CS6303 - Computer Architecture.

Unit I :- (Refer Problems in class Work)

Eight ideas - Components of a computer system - Technology - performance - powerwall - Uniprocessor to Multiprocessor Instructions - operations and operands - representing instructions - logical operations - Control operations - Addressing and addressing modes.

Key features:- ARM - Advanced Risk Machine.

ARM - is an Instruction Set Architecture used for the embedded devices

Classes of Computing Applications :-

① **Desktop Computer :**

→ best known form of Computing, characterized by personal Computer.

→ Delivers good performance to single users at low cost and usually execute third party software.

② **Servers :-**

A computer used for running larger programs for Multiple user often simultaneously and accessed only through network

→ Carry large Workload

→ Consists of either single complex application or Many small jobs (Web servers).

→ provide greater Expandability of both Computing and ip/op capacity.

→ Some Servers can be used for file Storage, small business Application they no need to have Screen or keyboard.

Supercomputers:

→ Hundreds to thousands of Processors usually Terabytes of Memory and petabytes of Storage.

→ cost Millions to hundreds of Millions of dollars.

→ Uses: Weather forecasting,
Oil Exploration,
Protein structure determination
& other large scale problems.

$1KB = 1024 \text{ bits} = 2^{10}$
$1MB = 1024 KB = 2^{20}$
$1GB = 1024 MB = 2^{30}$
$1TB = 1024 GB = 2^{40}$

datacenter:

A room or Building designed to handle power, cooling, and Networking Needed of a large number of Servers.

datacenters are used by Companies like google, ebay contain 1000's of processor TB of Memory, PB of Storage

Embedded Computers:-

A Computer inside another Device for running predefined Applications or Collection of Softwares. Embedded Computers includes microprocessor found Cars, Computers in cell phone

television, airplane, cargo ship.

→ Embedded Application has unique Application requirements.

Multicore Microprocessor:-

A µp with multiple processors integrated in a single IC.

Flash Memory:-

→ Nonvolatile semiconductor memory.
→ cheaper & slower than DRAM
→ faster and expensive than Disk.

Used in Mobile phones they replace the use of Disk.

Unit-1

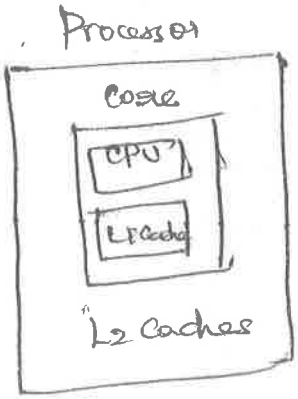
1) Components of Computer:-

There are 5 components of computer they are

1. Input
2. Output
3. Memory
4. Datapath } → processor.
5. & Control
6. ~~Control~~ ALU

Uniprocessors

A Uniprocessor system is defined as a computer system that has a single central processing Unit that is used to execute computer Task. As more and more Modern Software is able to to make use of Multiprocessing Architectures such as SMP (Symmetric multi processing) and MPP (massively parallel processing) the term Uniprocessor is therefore used to distinguish the class of computers where all processing tasks share a single CPU.



Instruction Register [IR]

The Instruction register (IR) holds the instruction that is currently being executed. Its output is available to the control circuits which generate the timing signal that control the various processing elements involved in executing the instruction.

Program Counter (PC)

Program counter incremented by PC ← PC + 4

Program counter (PC) keeps track of the execution of a program. It contains the Memory address of Executed.

* Common case is simpler than the rare case.
* Easy to enhance the common case.
* This idea is possible only with careful Equipmentation and Measurement.

④ Performance Via parallelism :-

Parallelism - (Simultaneous Execution of instruction) The parallelism technique used in the instruction is called Instruction Level Parallelism (ILP). Performance can be increased by performing operation in parallel.

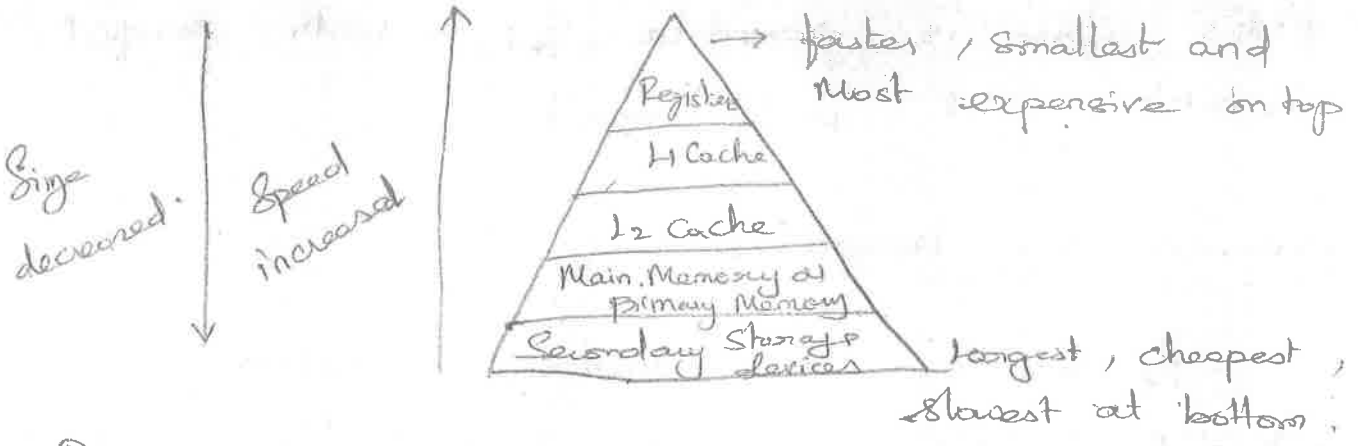
⑤ Performance Via pipelining :-

Overlapping the execution of the instruction to improve the performance of the CPU is called pipelining.

⑥ Performance Via prediction :-

Starting with the average guesses and start working rather than wait until you know for sure. Mechanism to recover from a misprediction is not much expensive

⑦ Hierarchy of Memories



Programmers want memory to be fast, large and cheap. Cost of memory today is often the majority of computer cost.

These problem can be addressed by Hierarchies of Memories where,

→ Fastest, Smallest and Most expensive on top
→ Largest, Cheapest slowest at the bottom.

Cache Memory gives illusion that main Memory is nearly as fast as the top of the hierarchy.

→ Layered triangle is used to represent Memory Hierarchy.

⑧ Dependability via redudancy:

→ Computer Need to be fast as well as dependable.

→ physical devices fail & we make systems dependable by including redundant components that take over when failures occurs and help to detect failures.

Logical Operations :-

Logical operations are used to control the program flow. The operations were added to programming language and instruction set architecture to simply among other things, Packing and unpacking of bits into words.

Logical operation	C operators	Java operators	ARM instruction
Bit-by-bit AND	&	&	AND
Bit-by-bit OR	|	|	ORR
Bit-by-bit NOT	~	~	MVN
Shift left	<<	<<	LSL
Shift Right	>>	>>>	LSR

Bit-by-bit AND Operation

AND is a bit-by-bit operation that leaves a 1 in the result if both bits of the operands are 1. for eg if register s_2 contains

0000 0000 0000 0000 0000 1101 1100 0000$_2$

and register s_1 contains

0000 0000 0000 0000 0000 0011 0011 0000$_2$

then after execution ARM instruction

AND $r5, $r1, $r2 ; reg $r5 = reg $r1 & reg $r2

the value of register $r5 is

0000 0000 0000 0000 0000 0001 0000 0000
2

AND can apply a bit pattern to set of bits to force 0's where there is a 0's in the bit pattern such a bit pattern with AND is traditionally called a "mask" conceals some bits.

② OR operation

It is a bit-by-bit operation that places a 1 in the result if either operand's bit is a 1. It is used to include bits in a word.
The ARM instruction for OR operations is

ORR $r5, $r1, $r2 ; reg $r5 = reg $r1 | reg $r2

the value in register $r5 is

$r1	0000	0000	0000	0000	0000	1100	0011	0000
$r2	0000	0000	0000	0000	0000	1101	1100	0000
$r5	0000	0000	0000	0000	0000	1101	1111	0000

③ NOT operation

The third logical operation is NOT takes one operand and places the 1 in the

result if one operand bit is a '0' and vice versa.

ARM instruction for NOT operation is MVN (Move Not)

MVN r_5, r_1 ; reg r_5 = ~ reg r_1.

Value of register r_5 is

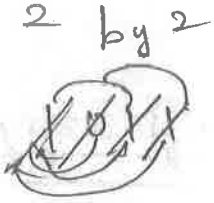
r_1 0000 0000 0000 0000 0000 1100 0011 0000$_2$

r_5 1111 1111 1111 1111 1111 0011 1100 1111$_2$

(4) Shift Operations :-

There are two shift operations they are

(i) Shift Left
(ii) Shift right.



Shift operation moves all the bits in a word to the left or right, filling the emptied bits with 0's.

(i) Shift left Operation :-

Shifts the bits by 'i' bits and fills the emptied by 0's.

