

B.Tech.

Fourth Semester Examination, May-2011

Computer Architecture and Organizations (CSE-210-F)

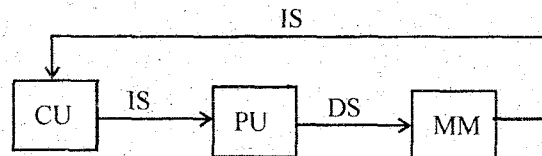
Note : First Question is compulsory. Attempt five questions in all selecting at least one question from each unit. All questions carry equal marks.

Q. 1. Write short notes on following :

- | | |
|-----------------------|----------------------|
| (a) Flip Flop | (b) SISD |
| (c) MSAM | (d) RAM |
| (e) ROM | (f) Interrupt |
| (g) Micro instruction | (h) Instruction Set. |

Ans. (a) Flip Flop : In electronics, a flip-flop or latch is a circuit that has two stable states and can be used to store state information. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs flip-flops and latches are a fundamental building block of digital electronics systems used in computers, communication and many other types of systems. Flip-flops are used as data storage elements, such data storage can be used for storage of state and such a circuit is described as sequential logic.

(b) SISD [Single Instruction Stream-Single Data Stream] : This organization, shows in figure (a), represents most serial computers available today. Most SISD uniprocessor systems are pipelined. An SISD computer may have more than one functional unit in it. In this, all the functional units are work under the supervision of one control unit and instructions are executed sequentially but may be overlapped in their execution stages [Pipelining].



CU = Control Unit

PU = Processor Unit

MM = Memory Module

IS = Instruction Stream

DS = Data Stream

(c) MSAM (Micro SCSI Architectural Model) : SCSI architectural model provide an abstract view of the way the SCSI devices communicates. It is intended to show how the different SCSI standards are inter-related. The main concepts and terminology of the SCSI architectural model are :

- Only the externally observable behaviour is defined in SCSI standards.
- The relationship between SCSI devices is described by a client server, service delivery model. This client is called a SCSI initiator and the server is called a SCSI target.

- (iii) A SCSI domain -consist of a least one. SCSI device, at least one SCSI target and at least one SCSI initiator interconnected by service delivery subsystem.

(d) RAM : Random Access Memory is a form of computer data storage. Today, it takes the form of integrated circuit that allow stored data to be accessed in any order with a worst case performance of constant time. Strictly speaking, modern types of DRAM are therefore not random access, as data is read in bursts, although the name DRAM/RAM has stuck. However, many types of SRAM, ROM, OTP and NOR flash are still random access even in a strict sense.

RAM is often associated with volatile types of memory [such as DRAM memory Modules] where its stored information is lost if the power is removed. Many other types of non-volatile memory are RAM as well, including most types of ROM and a type of flash memory called NOR-Flash.

(e) ROM : Read Only Memory is a class of storage medium used in computers and other electronic devices. Data stored in ROM cannot be modified or can be modified only slowly or with difficulty, so, it is mainly used to distribute firmware.

[Software that is very closely tied to specific hardware and unlikely to need frequent updates].

In its strictest sense, ROM refer only to Mask ROM, which is fabricated with the desired data permanently stored in it and thus can never be modified. Despite the simplicity speed and economies of scales of mask ROM, field-programmability often make reprogrammable memories more flexible and inexpensive.

(f) Interrupt : In computing, an interrupt is an asynchronous signal indicating the need for attention or synchronous events in software indicating the need for a change in execution.

A hardware interrupt causes the processor to save its state of execution and begin execution of an interrupt handler similar to a hardware interrupt.

Interrupts are a commonly used technique for computer multitasking especially in real time computing. Such a system is said to be interrupt driven.

Interrupts can be categorized into. Maskable interrupt, non-maskable interrupt [NMI], Inter Processor Interrupt [IPI] . software interrupt and spurious interrupt.

(g) Micro Instruction : A microinstruction is a simple command that makes the hardware operate property. The format is unique to each computer, but over example has a 24 bit micro instruction. Our instruction is broken down into nine parts :

(i) Bits (0–6) : This is used to specify a point in the code to jump to ne.... Whether or not the jump is made depends upon the result of the ALU operation.

(ii) Bit (7 and 8) : This is used to determines whether a jump in the code should be made or if simply the next instruction is to be executed. It can have the following values :

00 : Never of jump

01 : Jump if the ALU's N bits is 1

1 0 : Jump if the ALU's Z bits is 1.

