CS 6303

Computer Architecture

UNIT-1: OVERVIEW & INSTRUCTIONS.

Computer Architecture definition:

CA deals with designing and Implementation of instruction set, information format and memory addressing techniques of compu

CA is also concerned with structure and behaviour of the various function modules of computer and how they interact to provide the processing need of users.

Classes of Computers:

Feature	PMD	Desktop	Server	warehouse computer.	embedded
price	$100-1000	$1000-$2500	$5000-$10000	$100000-$200000	$10-$1K.

mainly 5 type of computer. They are

1. personal mobile device.
2. desktop / Laptop.
3. Server.
4. ware house Computer.
5. embedded computer.

1.1. EIGHT IDEAS (or) EIGHT RECENT IDEAS IN CA:

From the past 50 years of computer design, the computer architects propose 8 great ideas. They are

1. Design for moore's law.
2. Use Abstraction to simplify design.
3. Make the Common Case fast.
4. Performance via parallelism.
5. Performance via pipelining.
6. Performance via predictions.
7. Hierarchy of memory.
8. Dependability via Redundancy.

(2) 1. Design for moore's law:

* Developed by Gordon E. moore, co-founder of Intel.
* He states that "Design of Computer may take many year to complete".
* The design start with existing technology but at the end of design technology may grow and product has to be reworked.
* So computer designer must know / image about the design and when the design finish with future technology.
* According to moore's law "Integrated circuit resources doubles every 18-24 months". ie "Resource available per chip can easily double or quadruple btw start & end of design".

2. Use Abstraction to Simplify design:

* Abstraction are used to represent the design at different levels of representation for both hardware & software.
* In abstraction "lower level details are hidden to offer simple models at higher levels".
* This technique reduces the design time considerably.

3. make the common case fast:

* Any system must have 2 kind of function such as 1. common case 2. rare case.
* making the common case fast will enhance performance better than optimizing the rare case.
* design for common case is simpler than rare case also.

4. performance via parallelism:

* parallelism is a process of performing multiple jobs simultaneously. (parallely.)
* so that performance of computer can be increased.
* In parallelism larger problems are often subdivided into smaller unit and solved concurrently. : throughput will be increased automatically.

5. Performance via pipelining:

* pipelining is a method of data processing where the output of one element/Job is the input to next element/Job.
* So automatically improve in performance of computer.
* Rather than processing each instruction sequentially, every instruction is split up into a sequence of steps so that each step can be executed concurrently and in parallel to improve perform ance.

6. performance via prediction:

* In some cases it is better to predict (guess) and start working rather than wait until we know for sure.
* Above said condition should be done by assuming that mechanism will recover from misprediction is not too expensive.
* Usually Computer Architects will improve sys performance by proper prediction about future.

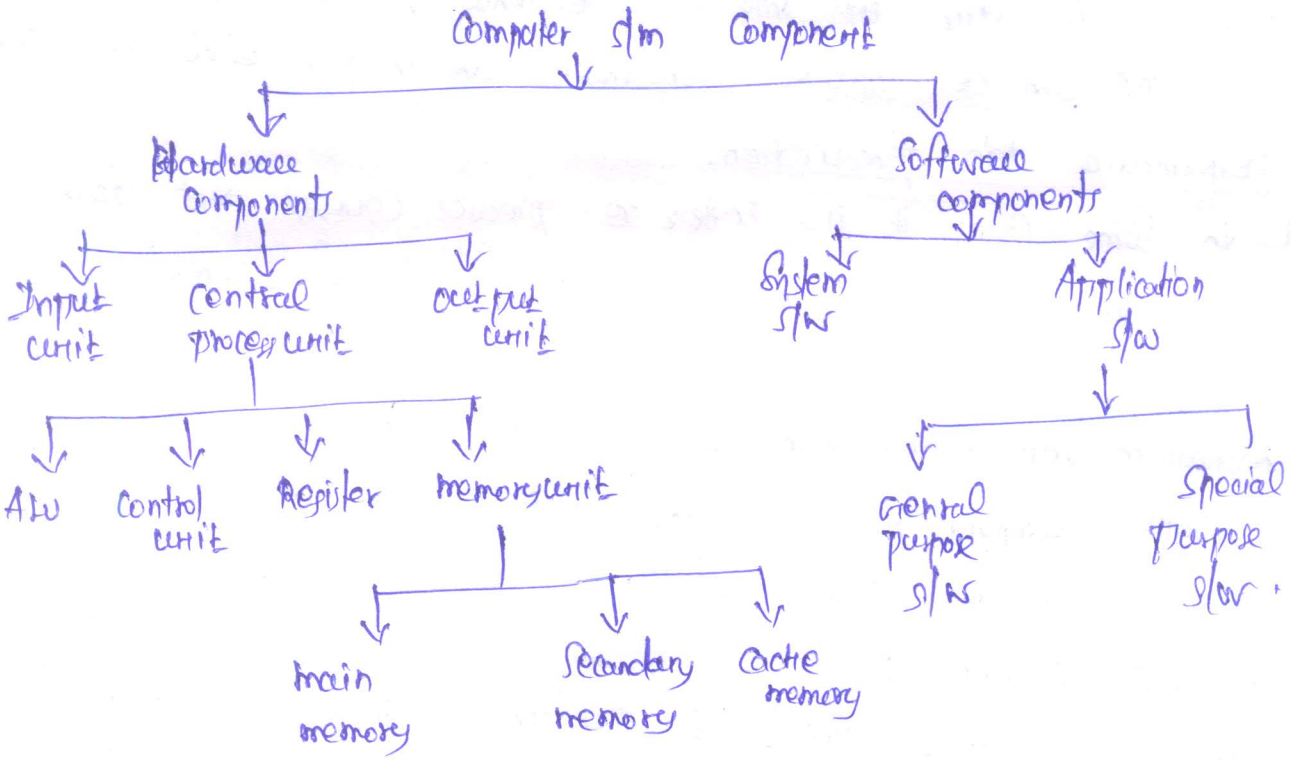
7. Hierarchy of memory:

* memory is most important functional component in computer.
* ~~memory~~ performance of computer directly depends on the memory speed and its size.
* User need the memory to be very fast, large but cheaper in price.
* unfortunately its is impossible to meet all 3 in one memory type.
* Nowa days computer cost depends on cost of memory. So then need to know the computer memory hierarchy.
* So fastest, smallest & most expensive memory/bit at top of hierarchy & slowest, largest & cheapest per bit at bottom of hierarchy.

8. Dependability via redundancy:

* Computers not only need to be fast but also need to be dependable.
* we know that single component failure in computer will shut down entire system.
* So we need to make the system dependable by including redundant components that can overtake/take over when failure occurs.

Computer is defined as electronic machine that accepts input from user/other device and process the data by performing some operation and produce the result in the form of human required.

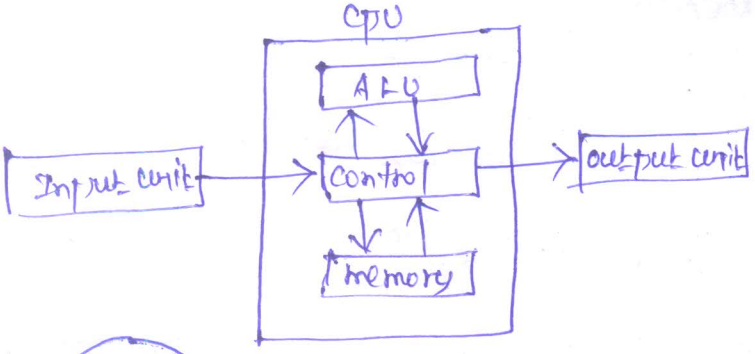


1.2.1. Hardware:

Electronic components interacted/interconnect in the computer s/m is referred as Hardware. Hardware is also called physical components of computer (i.e) "able to touch & feel".