The ARM instruction used to perform Shift left operation is LSL (logical shift left)

eg: if register r_0 contains,

0000 0000 0000 0000 0000 0000 0000 1001$_2$ = 9_{10}

When instruction to Shift left by 4 bits was executed then the new value is

0000 0000 0000 0000 0000 0000 1001 0000$_2$ = 144_{10}

Shift provides a logical benifit, shifting left by 'i' bits gives the same result as multiplying by 2^i.

in the above eg shift left by 4 bits of 9_{10} which results in value when 9_{10} is multiplied by 2^4 (4 bits shifts).

$$9 \times 2^4 = 9 \times 16 = 144.$$

Shift right operation:-

The dual of shift left is shift right operation. Shifts the bits right and fill with 0's.

The ARM instruction is LSR (Logical Shift Right)

eg:- if r_0 contains

0000 0000 0000 0000 1100 1000 0000 0000$_2$

Shift Right by 2 bits.

0000 0000 0000 0000 0011 0010 0000 0000$_2$

ARM Shift Operations are Not seperate Instructions. It offers the ability to shift the second operand as a part of any data processing instruction.

ADD $r_5, r_1, r_2, LSL\ \#2$; $r_5 = r_1 + (r_2 << 2)$

This instruction add register r_1 to register r_2 shifted left by 2 bits and puts the result in register r_0

eg:2

MOV r_6, r_5, LSR r_3 ; $r_6 = r_5 >> r_3$

(Moves the value of r_5 shifted by amount in reg r_3 into r_6).

→ The full ARM instruction set also include Exclusive OR (EOR) which set 1 when two bits diff, set 0 if they are same.

→ C compiler insert and extract fields using logical operations in ARM (AND, ORR, LSL, LSR).

→ Other instructions like ROR (Rotate Right) instead of discarding the bits on right shift brings them back into vacated location.

Performance

Definition:-

Performance is a measure which means how quickly a computer can execute the programs.

Performance is measured in different ways.

Performance can be increased if the execution time decreases and throughput increases.

Execution time or response time:-

The total time required for the Computer to complete a task including disk access, Memory access, I/O activities, operating system overhead, CPU execution time and so on. It is also called as elapsed time. (or) Time between the start and end of a task.

Throughput or Bandwidth:-

Throughput is defined as the no of tasks completed per unit time

To Max performance we want to minimize the execution time or response time of some task.

We can relate performance and execution time for a computer X.

$$\text{Performance}_X = \frac{1}{\text{execution time}_X}$$

For two computers X and Y if the performance of X is greater than performance of Y then,

$$\text{Performance}_X > \text{performance}_Y$$

$$\frac{1}{\text{Execution time}_X} > \frac{1}{\text{execution time}_Y}$$

$$\text{Execution time}_Y > \text{execution time}_X$$

(ie) Execution time of Y is longer than on X. if X is faster than Y.

When X is n times faster than Y we can write

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = n = \frac{\text{execution time}_Y}{\text{Execution time}_X}$$

Then execution time of Y is n times longer than X.

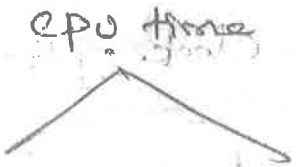
Measuring Performance :-

Time is the measure for computer performance. The computer that performs same amount of work in least time is the fastest.

A processor may work on several programs simultaneously. There is difference between the execution time and the time that the processor is working on behalf.

CPU execution time or CPU time :-

CPU time is the time the CPU spends computing for a task and doesn't include time spent on waiting for I/O or running other programs.



User CPU time (CPU time spent in the program)

System CPU time (CPU time spent in the operating system)

System performance refers to elapsed time

CPU performance refers to CPU time

Clock cycles or ticks or clock period or cycles :-

All the computers are constructed using a clock that determine when events takes place in hardware. These discrete time intervals are called clock cycles.

Clock period - Length of each clock cycle.

CPU performance & its factors :-

A formula that relates - Clock cycles and clock cycle time to CPU time is

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

Clock rate is inverse of clock cycle time,

$$\text{Clock cycle time} = \frac{1}{\text{Clock rate}}$$

The above equation gives,

$$\text{CPU execution time of a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

The above formula says that the designer can improve the performance by reducing the no of clock cycles required for a program.

Instruction performance :-

The computer has to execute the instructions to run the program. The execution time must depend on the number of instructions in a program.

∴ No of clock cycles required for a program can be written as

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \text{Average clock cycles per Instruction.}$$

CPI ("Cycles per Instruction) :-

It is the average number of clock cycles each instruction takes to execute.

Since different Instruction takes different amount of time depending on what they do CPI is the average of all the instructions executed in the program.

The Classic CPU Performance Equation:-

We can write the performance equation in terms of Instruction Count (No of Instructions executed by the program).

$$\text{CPU time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock cycle time}$$

Since Clock rate and Clock cycle time are inverses We can write

$$\text{CPU time} = \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock rate}}$$

Components of performance	Units of Measure
CPU execution time for a program	Seconds for the program
Instruction Count	Instruction executed for the program
Clock cycles per Instruction (CPI)	Average no of clock cycles per Instruction
Clock cycle time	Seconds per clock cycle

Instruction Mix:-

A measure of the dynamic frequency of instructions across one or many programs.

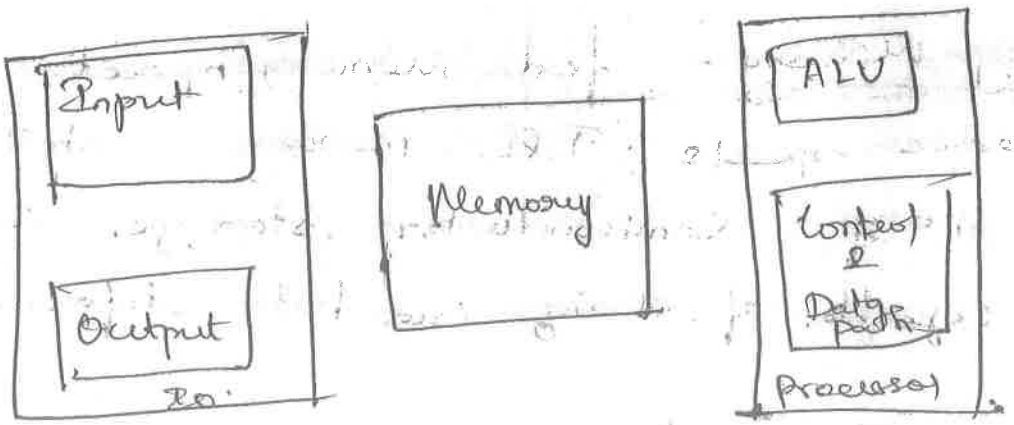
Since CPI varies by instruction Mix both the instruction count and CPI must be compared even if clock rates are identical.

Five Classic Components of a Computer.

The five classic components of a computer are

1. Input
2. Output
3. Memory
4. Data path
5. Control

→ Processor.



Input Unit :-

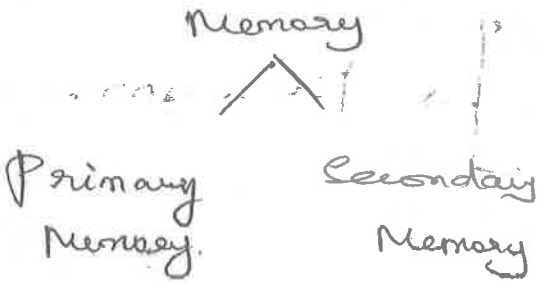
Computer accepts the coded information through input units which reads the data. When a key is pressed the corresponding digit or letter is automatically translated into binary codes and transmitted over a cable to memory or processor.

Some input devices are,

Keyboard, Joystick, track ball, Mouse, Scanner etc.

Memory Unit :-

Function of Memory Unit is to store programs and data. There are two types of Memory



Primary Memory - fast Memory operating at electronic speeds. The Memory contains a large no of semiconductor storage cells each capable of storing one bit information.

→ To provide easy access a distinct address is associated with each word location. Address refers to location.

→ No of bits in each word is "Word length" which range from 16-64 bits.

RAM is called as Random Access Memory. Which is Volatile Memory and datas in will be lost if system is turned off or shut down.

ROM - Read only Memory, is non Volatile Memory used for running Booting programs.

Cache - Smallest, fast RAM units are called Cache's. Cache Memory provides an illusion of Main Memory to the processor.

Secondary Storage :-

Primary Storage is expensive. Secondary Storage is used when large amount of data and programs have to be stored.

eg: Magnetic disk, tapes, optical disk (CD-ROM)

Arithmetic and Logic Unit :- (ALU)

Most Computers operations are executed in the ALU of the processor. When two numbers located in memory need to be added they are brought to processor and addition is carried by ALU.

When the data's are brought in they are stored in high speed storage elements called registers. Each Register store 1 word of data. ALU performs both Arithmetic & Logical operations like (Addition, Subtraction, Multiplication, division, AND, OR, NOT etc).