1 1 : Always jump.

Bits 9–11 : It is used to determine what register will be, the first input to the ALU. It is simply a number from zero to seven.

Bits 12–14 : B This part of the micro-instruction is used to determine what register will be the second input to the ALU. If the second input is not used, this field's value does not matter.

Bits 15–17 : C This part of micro instruction is used to determine where the output from the ALU is stored and again is just the number of a register.

Bit 18 : It is used to determine whether or not the ALU's output is stored in the register. Specified by the C field.

Bit 19 : It is used to determine whether or not the contents of the MBR register are written to RAM.

Bit 20 : It is used to determine if the MBR register is filled with a word from Ram.

Bit 21–23 : ALU this part of the microinstruction is used to determine what operation the ALU performs on its inputs.

(h) Instruction Set : An instruction set is the part of the computer architecture related to programming, including the native data types, instructions, register, addressing modes, memory architecture internet and exception handling and external input/output. Instruction set architecture include a specification of the set of opcodes [Machine language] and the native commands, implemented by a particular processor.

Instruction Type :

(i) Set : A register

(ii) Move : Data from a memory location to a register.

(iii) Read and Write : Data from hardware devices.

(iv) Compare : Two value in register.

(v) Add, Subtract, Multiply or Divide : The values of two register.

Unit-I

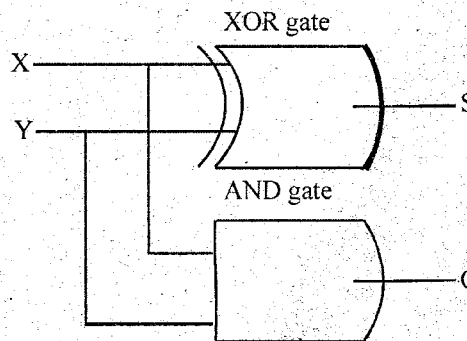
Q. 2. Explain half adder and full adder circuit in detail.

Ans. Half Adder : The half adder is an example of a simple, functional digital circuit built from two logic gates. The half adder adds to one-bit binary numbers [AB]. The output is the sum of the two bits [S] and carry [C].

Not how the same two inputs are directed to two different gates. The input to the XOR gate are also the inputs to the AND gate. The input "wires" to the XOR gate are tied to the input wires of the AND gate : thus when voltage is applied to the A input of the XOR gate, the A input to the AND gate receives the same voltage.

Select an input combination from the pull-down selector and view the resulting output.

Half Adder :



$$S = X \oplus Y = X'Y + XY'$$

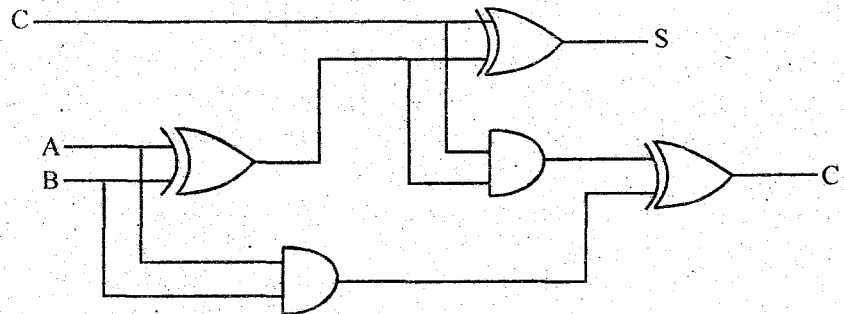
$$C = XY = XY$$

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Full Adder : The full adder circuit adds three one bit binary numbers (CAB) and outputs two one-bit binary numbers a sum (S) and a carry (C_1). The full adder is usually a component in a cascade of adders, which add 8, 16, 32 etc, binary numbers. The carry input for the full adder circuit is from the carry output from the circuit "above" itself in the cascade. The carry output from the full adder is fed to another full adder "below" itself in the cascade.

If you look closely, you will see the full adder is simply two half adders joined by an OR select an input combination from the pull-down selector and view the resulting output.