The H/w s/m mainly contain 3 units. Those are

1. Input unit
2. Central processing unit.
3. Output unit.



1.2.1.1. Input Unit:

* It is a electro mechanical device which allows the user to feed the Input to computer.
* Input unit is capable of accepting Input in any form and able to convert the I/p that can be easily understandable by computer (usually in I x O's).
* Ex: 1. mouse 2. Keyboard 3. Scanner 4. Light pens. 5. Joystick.

1.2.1.2. Central processing Unit:

* It is the brain/ heart of the computer. It mainly do the 4 operations.

4 operation of cpu:

1. Fetching the Instruction from memory.
2. Decoding the Instruction.
3. Executing over Instruction.
4. Storing the results back into memory.

* Simply cpu convert the I/p into required o/p.
* It mainly 3 units then are
  1. Arithmetic & Logic Unit (ALU).
  2. Control unit (CU)
  3. memory Unit.

1.2.1.2.1. Arithmetic & Logic Unit:

ALU is responsible for both Arithmetic & logical operations.

Arithmetic unit: * it mainly used to perform arithmetic operation such as add, sub, mul, division.

* To perform this operation operand has to bring from memory to processor after performing the results has to store back in memory.

(b) Logic unit:

* It mainly used to perform logical operations such as AND, OR, NOT and also comparision b/w the previous operations.
* It include $<$, $<=$, $>$, $>=$, $==$, $!=$ operations.

1.2.1.2.2. Control unit:

* It control & coordinate all the units.
* It act like a supervisior and control all other activities.
* It send & receive control signals from previous unit and act upon that.

1.2.1.2.3. memory unit:

* It mainly used to store programs and data & results.
* memory - How processor store & use immediate results/Instructions.
* It mainly of 3 type of memory.
    1. primary or main memory.
    2. Secondary memory.
    3. Cache memory

1. primary or main memory:

* It is fast memory that operates at electric speed.
* program must be stored in memory while they being execution.
* It Contain large no. of semiconductor storage cells.
* Each cell capable of carry 1bit information.
* Size of the main only is low but much expensive/costly.
* It is of 2 type. Then are 1. RAM. 2. Rom.

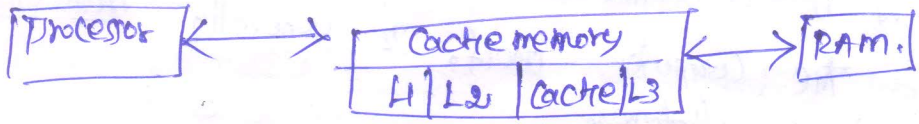
RAM	Rom
Random access memory	Read Only memory.
Ram is volatile	It is non volatile.
Data stored in RAM will be erased when we switch off the System.	Data stored in Rom will not be erased even if power is off.
It allows for both read & write.	It allow only for read.

2. Secondary memory:

* It is also auxiliary memory.
* It is mainly used for storing program & data & results.
* Where one can store very amount of data it is size very large but price is also cheap compared to main memory
* Ex: 1. Hard disk 2. CD-rom 3. DVD 4. Flash memory 5. magnetic tape.

3. Cache memory:

* It is a high speed memory that is located in between RAM & CPU(processor).



* It increase the speed of processing.
* processor first search the cache memory for data if found then it will use. Else it go for the RAM for data.
* It is of 2 form L1, L2, L3 cache.
* Size range from 256 KB to 2 MB only.

1.2.1.2.4. Register

* It is a special-purpose, high speed temporary memory unit.
* It hold various types of informations such as data, instruction, address, intermediate results.
* It mainly contain Register. They are
1. PC (Program Counter) 2. ACC (Accumulator). 3. IR (Instruction Register)
4. MAR (memory address register) 5. MBR (memory Buffer Register).
6. MDR (memory data register).

1.2.1.3. Output unit:

* It is mainly used to provide/show the output to user.
* It is capable of converting the machine language to human understanding form/language.
* The o/p may be 1. Hard Copy 2. Soft Copy.
* Ex: 1. monitor 2. printer. 3. Speaker. 4. plotters.

* def: It is a set of program, that is designed to perform a well defined function task. Program is a sequence of instruction written to solve a particular problem.
* It is also called "It tells computer what to do & how to do".
* It mainly of 2 types. Then are
  1. System s/w 2. Application Softwares.

1.2.2.1. System Softwares:

* def: S/m s/w is a collection of program designed to operate, control, process the computer itself. It generally written by the computer manufacturers.
* Features of S/m s/w:
  1. Fast in speed. 3. difficult to design 5. written in low level language.
  2. Small in size. 4. difficult to understand.
* Types of S/m s/w: It mainly of 2 types.
  1. System support prg: provide routine service function to computer prg & user.
  2. System Control prg: control the execution of programs.
  3. System development prg: assist in Creation of computer programs.
* Ex: 1. operating System 3. Editor 5. Linker 7. Interpreter.
  2. Device Drivers 4. Compiler 6. Loader 8. Debugger.

1. Editor: Create & modify source prg.
2. Assembler: Translate assembly language into machine language.
3. Linker: used to join several object file into one large object file.
4. Interpreter: used to convert the source code to machine level language like by line only one statement at a time.
5. Compiler: Convert the source code into machine language at a time by taking entire source prg. into account.
6. Debugger: allows the user to load his object code prg. into s/m memory & execute & debug them.
7. operating System: perform resource mgmt, provide Interface b/w user and machine.

1.2.2.2. Application S/w:

* It is used/designed to accomplish particular task. It mainly used by the end user.
* mainly designed to solve particular problem of user.
* This s/w may written in any language such as C, C++, Java etc.
* It mainly 2 & types. They are

1. General purpose 2. Special purpose appl. s/w.

1. General purpose:

* These are designed to satisfy common needs of various business.
* EX: 1. Ms word processor 2. Spreadsheet 3. Image editor.
4. presentation s/w 5. web browser 6. DTP 7. DBMS.

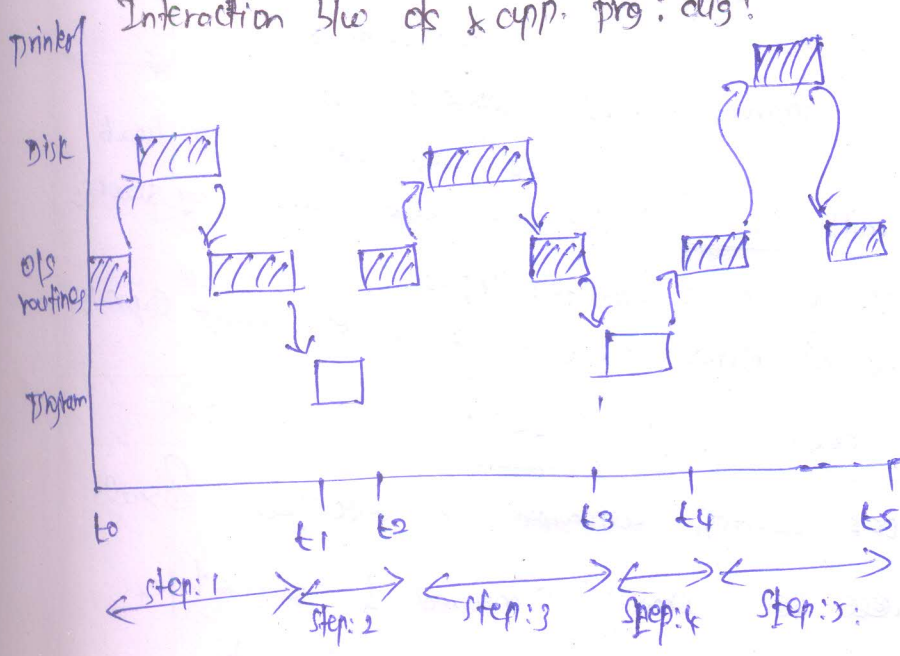
2. Specific purpose:

* These are created to satisfy specific needs of organization.
* EX: 1. payroll s/w 2. Railway ticket 3. ATM.