Output Unit :-

It sends the processed result to the outside world. Some output devices are Monitor, printer, Speaker etc. Some units such as graph display provide both output and Input function.

Control Unit :-

The control unit co-ordinates the memory, ALU, input and output units to store & process information.

Data transfer between the processor and memory are controlled by control Unit by timing signal.

:- Datapath :-

It manipulates the data coming through the processor. It also provides a small amount of temporary storage.

Technologies

Technology shapes what a computers will be able to do and how quickly they will evolve. There are some technologies that have been used for years.

year	Technologies used in computers	Relative performance / unit cost
1951	Vacuum Tubes	1
1965	Transistors	35
1975	Integrated circuits	900
1995	VLSI	2,400,000
2013	ULSI	6,200,000,000

Vacuum Tubes - It is a device that controls electrical signals through a vacuum sealed container. It was used in 1st generation of computers.

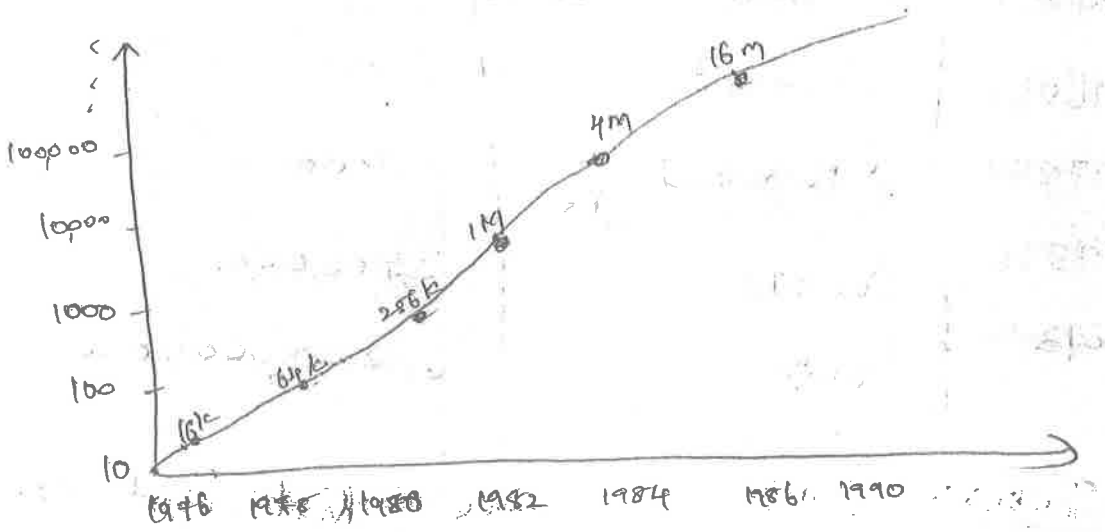
→ Also used for calculating, storing & controlling
→ Produced large heat.

Transistor :-

→ An on/off switch controlled by electricity.
→ IC's combines dozens to 100's of transistors into a single "chip".

→ The increase in no of transistors used in the IC's result in VLSI (Very large Scale Integrated Circuits) → Contains 1000's to Millions of Transistors.

The growth of DRAM (Dynamic RAM) technology for 35 years shows that Capacity Quadrupled every 3 years.



Manufacturing of chip:-

Manufacture starts with Silicon Ignot. Silicon is a substance made up of sand. It doesn't Conduct electricity well hence is called "Semiconductor".

→ Special chemical are added to make Silicon into conductor ~~and~~ Insulator, also as a switch

Silicon transform into the following.

① Excellent conductor of Electricity (Copper or Aluminum)
② " Insulator " (plastic or glass)
③ Areas that can conduct & insulate under special conditions (as a switch)

Silicon Crystal Ingot:

The Manufacturing process starts with Silicon Crystal Ingot

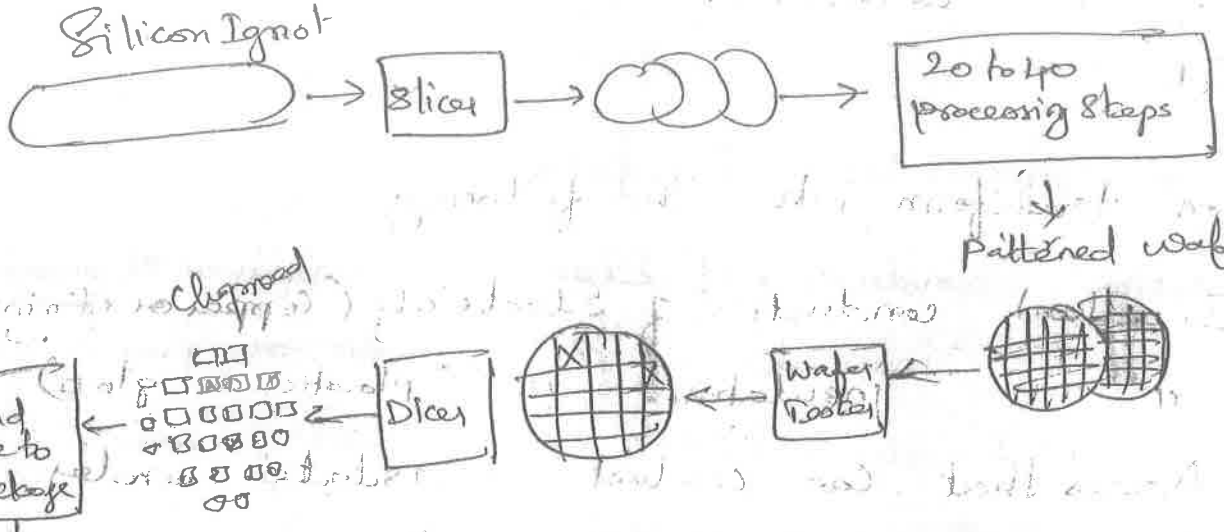
→ 8-12 inch in diameter
→ 12-24 inch long

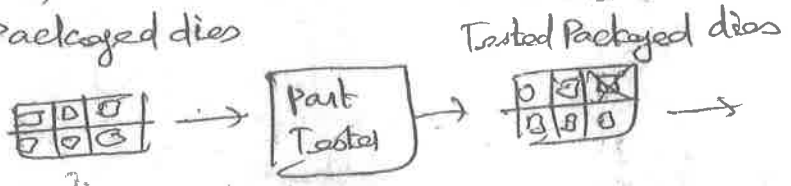
Wafers:

→ The Silicon Ingot is finely sliced into wafers no more than 0.1 inch thick.

→ These wafers are gone through a series of processing steps, where a pattern of chemicals are placed on each wafer creating a transistor,

Conductor and insulator.





Patterned wafers are tested with wafer tester and good parts of it are identified

The Yield is Percentage of good dies out of total dies on the wafer for eg $\frac{17}{20} = 85\%$ is yield.

The good dies are bonded into packages and tested one more time before shipping to customers

After wafer Testing the wafers are chopped up or diced into component called 'dies' known as 'chips'.

defect – A flaw in wafer or in patterning steps that can result in failure of the die containing that defect.

The cost of an IC rises quickly as die size increases, due to both lower yield and smaller no of dies that fit on wafer. To reduce the cost, we should shrink the large die. In 2012, 32 nm smalled die was processed.

Bonding :

After finding good die, they are connected to input / output pins of a package called bonding. These are tested and then shipped to the customer.

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per Wafer} \times \text{yield}}$$

$$\text{yield} = \frac{1}{1 + (\text{Defects per area} \times \text{D}}$$ or

$$\text{Dies per Wafer} = \frac{\text{Wafer Area}}{\text{Die Area}}$$

Hence depending on the defect rate, size of die and wafer cost are not linear in the die area.

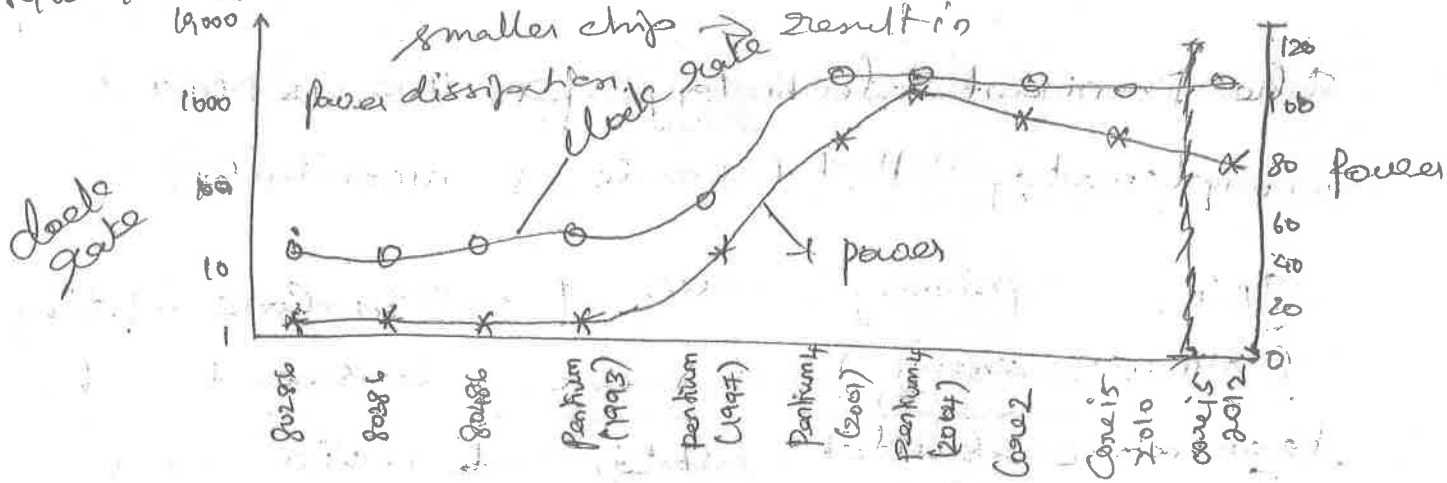
VLSI (Ultra Large Scale Integrated Circuits)

It is made up of embedding Millions of Transistors on a single silicon Semiconductor Microchip. VLSI is a succession of LSI & VLSI technology, same category of VLSI.