Full Address Circuit :



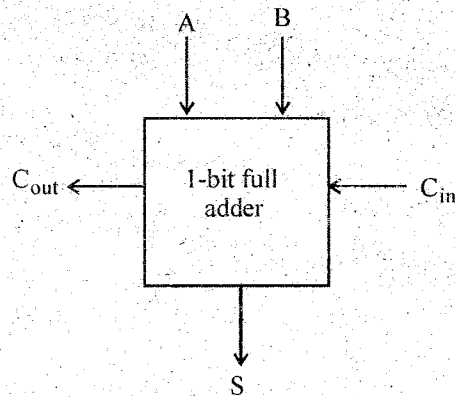
Truth table

Input			Output	
C	A	B	S	C
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full adder takes three one-bit numbers, often written as A, B and C_{in} ; A and B are operands and C_{in} is a bit carried in. The circuit produces a two-bit output sum typically represented by the signals C_{out} and S, Where $Sum = 2 \times C_{out} + S$. The one-bit full adders truth table.

A full adder can be implemented in many different ways such as with a custom transistor-level circuit or composed of other gates. One example, implementation as with $S = A \oplus B \oplus C_{in}$

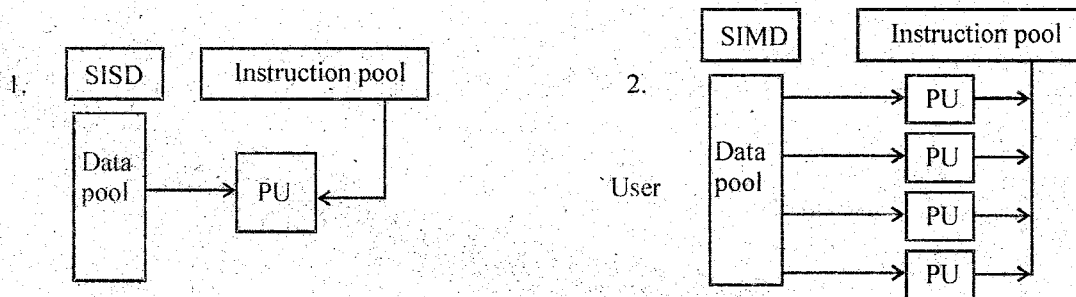
$$C_{out} = (A.B) + (C_{in} \cdot (A \oplus B))$$



Q. 3. (a) Explain Flynn's classifications of computers in detail.

Ans. There are four classifications defined by flynn are based upon the no. of concurrent instruction and data streams available in the architecture :

(i) Single Instruction Over Single Data Stream (SISD) : A sequential computer which exploits no parallelism in either the instruction or data streams. Single Control Unit (CU) fetches single instruction (IS) from memory. The CU then generates appropriate control signals to direct single processing element (PE) to operate on single Data Stream (DS) i.e., one operation at a time. Example : SISD architecture are traditional uniprocessor machines like a PC or old Mainframes.

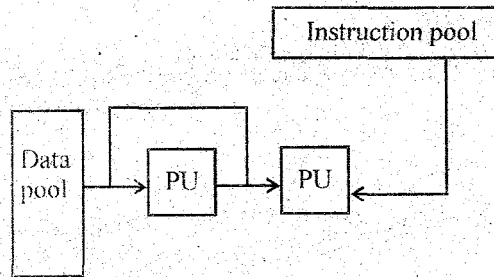


(ii) SIMD : A computer which exploit multiple data streams against a single instruction stream to perform operations which may be naturally parallelized for example, an array processor or CPU.

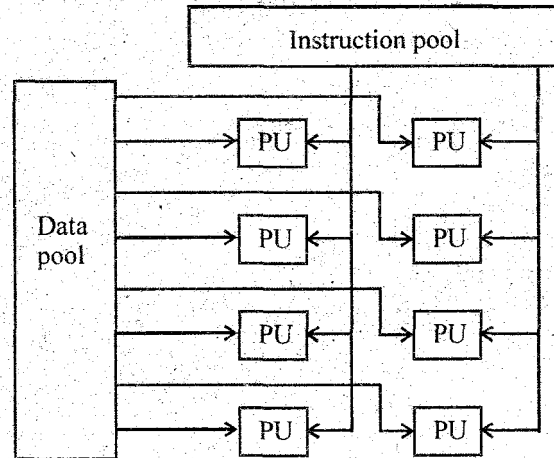
(iii) Multiple Instruction, Single Data Stream [MISD] : Multiple instruction operate on a single data stream. Uncommon architecture which is generally used for fault tolerance. Heterogenous systems Operate on

the same data stream and must agree on the result. Example : Include the space shuttle flight control computer.

3. MISD :



4. MIMD :



(iv) **MIMD** : Multiple autonomous processor simultaneously executing different instructions on different data. Distributed systems are generally recognized to be MIMD architectures either exploiting a single shared memory space or a distributed memory space.