* * *

Operating s/m: o/s is collection of routines that tells the computer what to do & how to do. O/s routines assign computer resources to individual appl. prg. It assign memory to program and data files and move data to and from memory and also handle I/o operations.

Interaction b/w o/s & appl. prg: o/s:



Step:1: os routine load application prg. from disk to memory.

Step:2: Execute the appl. prg until any resource required for further execution.

Step:3: o/s routines load the necessary resource into memory from disk.

Step:4: complete the execution of full appl. prg.

Step:5: o/s routine send the request to printer to print the result of application prg.

This type of concurrent execution is called multiprogram (or) multitasking.

1.3. TECHNOLOGY (OR) TECHNOLOGY FOR BUILDING x PROCESSOR:

(10) * Computer Architect must plan the design based on future technology. Changes

* processor and memory have improved at an incredible rate.
* From the time computer invention till this date there are 5 main technology introduced & used for building processor. Then are

year	technology used in computer	Relative performance / unit cost.
1942 - 1955	Vacuum tube	1
1955 - 1964	Transistor	35
1964 - 1975	Integrated circuit	900
1975 - present	processor - VLSI	2400 000
present - Future	AI - ULSI	6200 000 000

1.3.1. Vacuum tubes (First generation: 1942 - 1955):

* The first generation computer were using Vacuum tubes.
* Vacuum tube: is an electronic device made up of glass which has terminals used to transfer the data.
* The first electronic computer is ENIAC (Electronic Numerical Integrator and computer).
* It has nearly 18000 vacuum tubes & 1500 relays.
* It can able to perform 5000 addition / subtraction per second.
* But able to perform only decimal not on binary.
* It has a memory of 20 accumulator and can store 10 digit decimal number.

Advantage	Disadvantage
1. Vacuum tubes are only electronic component available on those days.	1. very large in Size.
2. Able to calculate data in millisecond.	2. produce large amounts of heat.
	3. Consume large amounts of power.
	4. used machine language only & used punch cards for input.
	6. magnetic drums were used for memory storage which has less storage capacity.

1.3.2. Transistor (Second Generation: 1955 - 1964):

* Transistor: is an ON/OFF Switch controlled by electrical Signal.
* Size of transistor were very small compared to vacuum tube.
* Hence tend to replace the vacuum tubes with transistor.

* with the help of transistor the computer able to perform both floating point & fixed point operations.

* Transistor mainly having 3 pins: 1. Collector 2. Emitter 3. base.
* Ex: IBm 1400 series.
* Advantage of transistor:
  1. Very small in size
  2. cheap in cost.
  3. less heat dissipation.
  4. Low power consumption.
  5. Greater Speed.
  6.

Advantage	Disadvantage
1. Smaller in size compared to first generation.	1. Cooling sys required.
2. more reliable than 1st generation.	2. maintanance required periodically.
3. Better portability due to small in size.	3. Only used for specific purpose.
4. Accuracy is improved.	4. Still using punched cards for I/p & print out for o/p.

1.3.3. Third Generation Computer : Integrated circuit : (1964-1975):

* 3rd generation Computer uses IC (Integrated circuit).
* IC contain hundred of transistor & hundreds of Capacitors.
* IC enabled lower cost, faster processor.
* Various techniques were introduced by using IC. Then are
  1. microprogramming 2. parallel processing 3. resource sharing.
  4. pipelining 5. multiprogramming.
* Ex: PDP-8.
* Because of IC no. of. I/O ports increased largely.

Advantage	Disadvantage
1. Small in size compared to the 2nd generation computer.	1. AC required.
2. more reliable 3. used less energy. 4. Better speed 5. High storage. 6. low cost compared to 2nd generation computer.	2. Highly sophisticated technology were required to manufacture IC.

1.3.4. Fourth generation: Microprocessor: (1974 - present):

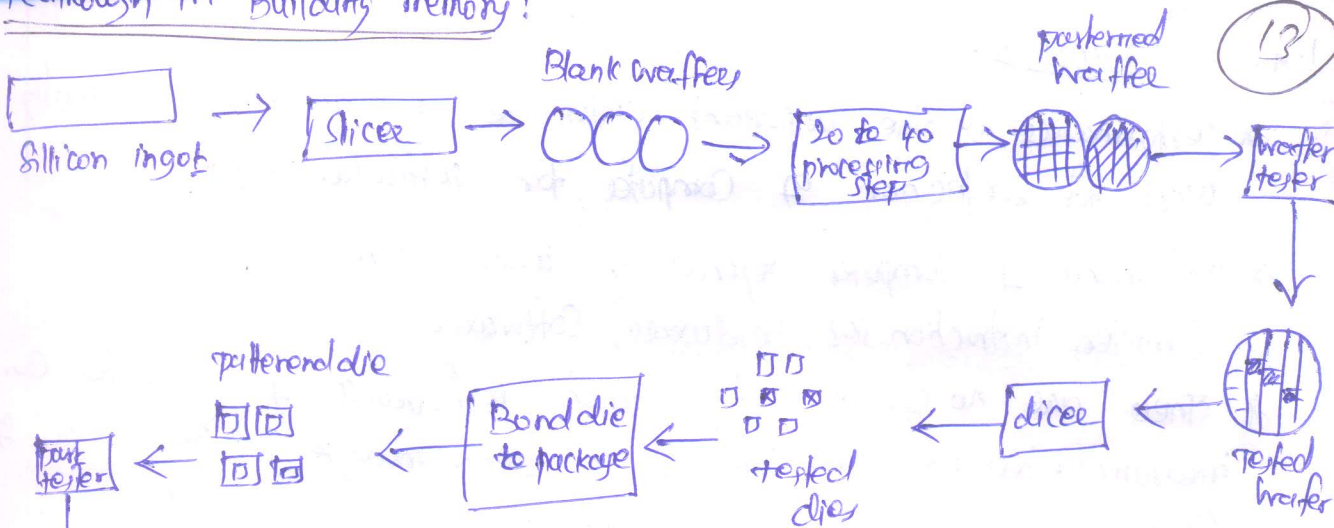
(12)
* Microprocessor were used in 4th generation computers.
* Microprocess contain thousands of transistor, resistor, capacitor in a single chip.
* VLSI (Very large Scale Integration) technology were used for manufacturing of microprocessors.
* Because of microprocessor the new technology were introduced. They are
  1. multiprocessing 2. pipelining 3. Virtual memory
* And also it lead to invention of desktop computer / laptops.

Advantage		Disadvantage
1. more powerfull	Compared to third generation computer.	1. technology was very poor on those days to prepare microprocessors.
2. more reliable		2.
3. Small in size		
4. Less power consumption		
5. Faster processing speed		
6. totally General purpose use.		