The Power Wall

Performance of the hottest chip grew by 52% per year 1986 – 2002, & grew only 20% in next 3 years → problem called power wall.

→ More transistors were used in smaller chip → result in power dissipation, clock rate



→ It shows that there is increase in the power & clock rate for eight generation of Intel microprocessors over 30 years.

→ Clock rate & power increased rapidly for decades then flattened. The reason is that they are co-related.

→ practical power limit is needed for cooling Commodity Microprocessor.

→ Architect days to reduce cost of powering, cooling 100,000 servers as their cost are high.

21) Just as measuring time in seconds is safer measure than MIPS, (the energy metric Joules is better measure than a power rate like watts. [Joules/second])

The Dominant technology for IC is CMOS (Complementary Metal oxide semiconductor).

CMOS - primary source of energy consumption (dynamic energy) - (ie) energy consumed when transistors switch states 0 to 1. Vice versa.

dynamic loading depends on Capacitive loading of each transistor and the voltage applied.

$$\text{Energy} \propto \text{Capacitive load} \times \text{Voltage}^2$$

This eqn is the energy of pulse during logic 0→1→0 or 1→0→1

then the energy of single transition is then

$$\text{Energy} \propto 1/2 \text{ capacitive load} \times \text{Voltage}^2$$

- Power required is product of energy of a transitions x the frequency of transitions.

$$\text{Power} \propto \frac{1}{2} \text{Capacitive Load} \times \text{Voltage}^2 \times \text{Frequency switched}$$

Frequency switched → function of clock rate.

Capacitive load per transistor is a function of both the number of transistors connected to an output (fanout) & technology.

→ power can be reduced by lowering the voltage. Since power is a function of voltage squared.

eg:- Suppose we want to develop a simple processor that has 85% of capacitive load of the more complex older processor. Further assume that it has adjustable so that exists so that it can reduce voltage 15% compared to processor B.

$$\frac{\text{Power}_{\text{new}}}{\text{Power}_{\text{old}}} = \frac{\{\text{Capacitive load} \times 0.85\} \times (\text{Voltage} \times 0.85)^2 \times \text{Frequency switched} \times (0.85)}{\text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency switched}}$$

(1)

$$\frac{0.85}{0.5} = 0.52$$

The new Processor user about half the power of the old processor.

Problem

→ Further lowering the voltage appears to make the transistors too leaky. Even today about 40% of power consumption in Server Chip is due to leakage.

To address problem designers have already attached large devices to increase cooling, and they turn off parts of the chip that are not used in a given clock cycle.

→ Although there are many more expensive ways to cool chip & thereby raise their power to say 300 watts these techniques are generally too expensive for personal Computers & even servers.

dynamic energy is the primary source of energy consumption in CMOS, Static energy consumption occurs because of leakage current that flow even when transistor is off.

In servers, leakage is responsible for 40% of energy consumption.

* ↑ in no of transistors ↑ power dissipation even if transistors are always off.
* A variety of design techniques & technology innovations are being deployed to control leakage but difficult to ↓ Voltage

Power is a challenge for IC for 2 reason

① power must brought in and distributed, around the chip.

② power is dissipated as heat and must be removed

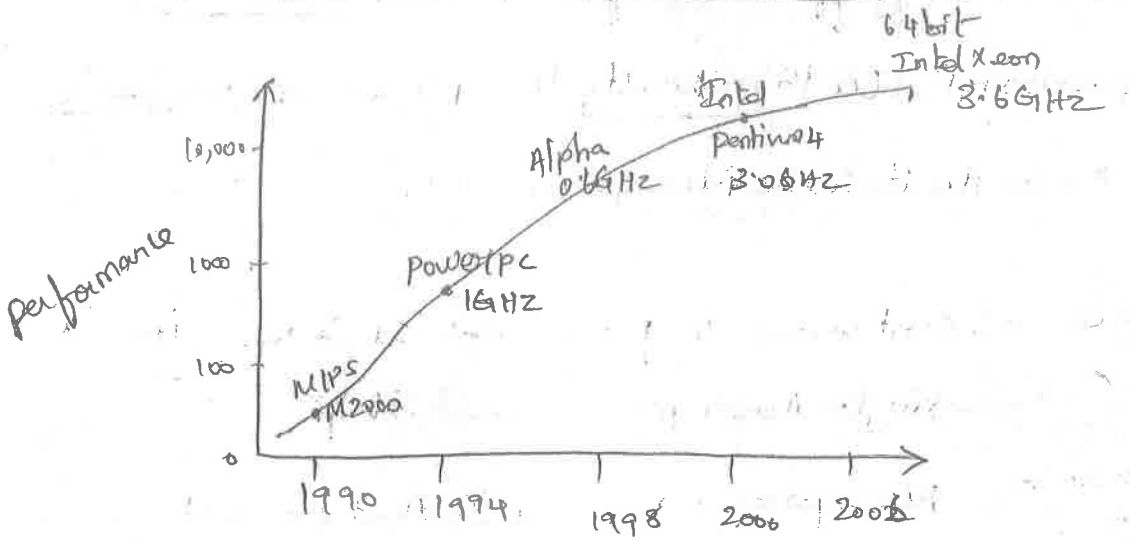
(Transistor → even when switch off regulates current at Voltage

required energy.

Leakage -

Leakage - loss of energy.

Uniprocessor to Multiprocessor



The chart measures the performance increase during years. The increase in growth was 52%. Performance of floating point oriented Calculation has also Increased faster.

Since performance increasing the response time is decreased. Instead of decreasing the response time with single program running on a sing processor It is better to go with 'Microprocessor' with multiple processor per chip. Where benifit is more on 'throughput' (ie) the no of tasks completed per unit time is more.

(1c)

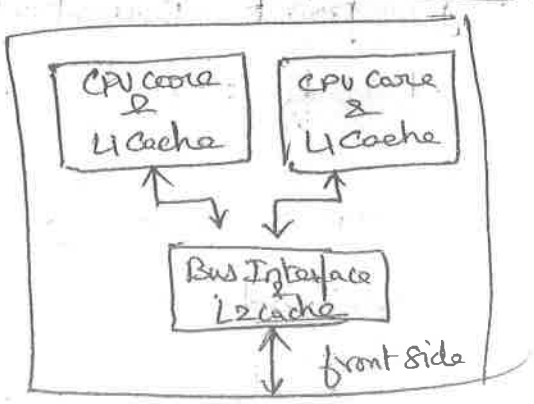
Processors are ~~before~~ referred as 'Cores'.

A microprocessor with multiple processor or core is called as Multicore Microprocessor.

Quad core – Contains 4 processor or core in a single microprocessor chip.

dual core – Microprocessor chip that contain 2 cores or processor.

Dual core processor



So programmer need rewrite the programs to take advantage of multiple processors.

Programmers should improve the code performance as no of cores doubles.

Hardware can execute Multiple Instructions at same time

It is so hard for programmers to write parallel programs due to difficulty in programming. In parallel Hardware the programmer must divide an application so that each processor has same amount of work to do at same time. (ie) Over head of Scheduling.

→

Advantages:

* Using Multiple Core increases the throughput
* Executes multiple instruction
* High performance.

Drawbacks

* Programming for parallel Hardware is difficult.
* Load balancing
* optimizing communication and synchronization is required.

This table shows no of processor or cores, clock rates, power of recent Microprocessors

product	Intel Nehalem	IBM Power6	Sun Ultra SPARC
Cores	4	2	8
clock Rate	~2.5 GHz	4.7 GHz	1.4 GHz
Power	~100 W	~160 W	94 W

INSTRUCTIONS

Instructions :-

Instructions are the commands that used to Command a Computer Hardware. It is the language that Computer Hardware can understand.

Instruction Set Architecture

It is an Interface between the Hardware and the Software. (It is a set of commands that can be understand by a ginen Architecture) → called Instruction Set. (or) Set of Instruction designed for a particular Hardware.

eg: ISA's are MIPS, ARM etc.

