in particular modern day 64-bit architecture microprocessors which are still in use and is not obsolete. 64-bit architecture refers to, in case of software used, machine code with 64-bit virtual memory addresses.64-bits is the word size that refers to computer architecture, buses, memory, CPU's and the software that runs on it.

As we will see that 32-bit memory roughly translates to 4 GB Memory i.e. 2^{32} instruction space i.e. roughly 4 GB memory. Whereas when we are using 64-bit processor on a 64-bit operating system on 64-bit addresses we have roughly about 2^{64} instruction space which translates to about 16 GB of free space.

In the table below we have classified the various processors of the Intel which are still in use on heat dissipation, bus speed, cache memory available with fabrication of the transistors on it. As we go on to the latest version we see the complexity to increase regarding fabrication and speed and also special features such as hyper threading's presence which increases in the complexity with the latest model.

Processor	Production date	Thermal Design Power (TDP)	Fabrication	Important Miscellaneous Point
Intel Pentium	1993-1999	Unknown	800 nm-350nm	Clock Rate (65 MHz-250 MHz) Bus Speed (50 MHz-66MHz)
Intel Atom	2008-09 (as Centrino Atom) 2008- Present (As atom)	0.65 W-13 W	32 nm, 45 nm	Bus Speed (400 MHz-2.5 GTS^{-1})
Intel Celeron	1998- Present	4 W- 86 W	7 nm- 250 nm	Bus Speed (266 MHz-3.6 GHz) Cache1-8kiB-64KiB per core Cache2-0 KiB-1MiB Cache3-0 KiB- 2 MiB
Intel Xeon	1998-Present	16 W- 165 W	7 nm- 250 nm	Bus Speed (100 MHz-6.4 GTs^{-1}) Cache1-8kiB-64KiB per core Cache2-256 KiB-12 MiB Cache3-4 KiB- 16 MiB
Intel Pentium	2009-Present	2.9 W-73 W	7 nm – 65 nm	Bus Speed (800 MHz-5 GTs^{-1}) Cache1-64KiB per core Cache2-2*256 KiB-2 MiB Cache3-0 KiB- 3 MiB Hyper-threading 2/w
Intel Core-i3	2010-Present	35 W- 91W	Intel 7nm-32 nm	Bus Speed (100 MHz-6.4 GTs^{-1}) Cache1-64KiB per core Cache2-256 KiB Cache3-3 MiB- 4 MiB Hyperthreading 2/w, 4.4/w

Intel Core – i5	2009- Present	17 W-125 W	Intel 7 nm- 45 nm	Bus Speed (2.5-6.4 GTs^{-1}) Cache1-64-80 KiB per core Cache2-256- 512 KiB Cache3-4 MiB- 12 MiB Hyperthreading 4-6-8/w
Intel Core- i7	2008-Present	35 W – 130 W	Intel 7 nm- 45 nm	Bus Speed (4.8-6.4 GTs^{-1}) Cache1-64-80 KiB per core Cache2-256- 512 KiB Cache3-6 MiB- 16 MiB Hyperthreading 4-6-8/w
Intel Core- i7 (Extreme edition)	2011-Present	130 W-150 W	14 nm-32 nm	Bus Speed (2.5-8 GTs^{-1}) Cache1-64 KiB per core Cache2-256 KiB per core Cache3-12 MiB- 20 MiB Hyperthreading 4-6-8 or 10/w
Intel Core i9	2018 - Present	35 W – 125 W	Intel 7 nm-14 nm	Bus Speed (2.5-6.4 GTs^{-1}) Cache1-64-80 KiB per core Cache2-256- 512 KiB Cache3-16 MiB Hyperthreading 6-8-10/w
Intel Core i9 (Extreme Edition)	Q3 2017 - Present	35 W – 165 W	Intel 14 nm	Bus Speed (8 GTs^{-1}) Cache1-64 KiB per core Cache2-1 MiB Cache3-13.75 MiB- 24.75 MiB Hyperthreading 8-18/w

Recommendations to increase speed and the subsequent memory allocation are as follows:

1. Fabrication can be reduced to minimum right now in the latest design it is of the order of nanometres, with careful selection of material of PCB, heat sink and fabrication process used it can be still brought down to minimum thereby increasing speed and reducing size.
2. Increase the size of cache memory which has increased with the use of 64-bit address register with 64-bit bus size and with 64-bit machine code's virtual memory design
3. Generation of a memory between ROM and RAM which is based on the user's programmability. If user wants he can store certain reusable machine language code on it which can be run with system files as and when the system loads at the distinction of User.
4. HLL (High level language) can be used to program simple programs which can be converted into machine language simultaneously as and when user is giving input thereby reducing time and also space to run the code.
5. Assembler as highlighted above should be designed in a way as it converts HLL or Assembly language program into the machine code it allows usage of abbreviations and subroutines with conditional looping so that given code compiles fast and also reuse of such code is easily done.
6. Our processor is digital right now. It accepts input in the form of 0 and 1's. We can shift towards Analog framework by introducing technical modifications so that even a 1 can be differentiated into several demarcated different levels by the Analog to digital convertor in the processor itself. So that when we are accepting a input it can go as more lower bits and thus processing can be made faster.
7. Heat dissipation is also an issue and most of today's processor loose or deviate from normalcy once they are heated up. So proper heat dispenser as discussed should be used to minimize heat dissipation which is still comparatively high as others.
8. An issue exists that till now, as we see in the table of comparison of microprocessors we have not seen any significant upgrade since 2011. We have transgressed from Intel Core i3 to Intel Core i9 – there has been no significant change in

A. Thermal Power Dissipation in fact it has increased

B. Fabrication too has not changed much. The number of transistors per unit area remains the

same as for now. It seems miniaturization with current technology has reached its stagnation points.

C. Material being used for the circuit design too is same.

D. Only change is in the Bus speed and increase of cache size which too in the later models is showing stagnation.

E. We have to look over the basic architecture of a microprocessor including ROM, RAM, as suggested an introduction to a newer memory segment that has characteristics of both ROM and RAM operating under the user's discretion. This will definitely increase the memory size and speed.

F. A proper user-friendly User-Interface that has directions and functions which allows user to leverage best from their system according to their usage specifications.

Reference:

[1] 'Evolution of Microprocessor', viewed on 5th October 2022, <https://www.elprocus.com/evolution-of-microprocessor-with-applications/>

[2] Streetman, Ben G., Banerjee, Sanjay, "Solid State Electronic Devices", 5th Edition, Prentice-Hall of India Private Limited

[3] Leach, Donald P., Malvino, Albert Paul, "Digital Principles and Applications", 5th Edition, Tata McGraw-Hill Publishing Company Limited

[4] Raisinghania, M.D., "Ordinary and Partial Differential Equations", 14th Edition, S. Chand & Company Limited

[5] Haykin, Simon, "Digital Communications", John Wiley and Sons

[6] Narayan, Shanti, Mittal, P.K., "Integral Calculus", S. Chand

[7] Kapila, Uma 2011, *Indian Economy: Performance and Policies,* 10th Edition Academic Foundation

[8] Robbins, Stephen P., Judge, Timothy A., Sanghi, Seema 2009, *Organizational Behavior,* 13th Edition, Pearson Prentice Hall

[9] Cooper, Donald R, Schindler, Pamela S 2009, *Business Research Method,* 9th Edition Tata McGraw Hill

[10] Ohmae, Kenichi 2013, *The Mind of the Strategist: The Art of Japanese Business,* 16th Edition Tata McGraw Hill