1.3.5. Fifth generation Computer: Artificial Intelligence: present - future:

* This computer are based on Artificial Intelligence.
* AI must support natural language processing, voice recognition, parallel processing, able to think itself & decide on what to do & how to do.
* ULSI (Ultra Large Scale Integration) technology it needed for manufacturing AI machines.
* It is process of integrating millions of transistor, resistors, capacitor, embedding them into single small chip.
* Still this type of computer in development stages.

Technology in Building memory!



dig: of: Chip manufacturing process

Explain each steps with 1-2 points. Highlight mainly Bonding & yield process.

Bonding: is a process connecting good dies with the I/p & o/p pins. Then these package/chip will be tested & if there is no mistake then shipped to customer.

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

$$\text{Dies/wafer} = \frac{\text{wafer area}}{\text{Die area}}$$

Bose Einstein formula $\Rightarrow$

$$\text{Die yield} = \frac{1}{\left[1 + (\text{Defects per area} \times \frac{\text{die area}}{2})\right]^{N}}$$

N may be b/w 11.5 to 13.5

Ex: Dig: Find die yield for die of 1.5 cm on side & 1.00 cm on a side assume defect density = 0.031 per cm^2 & N = 13.5.

Die yield = $\frac{1}{[1 + (0.031 \times 2.25)]^{13.5}} \approx 0.40$.

For 1.5 cm: $1.5 \times 1.5 = 2.25$

Die yield = $\frac{1}{[1 + [0.031 \times 1.00]]^{13.5}} \approx 0.66$

$\therefore$ Less than 50% of large dies are good But 2/3 of small die are good.

1.4. Performance:

* performance is the important attribute of computer. It mainly used for selection of computer for particular task.
* performance of computer depends on many things such as compiler, instruction set, hardware, software.
* There are no. of ways by which performance of computer can be measured. Here we will see how to define, measure the performance of computer & factor affecting performance.

1.4.1. Defining a performance:

* when we say one computer is faster than another we compare their speed and absorb that faster computer run a prg. in less time.
* when a computer center manager executing a large prg. on both system at a time. He will say faster one is finishing the task before the other one completes.
* computer user always interested in reducing the time between start and end of prg. i.e. reducing execution time.
* Execution time: is the time taken by computer to finish its task as soon as user given the I/p. It is also called Response time.
* Throughput: is total amount of work done in given time. the performance of computer is directly related to throughput.
* Response time: is the total time required for computer to complete its task including disk access, memory access, I/o activities, cpu execution time".
* Bandwidth: is also called throughput.

$$\therefore \text{performance}_A = \frac{1}{\text{Execution time}_A}$$

(or)

$$\text{performance}_B = \frac{1}{\text{Execution time}_B}$$

* Let us take Computer A is greater performance than Computer B

$$\text{performance}_A > \text{performance}_B \Rightarrow \frac{1}{\text{Execution time}_A} > \frac{1}{\text{Execution time}_B}$$

$\therefore$ Execution time$_B$ > Execution time$_A$.

* (ie) execution time of B is more than execution time of A.

* Now we will say that "Computer A is n time faster than Computer B".

(ie) $$\frac{\text{performance}_A}{\text{performance}_B} = n \Rightarrow \frac{\frac{1}{\text{Execution time}_A}}{\frac{1}{\text{Execution time}_B}} = n.$$

$$\boxed{\frac{\text{performance}_A}{\text{performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = n.} \Rightarrow \text{is equation of relative performance}$$

1.4.2. measuring performance:

* One important measure of computer performance is time.
* However time can be measured in different method.
* most used method is wall clock (or) response time (or) in seconds.

cpu time:

cpu time (or) cpu execution time: is time cpu spend for computing particular task. It doesn't include time spending on waiting for I/O or other prg.

clock cycles: are also called clocks, ticks, clock period, cycle. Length of each clock cycle is called clock period.

1.4.3. Basic cpu performance Equation:

* In order to find out cpu execution time we first need to find cpu execution time.
* So simple formula relating ref. no. of clock cycle for a program and clock cycle time. is:

(16) CPU Execution time for program = CPU clock cycles for prg × clock cycle time

we know that clock cycle time = $\frac{1}{\text{clock rate}}$

$$\therefore \left.\begin{array}{l}\text{CPU Execution time} \\ \text{for a prg.}\end{array}\right\} = \frac{\text{CPU clock cycle for prg}}{\text{Clock rate.}} \quad \rightarrow (1)$$

*The above formula clearly state that we can improve the performance of computer by reducing no. of. clock cycles required for prg.

$$\left.\begin{array}{l}\text{CPU clock cycle} \\ \text{for prg}\end{array}\right\} = \begin{array}{l}\text{No. of. Instruction} \\ \text{for prg}\end{array} \times \begin{array}{l}\text{Average clock cycles} \\ \text{per Instructions}\end{array}$$

$$\therefore \text{CPU clock cycle for prg} = \text{Instruction Count} \times \text{CPI} \quad \rightarrow (2)$$

where CPI → clock cycles per Instruction = $\frac{\text{CPU clock cycle}}{\text{Instruction Count}}$

* CPI: it avg. no. of. clock cycle per Instruction for prg. Different instruction may take different clock cycles.

∴ put eqn (2) in (1).

1.4.4. Classical CPU performance Equation:

* eqn. (1) can be rewritten as

$$\left.\begin{array}{l}\text{CPU execution time} \\ \text{for a prg}\end{array}\right\} = \frac{\text{Instruction Count} \times \text{CPI}}{\text{clock rate.}} \quad \rightarrow (3)$$

For n no. of. programs.

$$\text{CPU Execution time} = \sum_{i=1}^{n} \frac{\text{Instruction Count}_i \times \text{CPI}_i}{\text{clock rate}_i}$$

The above formula (3) can be used for comparision of performance of 2 computers or designs.

Factors affecting performance is:

1. Instruction Count.
2. CPI:
3. Clock rate:

Problems based on performance :

Model: 1: If Computer A runs a prg in 10 seconds & computer B runs the same prg in 25 seconds. how much A is faster than B?

Given: Execution time$_A$ = 10, Execution time$_B$ = 25.

w.k.t $\Rightarrow$ $\frac{\text{performance A}}{\text{performance B}} = \frac{\text{execution time B}}{\text{execution time A}} = n$

$\therefore n = \frac{25}{10} = 2.5$.

$\therefore$ Computer A is 2.5 times faster than Computer B.

Model: 2: Computer A runs a prg in 12 sec with 3 GHz Clock. we have to design a new computer B such that it also runs the same prg within 9 sec. Determine clock rate for B? Assume that due to increase in clock rate cpu design of computer B is affected & it requires 1.2 time as many clocks as computer A for execution of prg? Find the target clock rate of B?

Given: Cpu time$_A$ = 12 Cpu time$_B$ = 9.

Clock rate A = 3 GHz = 3×10^9 cycle/sec. Clock rate$_B$ = ? cycle/sec.

w.k.t $\Rightarrow$ $\text{Cpu time}_A = \frac{\text{Cpu clock cycle}_A}{\text{clock rate}_A} \Rightarrow 12 \text{ sec} = \frac{\text{cpu clock cycle}_A}{3 \times 10^9 \text{ cycle/sec}}$

$\therefore \text{Cpu clock cycle}_A = 12 \text{ sec} \times 3 \times 10^9 \frac{\text{cycle}}{\text{sec}} \Rightarrow 36 \times 10^9$ cycles.