Stored Program Concept :-

Instruction & data of many types can be stored in memory as numbers.

MIPS - Millions of Instruction per Second

MIPS Registers

There are 32 registers (0 - 31)

Register no	Alternative Name	Description
0	Zero	The value 0
1	$at	(assembler Temporary)
2-3	$v0 - $v1	values from expression evaluation & function result.
4-7	$a0 - $a3	arguments.
8-15	$t0 - $t7	temporaries
16-23	$s0 - $s7	(Saved values)
24-25	$t8 - $t9	temporaries additional to $t0 - $t7
26-27	$k0 - $k1	Interrupt Handler.
28	$gp	global pointer
29	$sp	stack pointer. (point to location of stack)
30	$s8 / $fp	Saved value / frame pointer
31	$ra	return Address.

Types of Instructions :-

1. Arithmetic Instructions
2. Data Transfer Instruction
3. Logical Instruction
4. Conditional Branch Instructions
5. Unconditional Branch Instructions

Operations of the Computer Hardware :-

The operators supported by MIPS instruction Set Architecture can be categorized as follow.

Category	Instruction	eg	Meaning
Arithmetic	add	ADD r_1, r_2, r_3	$r_1 = r_2 + r_3$
	Subtract	SUB r_1, r_2, r_3	$r_1 = r_2 - r_3$
	add immediate	ADDI $r_1, r_2, 20$	$r_1 = r_2 + 20$
Data Transfer	Load Word	LW $r_1, 20(r_2)$	$r_1 =$ memory $[r_2 + 20]$
	Store Word	SW $r_1, 20(r_2)$	memory $[r_2 + 20] = r_1$
	Load half	LH $r_1, 20(r_2)$	$r_1 =$ memory $[r_2 + 20]$
	load unsigned	LHU $r_1, 20(r_2)$	//
	Store half	SH $r_1, 20(r_2)$	memory $[r_2 + 20] = r_1$
	Store byte	SB $r_1, 20(r_2)$	//
Logical	AND	AND r_1, r_2, r_3	$r_1 = r_2 \& r_3$
	OR	OR r_1, r_2, r_3	$r_1 = r_2 \mid r_3$
	NOR	NOR r_1, r_2, r_3	$r_1 = \sim(r_2 \mid r_3)$
	AND Immediate	ANDI $r_1, r_2, 20$	$r_1 = r_2 \& 20$
	OR Immediate	ORI $r_1, r_2, 20$	$r_1 = r_2 \mid 20$
	Shift left logical	SLL $r_1, r_2, 10$	$r_1 = r_2 << 10$
	Shift Right //	SRL $r_1, r_2, 10$	$r_1 = r_3 >> 10$
Conditional Branch	Branch on equal	BEQ $r_1, r_2, 25$	if $(r_1 == r_2)$ goto $PC = PC + 4$
	Branch on not equal.	BNE $r_1, r_2, 25$	if $(r_1 != r_2)$ goto $PC = PC + 4$
Unconditional Branch	Jump	J 2500	go to 10000
	Jump register	JR $ra	goto $ra
	Jump & Link	JAL 2500	$ra = PC + 4$, goto 10000

Arithmetic Instructions

These instructions are used to perform Arithmetic Instructions like Addition, Subtraction, Addition Immediate, Subtract Immediate. This type of instructions uses 3 operands, 2 source operand and 1 destination operand. It is also called register - register instruction

eg: ADD r_1, r_2, r_3 (Add of r_2 & r_3 and store to r_1)

ADDI $r_1, r_2, 20$

↳ $r_1 \leftarrow r_2 + 20$

Immediate Operand

SUB r_1, r_2, r_3

Data Transfer Instructions:-

This instruction moves the data between memory and registers. If the data is moved from Register to Memory is called 'Store' Instruction. If the data is moved from memory to Register is called 'Load' Instruction

eg: $a = b[8]$

LW $\$s_1$ (32)($\$s_3$) (We assume a as s_1 and b as s_3).

8 × 4 where 4 is 4 byte

here s_3 is called Base Register.

8 is called as the offset

Store word Instruction

ii) SW $S_1, 20, (\$S_2)$ Memory $(\$S_2 + 20) = \$S_1$

Here the value in the S_1 register is stored in the memory location calculated by adding the base register S_2 with offset 20.

Logical Instruction:-

There are different logical instructions they are AND, OR, NOR, ANDImmediate, ORImmediate, Shift Left, Shift right.

AND $\$S_1, \$S_2, \$S_3$	$\$S_1 = \$S_2 \& \$S_3$
OR $\$S_1, \$S_2, \$S_3$	$\$S_1 = \$S_2 \mid \$S_3$
ANDI $\$S_1, \$S_2, 20$	$\$S_1 = \$S_2 \& 20$

Conditional Branch Instruction:-

A instruction that requires the comparison of two values and that allows for a subsequent transfer of control to a new address in the program based on the outcome of the comparison.

Some Conditional Branch instructions are BNE, BEQ, etc.

eg:-

if source program is

```
if (i==j)
f = g+h
else
f = g-h
```

MIPS Instruction (Assume 'i' is in $s1 & j is in $s2).

```
CMP  $s1, $s2
BNE  else ; goto else if $s1 ≠ $s2
```

(ie) $i \neq j$.

Unconditional Branches Instruction

These instruction doesn't have any condition to be evaluated it just transfer the control to a new location. These instructions are called Unconditional Branch instruction.

```
eg: j 2500   [Jump to location (2500×4)
                                = 10,000
    JR $ra   (Jump register)
             go to ra (return address)
```

Operations of the Computer Hardware

Every computer must be able to perform Arithmetic operation.

The MIPS notation for addition is

```
add $s1, $s2, $s3
```

or

```
add a, b, c.   # The sum of b and c is placed in a.
```

The natural no of operands for an operation like addition is 3 where two no's are added and placed in one register.

Hardware for a variable no of operands is more complicated than Hardware for fixed number. This situation illustrates 3 design principles of Hardware design

Design Principle 1:

Simplicity favours regularity.

1) Regularity Makes implementation simpl
2) Simplicity enables higher performance at low cost

eg: Consider Source program

```
a = b + c ;
d = a - e ;
```

MIPS Instruction generated by compile is

```
add a, b, c   # Sum of b & c in a
sub d, a, e   Subtraction of a, e d.
```

Consider the complex statement

$$f = (g+h) - (i+j);$$

The MIPS instruction is...

```
add t0, g, h   # temporary variable to contain g+h
add t1, i, j   #  "       "     "  t1   "  i+j
sub f, t0, t1  # f gets t0 - t1. which
                 is (g+h) - (i+j)
```

Operands of the Computer Hardware :-

Operands can either be in registers or memory. Most of Arithmetic & logical Instructions use register operands. MIPS architecture has 32×32 bit register file. A group of 32 bits called a Word. A character requires 1 byte of storage. integer requires 1 word (4 bytes) of storage.

32 bit refers to the size of a register in MIPS architecture.

The reason for the limit of registers may be found in 2nd underlying principle of H/w design

Design principle 2: Smaller is faster

* A very large no of registers may increase the clock cycle
* Use few registers to conserve energy.

eg:-

Compiling $a = b + c$ we get

Add $\$s_1, \$s_2, \$s_3$ # $\$s_1$ contains b+c;

Memory Operands :-

If the program contains lots of variables and more complex data structures like array & structures. These complex data contain many more data elements than the registers in computer. So processor can keep only small amount of data in registers. Hence Data Structure (array & structure) are kept in memory.

→ Generally registers are faster than Memory & have high throughput than memory.

→ To achieve high performance & conserve energy, compiler must use register efficiently.

→ Most frequently used data is stored in register and rest of the data are in memory is called Spilling Register (Stack).

Instruction set for MIPS include some instruction that transfer data between memory & register. Such instructions are called data transfer instruction.

Memory → is a n no of registers in sequence to form an array.

To access any one of the memory location, the instruction must supply memory (Index) address. Index (displacement or offset).
Starting address is at 0 (refered as base Address).

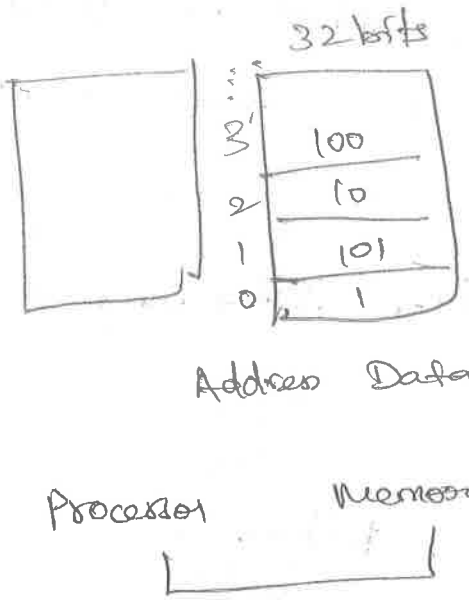
Memory Organization :-

All memory chips are byte oriented and it is viewed as a large single-dimensional array with address.