Q. 3. (b) Explain why operating system is essential for computers?

Ans. An operating system is software consisting of programs and data that run on computers, manages. Computer hardware resources and provides common services for execution of various application software.

The operating system is the most important type of system software in a computer system. Without an operating system, a user cannot run an application program on their computer, unless the application program is self booting.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between application programs and the computer hardware. Although the application code is usually executed directly by the hardware and will frequently cost the OS or the interrupted by it.

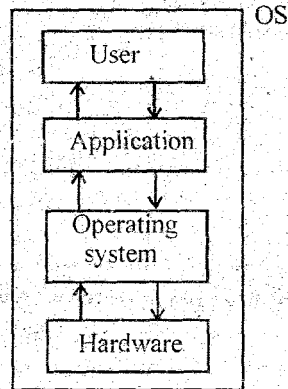
Operating systems are found on almost any devices that contains a computer from cellular phones and video consoles to supercomputers and web servers. Example of modern operating systems; BSD, Linux [Ubuntu, fedora, OpenSUSE, Mandriva, Arch Linux, Debian, Linux Mint], MacOS X, Microsoft Windows and Unix.

Operating Systems :

Common Features :

- (i) Process management
- (ii) Interrupts
- (iii) Memory management
- (iv) File system
- (v) Networking (TCP/IP, UDP)
- (vi) Device Driven

- (vii) Security (process/Memory)
- (viii) Input/Output protection.



Unit-II

Q. 4. (a) Differentiate RISC and CISC classification?

Ans.

CISC	RISC
(i) Means Complex instruction set computer.	(i) Means reduced instruction set computer.
(ii) A CISC system has complex instructions such as direct addition between data in two memory location. e.g., 8085.	(ii) A RISC system has reduced number of instructions and more importantly it is load store architecture where pipelining can be implemented easily. e.g., ARMELAR.
(iii) Instruction set : large set of instruction with variable size (16 to 24).	(iii) Instruction set : Small set of instruction with fixed size [32-bit].
(iv) Addressing Modes (12-24).	(iv) Addressing modes : [3-5]
(v) General purpose registers (8 to 24)	(v) General purpose register [32-192]
(vi) Clock rate [33 to 50 MHz in 1992]	(vi) Clock rate (33 to 50 MHz) in 1992.
Advantage of RISC :	Advantage of CISC :
(i) Faster [APPL]	(i) Less expensive due to the use of microcode no need to hardware a control unit [APPL].
(ii) Simpler hardware	(ii) Fewer instruction could be used to implement a given task, allowing for more efficient use of memory [APPL].
(iii) Shorter design cycle due to simplex hardware (CHAR)	(iii) More instruction can fit into the cache since the instructions are not a fixed size [TURL].
(iv) Lower cost since more part can be placed on a single chip [APPL].	

Disadvantage of RISC :

- (i) Programmer must pay close attention scheduling so that the processor does not spend a large amount of time waiting for an instruction to execute [APPL].
- (ii) Debugging can be difficult due to the instruction scheduling.
- (iii) Require very fast memory systems to feed them instruction.

Disadvantage (CISC) :

- (i) Instruction set and chip hardware became more complex with each generation of computers.
- (ii) Different instructions take different amount of time to execute due to their variable length.
- (iii) Many instructions are not used frequently approx 20% of the available instructions are used in a typical program.

Q. 4. (b) Define addressing mode? Define types of addressing modes with suitable examples.

Ans. Addressing Mode : Addressing mode are an aspect of the instruction set architecture in most Central Processing Unit [CPU] designs. The various addressing modes that are defined in a given instruction set architecture define how machine language instruction in that architecture identify the operand [or operands] of each instruction. An addressing mode specifies how to calculate the effective memory address of an operand by using information held in registers and/or constants contained within a machine instruction or else where.

In computer programming addressing modes are primarily of interest to compiler writers and to those who U code directly in assembly language.

Type of Addressing Mode : There are many type of addressing mode :

(i) Immediate Addressing Mode : This is the simplest form of addressing. Here, the operand is given in the instruction itself. This mode is used to define a constant or set initial values of variables. The advantage of this mode is that no memory reference other than instruction fetch is required to obtain operand. The disadvantage is that the size of the number is limited to the size of the address field, which most instruction sets is small compared to word length.

(ii) Direct Addressing : In direct addressing mode, effective address of the operand is given in the address field to the instruction. It requires one memory reference to read the operand from the given location and provides only a limited address space, length of the address field is usually less than the word length.