$\text{Cpu time}_B = \frac{\text{Cpu clock cycle}_B}{\text{Clock rate}_B} \Rightarrow \text{Clock rate}_B = \frac{\text{Cpu clock cycle}_B}{\text{cpu time}_B} = \frac{1.2 \times \text{Cpu clock cycle}_A}{9 \text{ sec}}$

$\text{Clock rate}_B \Rightarrow \frac{1.2 \times 36 \times 10^9 \text{ cycles}}{9 \text{ sec}} = 4.8 \times 10^9 \frac{\text{cycle}}{\text{sec}} = 4.8 \text{ GHz}$.

$\therefore$ Clock rate$_B$ = 4.8 GHz

Model: 3: Let us assume that two computers use same instruction set architecture Computer A has clock cycle of 2 ops & CPI of 2.0 for some prg Computer B take clock cycle of 1 ops & CPI of 1.2. Which computer is faster & how much times?

Given: Computer A clock cycle time = 250 ps $CPI_A = 2.0$,
Computer B clock cycle time = 500 ps. $CPI_B = 1.2$.

(18)

∴ w.k.t ⇒ CPU time$_A$ = CPU clock cycle$_A$ × Clock cycle time$_A$

= 250 ps × 2.0 × 1 = 1500 ps

where CPU clock cycle = CPI × IC.

CPU time$_B$ = CPU Clock Cycle$_B$ × Clock cycle time$_B$

CPU time$_B$ = 500 × 1.2 × 1 = 600 ps.

∴ $\frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{600 \text{ ps}}{500 \text{ ps}} = 1.2 = \frac{\text{performance}_A}{\text{performance}_B}$

∴ Computer A is 1.2 times faster than Computer B.

model:4: Below table shows that 2 code sequence with no. of instruction of different classes with each code sequence for class A, B, C. Table B) Show CPI for each class. find

(i) which code sequence will execute more instruction.
(ii) which code will execute quickly.
(iii) CPI for each code sequence.

Code sequence	Instruction Count		
	A	B	C
1	4	2	4
2	8	2	2

	CPI for each class		
	A	B	C
CPI	1	2	3

Sol:1: To find more no. of instruction:

∴ Code sequence 1 = 4 + 2 + 4 = 10 instruction.
Code sequence 2 = 8 + 2 + 2 = 12 instruction.

Sol:2: CPU clock cycle code 1 = ((4×1) + (2×2) + (4×3)) = 20 cycle.
CPU clock cycle code 2 = ((8×1) + (2×2) + (2×3)) = 18 cycle.

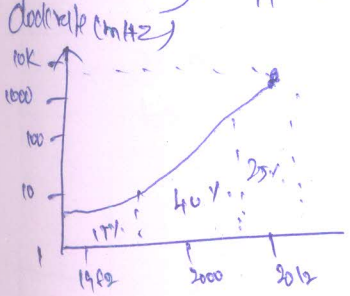
Sol:3: w.k.t ⇒ $CPI = \frac{\text{CPU clock cycle}}{\text{Instruction count}}$ ⇒

$CPI_1 = \frac{20}{10} = 2.0$ $CPI_2 = \frac{18}{12} = 1.5$.

Answer:
(i) Code sequence 2 will execute more no. of instructions.
(ii) Code sequence 2 will execute quickly even though it execute more no. of instruction.
(iii) $CPI_1 = 2.0$ $CPI_2 = 1.5$.

1.5. POWER WALL:

* Today power is the biggest challenge facing the computer designer. Because first power has to bring/brought in and distribute around the chip.
* modern microprocessor/IC chips having hundreds of pins for power.
* power dissipated as a heat is the second problem that heat must be removed.
* As a designer we have to know answers for mainly three questions.
  1. what is the maximum power ever required for processor?
  2. what is the sustained thermal design power (TDP)?
  3. what is metric used for measuring energy or power for processor?
* Running a processor at high clock speed allows better performance. But it produce lot of heat & consume more power also.
* For this computer architect mainly design power system with modern technology. Then are some solution for problem of power consumption:
* 1. Do Nothing well: when the s/m is in idle mode the power which using by the system will be automatically curtted off.
* 2. Dynamic voltage-Frequency Scalling (DVFS): whenever the s/m is having low work load it doesn't operate at high clock. So simultaneously the s/m will consume less power parallely produce less heat.
* 3. Design for typical case: Based on the use/mode we can design the power consumption of the device.
* 4. Over clocking: it a technology allowing system to run high speed/clock rate for shorter time by switching off other working application/system.



* dynamic energy required cpu is:

$$\text{Energy}_{\text{dynamic}} \propto \frac{\text{Capacitive load} \times \text{voltage}^2}{2}$$

* power dynamic required by cpu is

$$\text{Power}_{\text{dynamic}} \propto \left(\frac{\text{Capacitive load} \times \text{voltage}^2 \times \text{Frequency}}{2}\right)$$

* The above Equation for half of cycle. for one full cycle

(20) * power consumed by cpu $\Rightarrow P = CV^2f$

where P = power. C = capacitive load V = voltage f = frequency.

* The above Equation clearly states the power required by cpu.

problem based on power:

model:1: If a new processor has 85% of capacitive load of old processor. and voltage supply also reduced by 20% & 25% shrink in frequency. Find the impact on dynamic power consumption

w.k.t = $P = CV^2f$

$$\therefore \frac{\text{Power}_{\text{new processor}}}{\text{Power}_{\text{old processor}}} = \frac{(C \times 85\%) \times (V \times 0.8)^2 \times (f \times 0.75)}{CV^2f} = \frac{0.85 \times C \times 0.8^2 \times V^2 \times 0.75 \times f}{CV^2f}$$

$\therefore \frac{\text{Power}_{\text{new}}}{\text{Power old}} \Rightarrow 0.85 \times (0.8)^2 \times 0.75 = 0.408$

$\therefore$ New processor use 40% of power that old processor uses.

1.6. UNIPROCESSOR TO MULTIPROCESSOR:

* Because of limitation forced by the power consumption there is a change in the design approach of microprocessor. Rather than continuing to decrease the response time of prg running in single processor.
* designer came up with multiple processor per chip. The main motive is to increase the throughput rather to decrease response time.
* To reduce the confusion between the word processor and micro processor we use the latest name called "MULTI CORE MICROPROCESSOR".

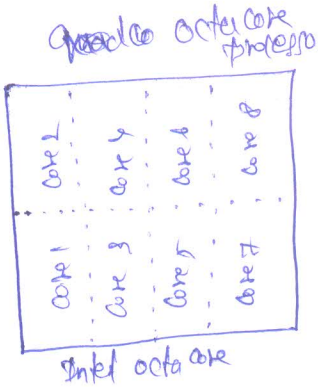
* Ex: Intel dual core microprocessor chip contain only 2 processor but intel quad core microprocessor chip contain 4 processor or 4 cores.

* In previous days, programmers rely on hardware, architecture and compilers to double their performance. (2)

* But nowadays programmer itself re-writting a prg. that will take advantage of parallelism & pipelining & load balancing.

* But writing a such a prg with above said all aspect is difficult. Because

1. It increase difficulty level of programming.
2. Prg. needs to be synchronized with eachother.
2. It is necessary that prg. needs to be divide so that each processor will get equal load balance.
4. Need to maintain co-ordination b/w parallelism & pipelining.



* Since the above dig. shows that single processor (looking at outside) contain 8 equal processor.

* So that if we give a big task to that processor it will split and work on the task parallelly and give the result within the short time.

* So the throughput of the s/m will automatically increase.

Advantage of multiprocessor:

1. High throughput 2. Reduced response time 3. Low power consumption.
4. Reduced cost 5. High Reliability.

NOTE: (Explain each point with 2 points).