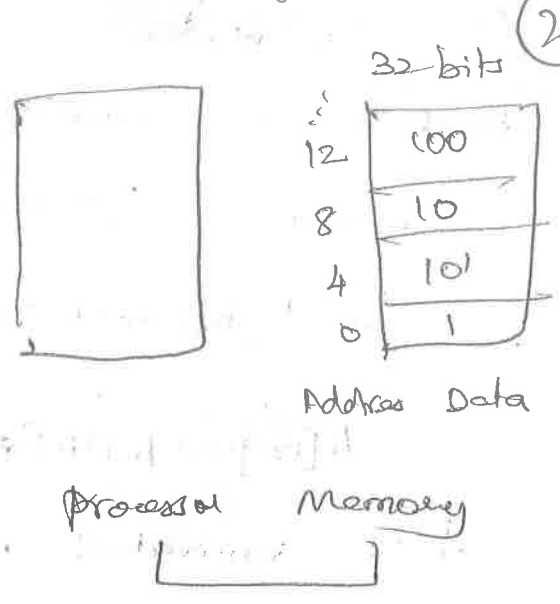
An address is an index into the array starting at 0 called Byte addressing.

To get the physical Address the offset address must be added to the base register. which is equivalent to 4 x 8 (32).

Word Address



Byte Address



eg: Index of third element is 2 then the value of memory[2] is 10.

The Data Transfer instruction that copies data between memory & Registers. Load Instruction Copies or Transfer data from memory to Register.

eg: LW $s1, 20($s2) (load Word) ($s1 = memory [$s2+20])

Store instruction transfer the data from register to memory

eg: SW $s1, 20($s2) (memory [$s1 + 20] = $s1)

Consider Source code :- (h is associated with $s2, Base Addr of Array A is in $s3.

g = h + A[8] →
LW $t0, 8($s3) # Temporary Register get A[8]
add $s1, $s2, $t0 # Base Address offset

In MIPS word must start at addresses that are multiples of 4. This requirement is called Alignment restriction.

In Byte Addressing. Consider the example

$A[12] = h + A[8]$

h is associated with register $\$s_2$,
base address of the array A is in $\$s_3$

The compiler generates.

$\$t_0 = Memory[\$s_3 + 32]$

Lw $t0, 32($s3) # Temporary reg $t0 gets A[8]

add $t0, $s2, $t0 # Temporary reg $t0 gets h + A[8]

The final instruction store sum into A[12] $(4 \times 12) = 48$
as the offset & register $s3 as the base register

Sw $t0, 48($s3) # stores h + A[8] back into A[12].

Constant or Immediate Operands :-

Incrementing an index to point to next element of an array uses Immediate operand More than 50% of MIPS arithmetic & logical instructions have a constant as operand.

Design Principle : 3 :- Make the common case fast.

→ Immediate operand avoids a load instruction and uses less energy.

a = h + 25 | assume h is associated S_2 and a is associated with S_3

Addi S_3, S_2, 25. } this can be written using load instruction.

→ { lw $t0, Addr Constant 25($s1)
add S_3, S_3, $t0.

Representing Instructions in the Computer

Instructions are kept in the Computer as a series of high and low electronic signals and may be represented as numbers. Each piece of an instruction can be considered as an individual numbers, and placing side by side these no's forms the Instruction.

$S0 to $S7 Maps to Register 16 to 23

$t0 to $t7 Maps to Register 8-15

then $s0 means the no of Register is 16, $s1 means 17 etc.

Opcode → Instruction format

	opcode	funct
Add	0	32
Sub	0	34
lw	35	-
sw	43	-

	opcode
J	2
Jal	3

A form of representation of an instruction composed of fields of Binary numbers. Or The Layout of Instruction in numbers is called Instruction format.

Machine language:

The numeric version of instruction is Machine language. Sequence of Machine language forms Machine Code.

It would appear tedious for reading and writing strings of binary No's we avoid them by using higher base like 'Hexadecimal' other than binary. Since almost all computer data sizes are multiples of 4, Hexadecimal is popular

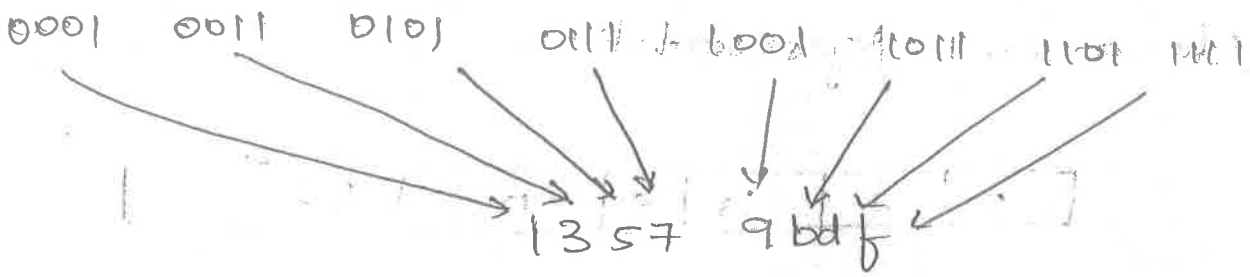
Hexa	binary	Hexa	binary	Hexa	Binary
0_{16}	0000_2	5_{16}	0101_2	A_{16}	1010_2
1_{16}	0001_2	6_{16}	0110_2	B_{16}	1011_2
2_{16}	0010_2	7_{16}	0111_2	C_{16}	1100_2
3_{16}	0011_2	8_{16}	1000_2	D_{16}	1101_2
4_{16}	0100_2	9_{16}	1001_2	E_{16}	1110_2
				F_{16}	1111_2

eg: Convert $ecas\ 6420_{Hex}$ to Binary

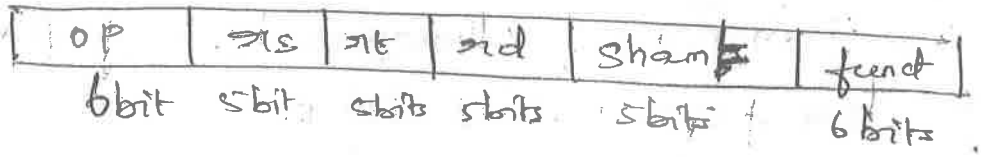
ecas 6420

1110 1100 1010 1000 0110 0100 0010 0000_2

- Convert 0001 0011 0101 0111 1001 1011 1101 1111 $_2$ into Hexadecimal.



R-type or R-format :-



This format is used for Register Instructions.

MIPS fields :- Each of segments of instruction is called field.

There are 6 fields in the R-type Instruction.

op - Basic operation of the instruction, called opcode
rs - The first register source operand
rt - The second register source operand
rd - The register destination operand, hold the result
shamt - shift amount
funct - function, this field is called function code.

Opcod - field that denotes operation & format of an Instruction.

Eg:- Consider MIPS language for instruction

add $t0, $s1, $s2

The decimal representation

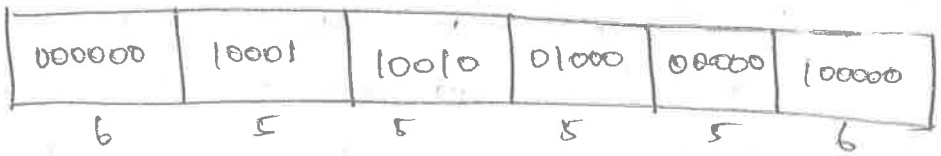
First & last field is 0 & 32 which says that operation is Addition

2nd field gives no of the register of first source operand $s1 which is 17

3rd field gives no of the register of the 2nd source operand $s2 which is 18

4th field gives the destination operand's register no $t0 is 8.

5th field for 'shift' which is unused. So set to 0.



The Binary Representation is.

A problem occurs when an Instruction needs longer fields that these above. eg load word instruction must specify 2 register and

a constant. If the address were to use one of the 5 bits field in the format of the load word instruction would limit to only 25 or 32

The constant used often needs to be larger than 32. So 5-bit too small to use.

Hence we have conflict to keep one single Instruction format.

Design Principles: Good Design Demands good Compromises.

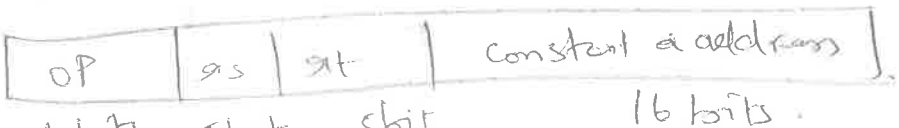
I-format or I-type :

The second format of instruction used for immediate and Data transfer instructions.

The last 3 fields of the R-format instruction is joined as a single field of 16 bits.

The 16 bit address means a load word instruction can load any word within a region of $\pm 2^{15}$ or 32768 bytes of address in base Register RS.

Adds immediate is limited to constant no longer than $\pm 2^{15}$. ~~eg: lw $t0, 32($s3) # temporary reg $to get A[8]~~



eg:

lw $to, 32($s3) # temporary Reg $to gets A[8]

decimal format

35	19	8	32
6bits	5bits	5bits	16 bits.