Example : Move F, R0, Add Q, ROP and Q are the address of operand.

(iii) Indirect addressing

(iv) Register addressing

(v) Register indirect addressing

(vi) Displacement addressing

(vii) Stack addressing.

Stack Addressing : Stack is a linear array of locations referred to as last in first out queue. The stack is a reserved block of location, appended or deleted only at the top of the stack. Stack pointer is a register which stores the address of top of stack location. This mode of addressing is also known as implicit addressing.

Q. 5. Explain microinstruction format also explain how the field of micro operation is decoded?

Ans. Micro instruction is discussed in the first question of (g) part here 9 discuss micro instruction format and decode.

Micro Instruction Format is :

Micro Instruction Format :

	Dest	ALU	SrC ₁	SrC ₂	Constant	MISC	Cond.	Jamp addr.
00000	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE
100001	C	ADD	A	4	MDR \leftarrow M[MAR]	Unicode		
20010	PC	SUB	B		N[MAR] \leftarrow MDR	Zero		
30011	MAR	RR	PC		A \leftarrow RS ₁	Negative?		
40100	MDR	RL	MAR		B \leftarrow RS ₂	Carry?		
50101	IR	SR	MDR		B \leftarrow R _d	overflow?		
60110		SL	1R (16 bits)		R _d \leftarrow C	Decode		
70111		AND	1R (26 bits)		R ₃₁ \leftarrow C			
81000		OR	Constant					
91001		XOR						
101010		NOT						
111011		Pass S ₁						
121100		Pass S ₂						
131101								
141110								
151111								

Opcode	Absolute Addr	Label
LD	5	Load :
ST	8	Store :
BEQ	12	BEQ :
BNF	14	BNE :
BL	16	BL :
BGE	18	BGE :
JAL	20	JAL :
J (PC,relative)	22	JPC :
J (Register indirect)	23	JR :
ADD (R-R-R)	24	ADDR :

SUB (R-R-R)	25	SUBR:
RR (R-R-R)	26	RRR:
RL (R-R-R)	27	RLR:
SR (R-R-R)	28	SRR:
SL (R-R-R)	29	SLR:
AND (R-R-R)	30	ANDR:
OR (R-R-R)	31	ORR:
XOR (R-R-R)	32	XORR:
NOT (R-R-R)	33	NOTR:
ADD (R-R-I)	34	ADDR:
SUB (R-R-I)	35	SUBR:
RR (R-R-I)	36	RR1:
RL (R-R-I)	37	RL1:
SR (R-R-I)	38	SR1:
SL (R-R-I)	39	SL1:
AND (R-R-I)	40	AND1:
OR (R-R-I)	41	OR1:
XOR (R-R-I)	42	XOR1:
NOT (R-R-I)	43	NOT1:

Unit-III

Q. 6. Explain CPU architecture types with examples.

Ans. There are two types of CPU : Architecture CISC, RISC. The type of work a processor carries out is defined by its instructions. These instructions are coded in binary. All modern processors have their own instructions built in for common tasks.

Having these instruction set built in allow the processor to carry out certain operations much faster.

The instruction sets that are built in depend on the processor's architecture. There are two main types of processor architecture on the market, CISC and RISC.

CISC [Complex Instruction Set Computer] : CISC processors have more internal instructions than its RISC counter part allowing a more diverse set of operations.

Although this may sound the best option. CISC processors are generally SQ lower due to the complexity of the instructions. Some people think the benefit of having more complex instructions built-in out weight the performance close, but it would depend on the applications that the processor is going to run.

RISC [Reduced Instruction Set Computer] : RISC processors, as the name suggests have fewer built in instructions, this can add to the overall speed of the processor due to the simplicity of the instructions, but again the performance would depend on the type of applications the processor was to be used for. Most modern processors have built in instructions specifically designed for certain applications such as 3D graphics,

audio manipulation etc. Example of this would be the MMX [Multimedia Extension] technology which Intel built in to its pentium architecture in the late nineties. This was a special set of internal instructions that allowed the faster processing of audio and visual algorithm.

Q. 7. How many 128×8 RAM chips needed to provide memory capacity of 2048 bytes. How many lines of address must be used to access 2048 bytes of memory. How many of these lines will be common to all chips? How many lines must be decoded for line select?