Difference b/w uniprocessor & multiprocessor.

UNIPROCESSOR	MULTIPROCESSOR
1. Contain single processing unit	It contain 2 or more processing unit.
2. Cost is less.	Cost is high.
3. Task cannot be divide	Task can be divided
4. Used for office/ home purpose	Used for large organization.

1.7. Instructions:

* To command a computer hardware you must speak its language. The words of computer language is called Instructions. and its vocabulary is called Instruction set.

(22)

* Instruction set will describe the functions of architecture so every computer architecture must know the instruction set.
* Instruction set come from mips technology and mainly 3 popular instruction sets.
  1. ARM V7: Similar to mips. more than 9 billion chips were manufactured in 2011 making most popular instruction set in world.
  2. Intel x86: powers both pc and cloud of post pc era.
  3. ARM V8: it extended addressing size of ARM V7 32 to 64 bits.

Elements of Instructions:

1. opcode: Specify the operation to be performed. This operation is specified in binary code. Hence name operation code (or) opcode.
2. Source/Destination operand: It specify the source/destination operand for instruction.
3. Source operand address: Specify the requirement of one or more source operand.
4. Destination operand address: operation executed by the cpu will be stored producing result. That will be stored in destination operand.
5. Next instruction: Specify and tells the cpu from where the instruction for of next operation need to fetch after complete of current instruction.

Representation of instruction:

Opcode	Operand address 1 / variable	Operand address 2 / variable

EX:

Add a, b, c

Instruction types based on no. of address: (or) types of Instruction format:

1. 0(zero)-operand instruction:
   * This instruction don't have address fields.
   * They are called zero address instruction (or) stack instructions.
   * They also don't have Source/destination address.
   * It also called pushdown stack.

* Ex: To perform C = A+B :

PUSH A ; PUSH B ; ADD ; POP C

* Syntax : Stack_operation variable;

* Advantage: 1. Simple model for evaluation.
2. Instruction length are short.

* Disadvantage: 1. Stack can't be randomly accessed.
2. One stack is busy for every operation it creates a bottle neck.

2. 1 (One)-operand Instruction:

* This instruction contain one address field.
* Then are called one address instruction or Accumulator instruction.
* Syntax : operation destination_variable ;
* Ex : C = A + B can be done as

LOAD A ; ADD B ; STORE C :

* Advantage: 1. Instructions are very short.
* disadvantage : 1. Since Accumulator is the only storage place so memory traffic is high as possible.

3. 2 (two)-operand Instruction:

* This instruction mainly contain 2 fields namely source, destination fields.
* Each field specify either processor register or memory.
* Then are called general purpose register instructions.
* Syntax: Opcode destination , Source;
* Ex: To perform C = A + B;

LOAD R1, A ; LOAD R2, B ; ADD R1, R2, STORE C, R1.

* Advantage: 1. make coding very easy.
* disadvantage: 1. All operand must be named so instruction length is long.

3. 3 (three)-operand Instruction:

* This operation instruction contain 3 address field.
* This is also called general purpose register instruction.
* Syntax: opcode Source1, Source2, Destination;
* Ex: C = A + B

ADD A, B, C (or) LOAD R1, A; LOAD R2, B; ADD R1, R2, R3, STORE C, R3;

* Advantage: 1. coding makes easy.
* Disadvantage: 1. Instruction length is large due to each operand must be named.

Problem: write a prg to evaluate Y = (A+B) * (C+D) using 3, 2, 1, 0 address instruction.

1) using 3 instruction:

```
ADD   R1, A, B;
ADD   R2, C, D;
MUL   Y, R1, R2;
```

2) using 2 instruction;

```
MOV   R1, A;
ADD   R1, B;
MOV   R2, C;
ADD   R2, D;
MUL   R1, R2;
STOR  Y, R1;
```

3. using 1 instruction;

```
LOAD  A;
ADD   B
STORE T;
LOAD  C;
ADD   D;
MUL   T;
STORE Y;
```

4. using 0 Instruction:

```
PUSH A;
PUSH B;
ADD;
PUSH C;
PUSH D;
ADD;
MUL;
POP Y;
```

1.8. operations of computer Hardware:

* Every computer must perform some different kinds of operations based on application areas.
* Some computers perform simple operation set some will perform complex operation based on instruction set.
* To perform fundamental arithmetic operations programming language need certain instructions. The MIPS assembly language notation for performing addition operation is

```
add a, b, c
```

* The above instruction will instruct to computer to perform addition on b & c and put the sum result in a.
* Suppose if need to add 4 variables b, c, d, e then

```
add a, b, c; add a, a, d; add a, a, e;
```

1.8.1. Types of operation:

Category	Instruction	EX.	meaning	
Arithmetic	Add	add a, b, c	a = b + c	
	Subtract	Sub a, b, c	a = b - c	
	add immediate	addi a, b, 20	a = b + 20	
Data transfer	Load word	lw a, 20(b)	a = memory [b+20]	word from memory to register
	Store word	sw a, 20(b)	memory (b+20) = a	word from register to memory
	move word	mv a, b	a = a b.	move word from Sto a
	Exchange	Exc a, b	Acc = a; a = b; b = Acc	Swap content of source & destination
	push	PUSH A		Transfer word from source to top of stack.
	pop	POP A		Transfer word from top of stack to destination.
	Set	Set A		Set 1 to destination
	clear	Clear A		Set 0 to destination.

Category	Instruction	Example	meaning	Comments
Logical	and	and a,b,c	a = b & c	
	or	or a,b,c	a = b | c	
	NOR	Nor a,b,c	a = ~(b | c)	
	shift left logical	Sll a,b,c	a = b << 10	
	shift right logical	Srl a,b,c	a = b >> 10	
	Compare	comp a,b,10	a = = 10	make comparison & 2 or more operand need space for flag.
	Test			
	Ex OR	XOR a,b,c	a = b ^ c	
Conditional	Branch on equal	beq a,b,25	if (a = = b) go to 25 stmt line	
	Branch on not equal	bne a,b,25	if (a != b) goto 25 stmt line	
unconditional (or)	Jump	J 2500	goto 2500 line	
	Jump register	Jr a	goto a;	
	Jump & link	jal 2500	go to 2500 line & link.	
transfer control	Return	I/O operation	Input (read)	
	Execute		output (write)	
	Skip		Start I/O	
	Halt		test I/O	
	Wait			

* To keep the hardware simple every instruction has to have exactly three operand, no more & no less.

* Hardware technology has 3 design principle; that will give detailed information about fixed & variable no. of. operand & operation.

Design principle: 1: Simplicity favour regularity.

Design principle: 2: Smaller is faster.

Design principle: 3: Good design demand good compromises.

1.9 Operands of computer Hardware:

* operand is a variable used to perform any kind of operations on it.
* Usage of operand is very S/W one language to another language.
* Operands must be from a limited no. of special locations built directly in hardware called registers.
* mips architecture has 32 bit register and group of 32 bit are called word.
* word: is the natural unit of access in a computer usually a group of 32 bits. Size of a register in mips architecture is called word.

D/S/W Variable & register:

	Variable	Register.
1.	Holds a value that can change depending on condition or an info. passed into program.	It is very small amount of memory but very fast that is built in cpu in order to speedup the operation.
2.	Large no. of variables are there	Limited no. of registers.
3.	Low speed to access data	High speed access of data.

* The main reason for having only 32 register is due to design principle of 2 (i.e smaller is faster).
* Register are the fastest place to hold / access data in computer.