Binary format

100011	10011	01000	0000000001 00000
6bit			

Instruction	Format	OP	rs	rt	rd	shmt	funct	Ad
add	R	0	reg	reg	reg	0	32	NA
sub	R	0	reg	reg	reg	0	34	NA
addi	I	8	reg	reg	NA	NA	NA	Const
lw	I	35	reg	reg	NA	NA	NA	add
sw	I	43	reg	reg	NA	NA	NA	Add

A[300] = h + A[300].

eg:

```
LW   $to, 1200($t1)  # temp reg $to get A[300]
add  $to, $s2, $to   # tem reg get $to get h+A[300]
SW   $to, 1200($t1)  Stores h+A[300] back into A[300]
```

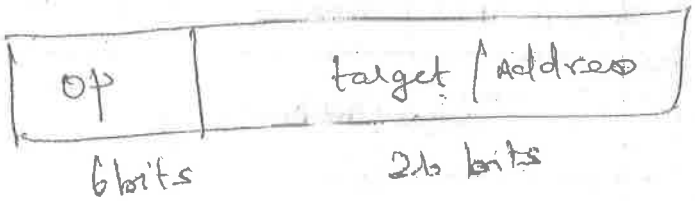
6bits	5bits	5bits	5bits	5bits	6 bits
OP	rs	rt	rd	address/ shmt	funct
35	9	8		1200	
0	18	8	8	0	32
43	9	8		1200	

Write in Binary

100011	01001	01000	0000 0100 1011 0000
000000	01001	01000	01000	00000	100000
101011	01001	01000	0000 0100 1011 0000

J - type or J - format :-

- This format is used for Representing Jump Instruction. This representation has only two fields.

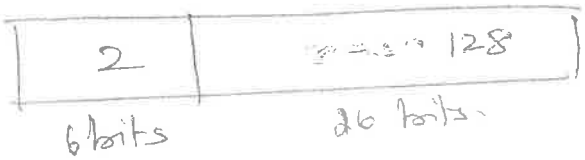


where 6 bits are opcode, and remaining for holding the target or Address.

Instruction	opcode
J	2 [000010]
Jal	3 [000011]

eg: J 32

Deimal Representation



Binary Representation

MIPS Instruction format

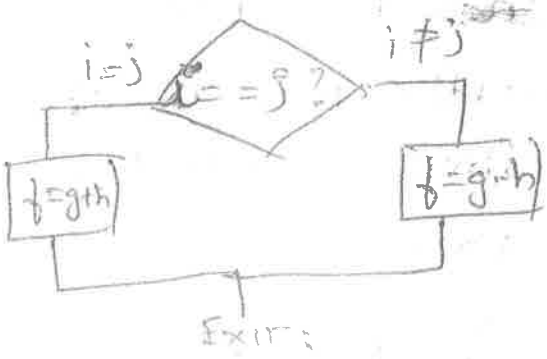
Format	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	comments
R	OP	rs	rt	rd	shamt	funct	Arithmetic
I	OP	rs	rt	addr / Immediate			Data Transfer, Branch, immediate
J	OP	Target Address					Jump.

Control Instructions or Control operations

Control Instruction includes Conditional branch, Unconditional branch, procedures, Switch Case Statements. Decision Making is commonly represented in programming languages using the if statement, combined with goto and Lables.

Conditional Branch:

An Instruction that requires the Comparison of two values and that allows for a subsequent transfer of control to a new address in the program based on the outcome of the Comparison.



eg: if (i==j) f=g+h
else f=g-h.

MIPS Instruction

```
bne $s3, $s4, ELSE  # goto else if i≠j
add $s0, $s1, $s3   # f=g+h (skipped i≠j)
j EXIT              # goto exit
ELSE : sub $s0, $s1, $s2   # f=g-h
Exit :
```

Loops:

Decision are important both for choosing between two alternatives — found in if statements.

Loops - Iterating a computation.

```
while (save[i] == k)
  i += 1;
```

MIPS code:- k is assigned with $s5, $s6 to contain save[].
$s3 associated with i.

→ Add i to base array save.

```
Loop: sll $t1, $s3, 2   # $t1 = i x 4 due to byte   i x 4 = 2^i (shift 2 time = multiply Address word by 4)
add $t1, $t1, $s6       # to get address of save[i], problem we need to add ($t1) base of $s6
lw $t0, 0($t1)          # temp reg $t0 = save[i]
bne $t0, $s5, EXIT      got to exit save[i] ≠ k.
addi $s3, $s3, 1        # i = i+1.
jLoop #  goto loop
EXIT.
```

basic block :-

A sequence of instruction without branches and without branch targets or branch labels called basic block.

Case / Switch Statements :-

A Switch Case statement that allows the programmer to select one of many alternatives depending on a single value.

The simplest way to implement Switch is through a sequence of conditional tests, turning the switch statement into chain of if-then-else statement.

The Alternatives may be more efficiently encoded as a table of addresses of alternative instructions sequences called Jump Address table or Jump table.

Jump Register (JR) instruction is used for this condition.

Jump table - A table of Address of alternative instruction sequences.

Supporting Procedures in Computer Hardware :-

A procedure, or function is used to structure programs which supports code reusability. Procedures are one way to implement abstraction in software.

Procedure :-

A stored subroutine that performs a specific task based on the parameter with which it is provided.

Execution of procedure must follow the following six steps :-

1. Put parameters in a place where the procedure can access them
2. Transfer control to the procedure
3. Acquire the storage resources needed for procedure
4. Perform Desired tasks.
5. Put the result value in a place where the calling program can access
6. Return control to the point of origin.

MIPS Instruction for procedure

Jump and link (Jal) instruction

Jal procedure Address - Jumps to an address and simultaneously saves to address of the following instruction in Register $ra.

(ie) $ra = PC+4

MIPS software follows the following convention for procedure calling in allocating its 32 registers

$a0 - $a3 - 4 arguments register in which to pass parameters

$v0 - $v3 - value register in which to return values.

$ra : - to return to point to origin.

Jr $ra

→ It transfers control Back to the caller, the jmp function just has to jump to the address that was stored in $ra

Caller - A program that calls a procedure & provides necessary parameters

Coprocessor Instruction:-

Coprocessor are additional processor which performs floating point operations

All MIPS processor has two standard coprocessor CP0, CP1. Future inclusion of two additional Co-processors CP2 and CP3.

All Co-processor use opcode 01000x
Where the last two bits specify coprocessor number

opcode for Co-processor 1 = 010001

eg: add.s fd, fs, ft

[floating point add Single].
↳ 32 bit Single precision

Addressing Mode :-

Addressing Mode :-

The term addressing mode refers to the way in which the operands of an instructions are specified. It is also defined as the different ways of determining the address of the operands.

The Information contained in the instruction code is the value of the operand or the address of the result or operand.

There are 5 types of MIPS addressing modes :-

1. Immediate Addressing
2. Register Addressing
3. Based or displacement addressing
4. PC - relative addressing
5. Pseudo/direct Addressing

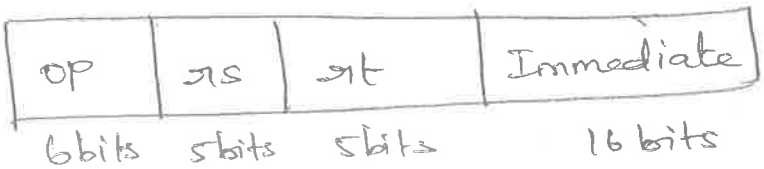
1) Immediate Addressing :-

In this addressing mode, the operand is a constand and it is specified as the part of instruction.

Immediate addressing has the advantage of not requiring an extra memory access to fetch the operand. The operand is limited to 16 bit in size.

The branch instruction format can also be considered an example of Immediate addressing since destination is held in instruction.

eg : (i) Add s_0, s_1, 20

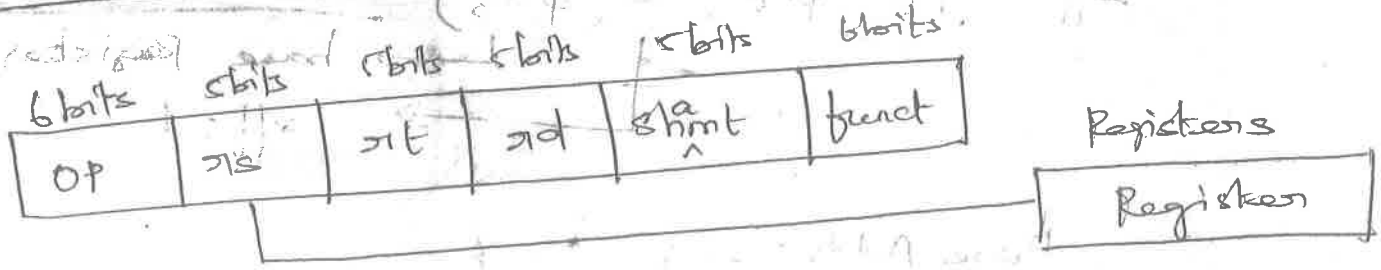


Immediate addressing.