Ans. (i) How many 128×8 RAM chips needed to provide memory capacity of 2048 bytes,

$$\begin{aligned} \Rightarrow & 128 \times 8 \text{ RAM} \\ \Rightarrow & 128 \text{ bytes [8 bits]/chip} \\ \Rightarrow & \text{Nbr of chips} = 2048/128 \\ & = 16 \end{aligned}$$

How many lines of address must be used to access 2048 bytes of memory. How many of these lines will be common to all chips,

$2048 = 2^d$, where d is nbr is address lines $\Rightarrow d = 11$ we have 16 chips \Rightarrow 4 bits for chip select number of common address line is $= 11 - 4 = 7$.

Summary : 4 bits [MSB] to select the correct chip and 7 bits [LSB] to select the memory location inside the selected chip.

How many lines must be decoded for line select,

We need

$$\Rightarrow (4 \times 16) \text{ decoder.}$$

Unit-IV

Q. 8. (a) Explain direct mapped cache organization.

Ans. Cache organization with in the cache, there are three basic types of organization :

- (i) Direct mapped cache organization.
- (ii) Fully mapped cache organization.
- (iii) Set associative cache organization.

Here we explain Direct mapped cache organization.

In a direct mapped cache, lower order line address bits are used not access the directory. Since multiple line addresses map into the same location in the cache directory the upper line address bits [tag bits] must be compared with the directory address to ensure a hit.

If a comparison is not valid the result is a cache miss or simply a miss. The address given to the cache by the processor actually is subdivided into several pieces, each of which has a different role in accessing data.

Fully Associative Mapped : In fully associative mapping, when a request is made to the cache, the requested address is compared in directory against all entries in the directory if the request address is found [a directory hit] the corresponding location in the cache is fetched and returned to the processor; otherwise, a miss occurs.

Q. 8. (b) Explain processor level parallelism with example.

Ans. As processor designers extract higher levels of performance, the methods used are increasingly resulting in under utilisation of the hardware resources.

To reduce the processor cycle period the practice of "super-pipelining" that is pipelining functional units, has resulted in processors with much deeper pipelines. One of the consequences of this is that mispredicted branches have a much higher penalty than in processors with shorter pipelines. The delay between main memory and the on chip caches means that a cache miss can result in stalls of many tens [if not hundreds] of cycles.

Multiprocessor and NUMA systems can also increase the cache miss delay due to the coherency-protocol overhead and location of data. To take advantage of Instruction-Level-Parallelism [ILP], processor designers have increased the number of issue slots and functional units. In situations in which the application being executed cannot take advantage of ILP, the processor may only use a fraction of the available execution resources.

To increase performance using the existing, under-utilised, hardware and in an attempt to reduce the power performance disparity, processor designers are turning to multi-threading the processor.

Q. 9. Explain with examples various types of instructions in 8086.

Ans. There are 79 basic 8086 instructions in 8086 microprocessor :

AAA, AAD, AAM, AAS, ADC, ADD, AND, CALL, CBW, CLC, CLD, CLI, CMC, CMP, CMPS, CMPXCHG, CWD, DAADAS, DES, DIV, ESC, HLT, IDIV, IMUL, IN, INC, INT, INTO, IRET/IRETD, JXX, JCXZ/JECXZ, JMP, LAHF, LDS, LEA, LES, LOOK, LODS, LOOP, MOV, MOVS, LOOTE/LOOPZ, LOOPNZ/LOOPNE, NUL, NEG, NOP, NOT, OR, OUT, POP, POPF/POPED, PUSH, RCC, RCR, PUSHF/PUSHFD, REP, RERE/REPZ, RET/RETF, REPNE/REPNZ, ROL, ROR, SAHF, SAL/SHL, SAR, SBB, SCAS, SHL, SHR, STC, STD, STI, STOS, SUB, XOR, TEST, WAIT/FWAIT, XCHG, XLAT/XLATP.

Instruction Types :

Instructions vary from one CPU to another, General groupings possible :

- (i) Arithmetic/Logic
 - ⇒ Add, Subtract, AND, OR, Shifts.
 - ⇒ Performed by ALU.
- (ii) Data movement
 - Load, Store (to/from registers/memory)
- (iii) Transfer of control
 - Jump, Branch, Procedure call
- (iv) Test/Compare
 - Set condition flags
- (v) Input/Output
 - In, Out [only on some CPUs]
- (vi) Others.

Example :

Notes	Instruction	Operations
Addition	ADD dst, src	$dst = dst + src$
Subtraction	SUB dst, src	$dst = dst - src$
Compares set	CMP dst, src	$dst - src$