1.9.1. Memory operand:

* prog. languages have simple variables that contains single data elements.
* But variables having more complex data structure like arrays and structure have large amount of elements.
* In that time computer cannot hold / access such a large structures.
* To access / store the large structure memory operands are needed.
* processor can keep only a small amount of data in memory but memory can store billions of data elements.
* In mips architecture all arithmetic operations are carried out only on register & also for data transfer S/W memory.

(28) Data transfer instructions:

* It is a command that move data b/w memory & registers.

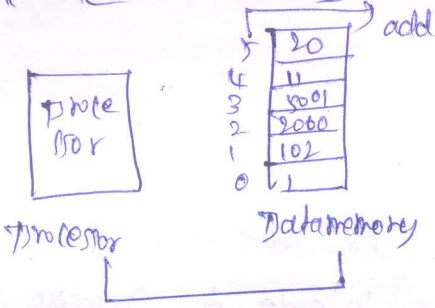
Address:

* To access a word in memory we need memory address.
* Address is a value used to specify the location of a specific data element within a memory array.

memory:

* It is a huge, large dimensional array with address acting as the index to the array starting with 0.

Ex:



The address of third element is 2. & value of memory(2) = 2000.

Alignment restriction:

* In MIPS, words must start at address 0 and the continuous words will be in multiples of 4. This requirement is called alignment restriction.

Big Endian & Little Endian:

* 8 bit bytes are divided into two parts.
* Big end: Address of the left most bytes.
* Little end: Address of the right most bytes.

D/S/W memory & Registers

	memory	Registers
1.	memory speed is slower than registers.	Register is faster than memory.
2.	It takes more time for data access.	It takes less time.
3.	less throughput	more throughput.
4.	Using a memory will take many cycles.	Less cycles required.

1.9.2 Constant or Immediate operands:

* Constant variables are used as one of the operand for many arithmetic operation in mips architecture.
* Constant variables are placed in memory when they is loaded.
* The quick add instruction with one constant operand is called add arithmetic (or) addi
* Ex: addi a, b, 4 = a = b+4;
* Advantage: 1. It use less energy 2. perform operation more fast.

1.10. Representing Instruction in Computer:

* An instruction is an order given to computer processor by a computer program.
* At lowest level each instruction is a sequence of 0's & 1's that describe the physical operation.
* Registers are used for carrying data that used for execution.
* Instructions are kept in computer as series of High & Low electronic signals.
* Each instruction are used to perform some task and representation of instruction may vary from one language to another.

Instruction format: is a form of representation of an instruction composed of fields of binary numbers.

Registers: In computer hardware register are referred as instructions.

* In mips assembly language, register $s0 to $s7 is mapped to 16 to 23. and register $t0 to $t7 is mapped to register 8 to 15.
* 111th $s0 = 16, ...

machine language:

* In mips all instruction are 32 bit long. So the machine language is binary representation used for communication within computer.
* Instructions used in machine language is called machine code.

Ex: add $t0, $s1, $s2;

30

0	17	18	8	0	32

→ decimal format.

* First & last field (0 & 32) tell mips instruction for addition.
* Second field (17) = $s1. 3rd field 18 = $s2.
* fourth field contain no. of register to receive sum (8 = $t0)
* $s1 added with $s2 and placed the sum in $t0.

000000	10001	10010	01000	00000	100000
6bit	5	5	5	5	6

Hexadecimal Number:

* In computer we can use binary number to read/write data. In that even small no. also require large amount of bits.
* So we can use Higher base for easy conversion. ∴ Hexadecimal will use.
* Base value of Hexadecimal is 16 & its power is 2.
* To avoid confusion b/w various numbers we can use subscript values.

mips Fields:

mips fields has two kinds of format such as

1. R type (or) R-format (for register).
2. I type (or) I-format (for Immediate).

1. R-Format:

op	rs	rt	rd	shamt	funct
6bit	5	5	5	5	6bit.

* op = opcode denote the operation need to be performed.
* rs = first register source operand.
* rt = second register source operand.
* rd = destination register.
* shamt = shift amount.
* funct = function code select specific variant of operation in op field.

2. I format:

op	rs	rt	constant/address.

used for Immediate

add :	R :	0	reg1	reg2	reg3	0	32	n.a
add I :	I :	8	reg1	reg2	reg3	n.a	n.a	constant

1.11. Logical operations:

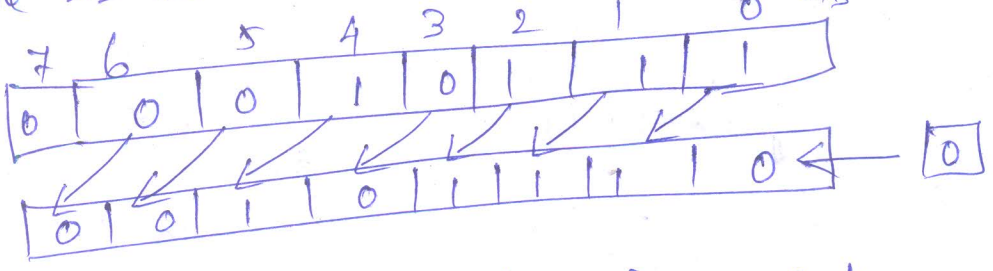
* Initially computer preform their operations on full words.
* After that it is capable of performing operation on fields of bits by bit within a word.
* Using single bit of information some operation can be executed in computer called logical operations.

def of logical operation: It is an operation/instruction in which the quantity being operated on bit by bit and the result of the operation will have either 0 or 1 only two values.

S.No	Logical operand	operator	Java operator	MIPS
1	Shift left	<<	<<	SLL
2	Shift right	>>	>>	SRL
3	Bit by Bit AND	&	&	and, andi
4	Bit by Bit OR	|	|	or, ori
5	Bit by Bit NOT	~	~	not

1. Shift left logical (SLL):

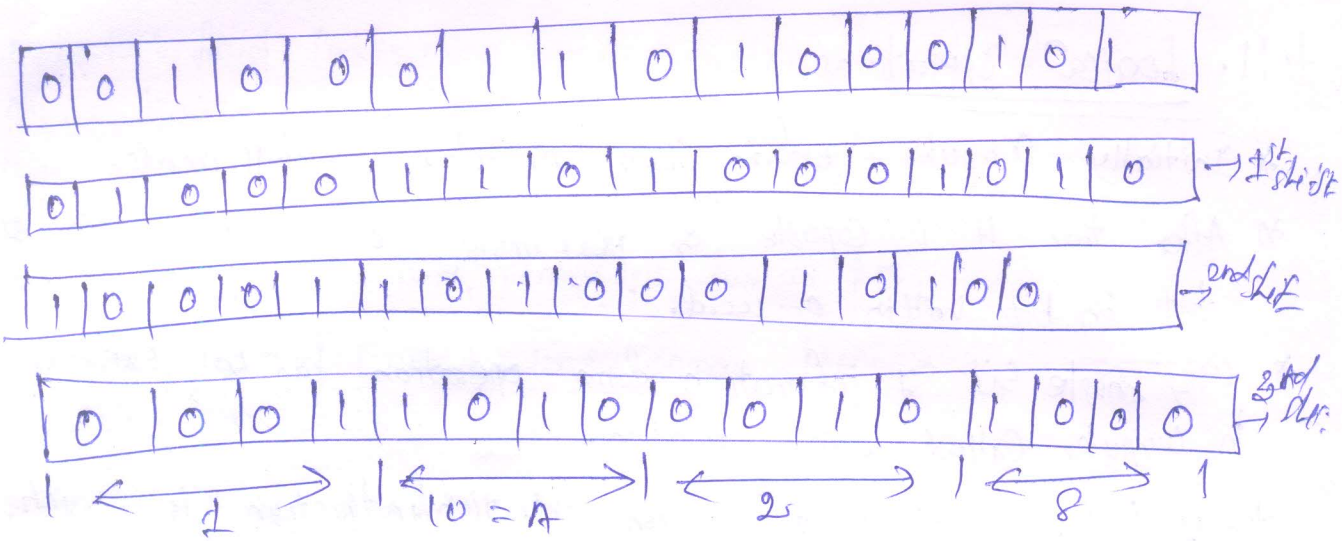
* It moves all the bits in a word to the left side by one bit and empty position is filled with 0.
* Ex:



* Try your self for 2345 - shift left by 3 times.
* Ex: SLL R1, R2, 10 = R1 = R2 << 10

First write it equivalent binary no.