2) Register Addressing :

In this addressing Mode, Name of the register is used to specify operand. This type of addressing Mode can be represented using R-format or R-type instruction representation. Here each register fields are 5 bits long and opcode and funct are 6 bits long.

Register Addressing



If it is a shift operation like SLL, SRL then shamt bit is set to 1 otherwise it is set to '0'.

eg: Add S_0, S_1, S_2

S_0 – Register 16
S_1 – Register 17
S_2 – Register 18

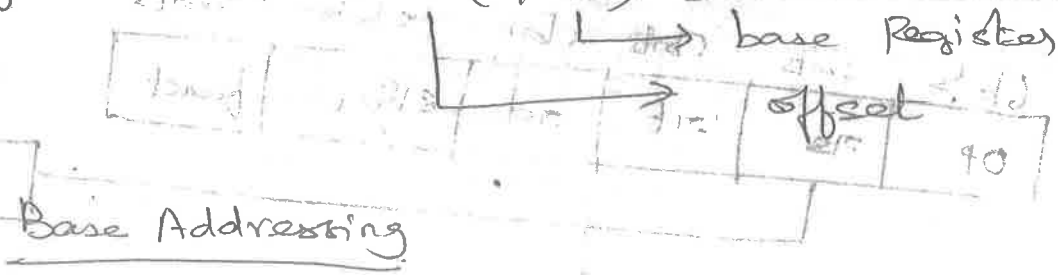
0	17	18	16	0	32

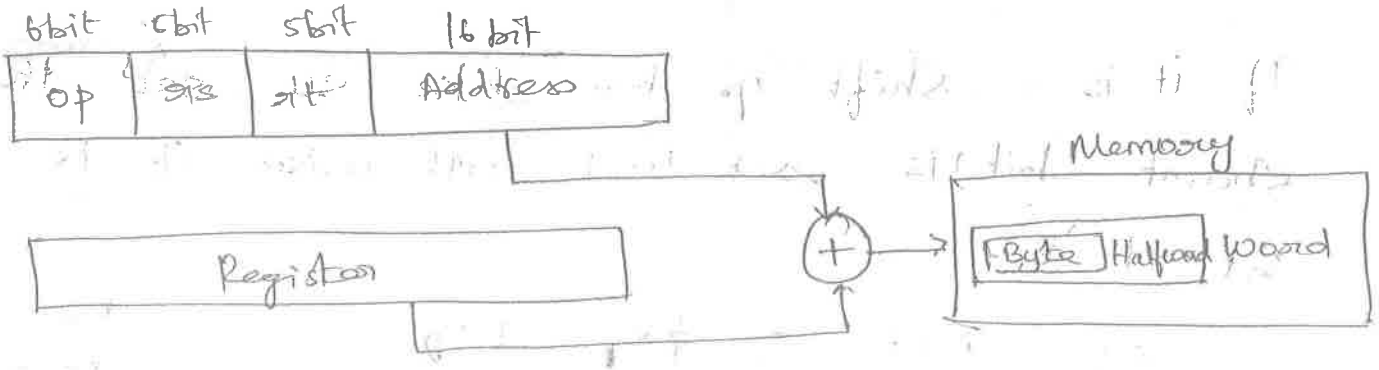
③ Base or displacement Addressing :-

In this mode, the operand is at the memory whose address is the sum of a Register and a constant in the instruction. The address is called as the "base address" and the constant to be added is called as "offset".

This type of addressing uses I-type or I-format instruction Representation.

eg: lw $t0, 32($s3) # t0 = Memory[$s3+32]





4) Pc - relative Addressing :-

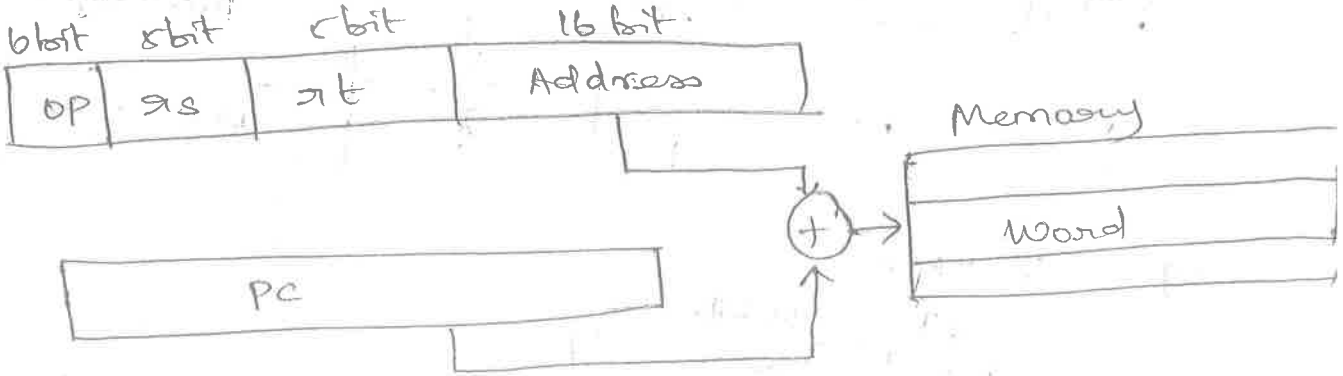
In this addressing mode, the branch address is sum of the Pc and constant in the instruction. It is used in Conditional branch. Branch instruction can only move, above or below the program Counter because the offset is 16 bit, no program Counter could be bigger than 2^{16} which is too small. It uses I-format instruction Representation.

Program Counter = Register + Branch Address

eg: beq $s_3, $s_4, L_1

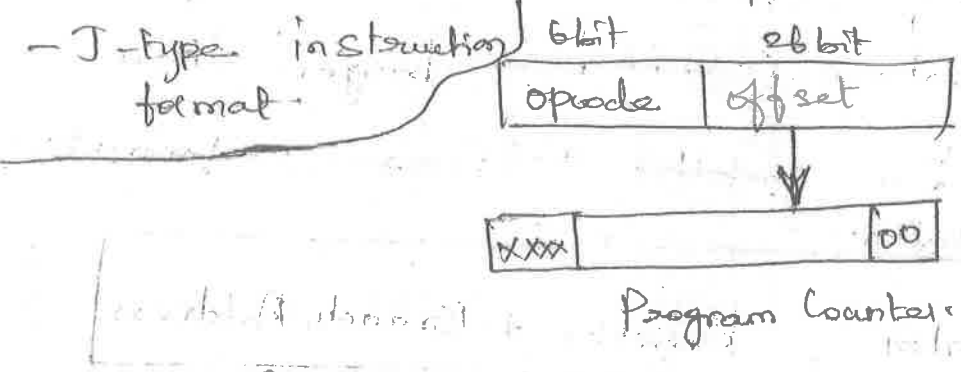
if $s_3 = $s_4 move to Label L_1.

Pc - relative addressing



5). Pseudodirect Addressing :-

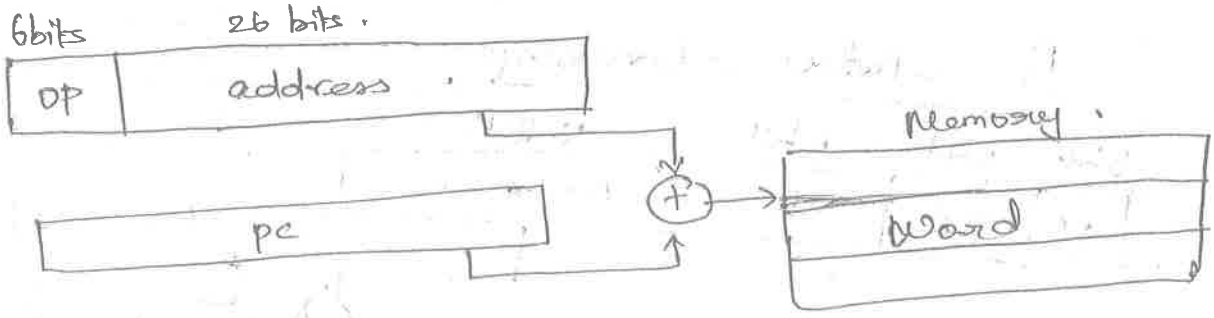
In this addressing mode, memory address is mostly embedded in the instructions. The effective address is calculated by taking the upper 4 bit of the program counter (PC) concatenated to the 26 bit immediate value, and the lower two bits are 00 as shown in below. The MIPS Jump instruction ~~format~~ uses - J-type instruction format.



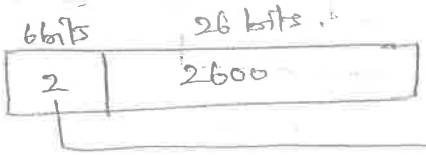
eg: J 2600 [goto 10400]

→ 2600 x 4 [byte addressly

Pseudo direct Addressing



eg: J 2600



→ opcode value for Jump instruction.