(32) 2 → 0010 , 3 → 0011 , 4 → 0100 , 5 → 0101



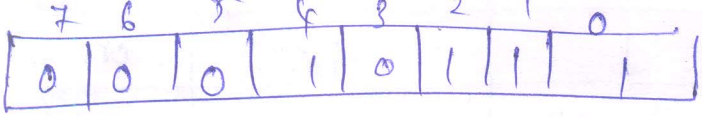
∴ $(2345)_{10}$ after 3 shift left = A28.

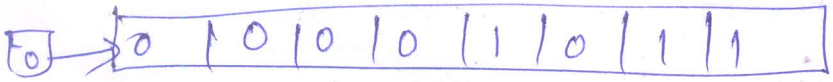
Now do it for shift right logically:

2. Shift right logically:

* It moves all the bits in a word to right side by a bit and empty position is filled with 0.

* Ex:





* Try (2345) 3 bit right shift = 046B.

* Ex: SRL R1, R2, 10 = R1 = R2 >> 10;

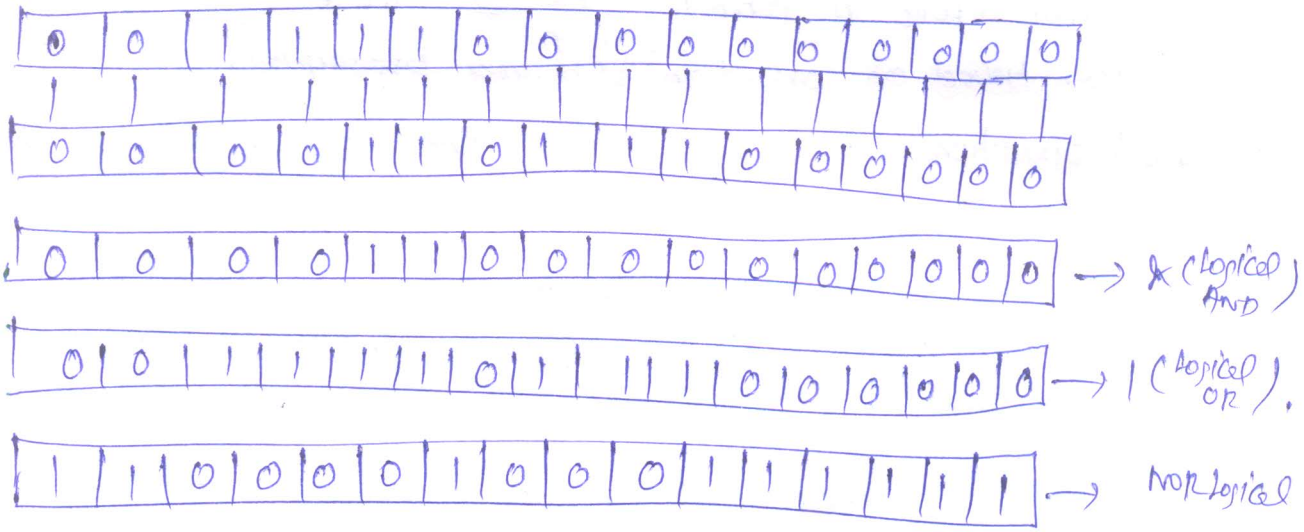
3. Logical AND operation:

Truth table			OR	NOT	EXOR	NOR
0	0	0	0	0	0	1
0	1	0	1	1	1	0
1	0	0	1		1	0
1	1	1	1		0	0

Logical AND operation is a bit by bit operation with two operands that calculate the 1 only if there is a 1 in both operands.

Ex: 0011 1100 0000 0000 → a
0000 1101 1100 0000 → b.

* perform and c, a, b;

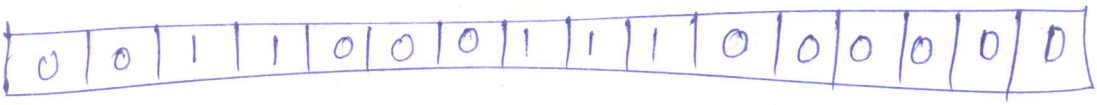


4. Logical OR: or c, a, b;

5. Logical NOR: NOR is negotiation of $\overline{OR}$.
This can be done in 2 method: NOR c, a, b;
method: 1: First find OR and then take NOT.
method: 2: Use Truth table for NOR (00 → 1; other → 0):

6. Logical NOT (~):
* Here replace 0 → 1 & 1 → 0.
Ex: find NOT for: 0010 1010 1100 1000
(~) NOT = 1101 0101 0011 0111

7. Logical XOR (^): for a &b.



1.12. Control operations (or) Decision making Instructions:

* many things can be used to make distinguish between a computer and simple calculator.
* The main aspect is computer able to make decision. It can be taken based on operation performed by computer.
* Control operations are used to control the flow of execution using some decision control method.
* Decision making is commonly used in programming language to indicate that based on condition what operation the computer has to execute.
* mips assembly language mainly include two decision making instruction 1. Beq (Branch if equal).
  2. Bne (Branch not Equal).
* 1. BEQ: Branch if equal

  Format: beq register 1, register 2, L1;

  If the value in register 1 is equal to value in register 2 then go to statement labeled L1.
* 2. Bne: Branch not equal:

  Format: Bne register 1, register 2, L2;

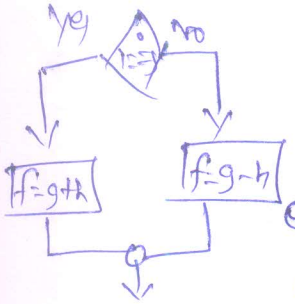
  If the value in register 1 is not equal to value in register 2 then go to statement labeled L2.

1.12.1. Conditional Branches:

* In mips language beq, bne are the conditional branches.
* def. of. Conditional Branches: is an instruction that requires comparision of two values and that allows for a subsequent transfer of control to a new address in program based on the out come of comparision.
* Ex:

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

Find the corresponding mips code for above c statements?



1. For first condition instruction beq should be
2. for second instruction we have to use bne.

```
        bne i,j, else ; // if i≠j then go to else
        add f, g, h ;  // f = g+h.
        j exit;
else: sub f, g, h;
exit;
```

1.12.2. Loops:

* Looping statements are mainly used to execute certain instruction more than one time untill some condition gets failed.
* for looping some decision need to be perform to obtain the task.
* for ex: Look below c program write equivalent mips code.

```
while (a[i] == k)
{
  i = i+1;
}
```

1. 1st step load a[i] in to temporary memory.
2. Before loading to a[i] we need to know address.
3. have to address and multiply it by 4 for each loop.

```
Loop: sll t, i, 2      → temp.
      add i, i, t      → base.
      lw t, 0(t)
      bne t, j, exit;
      addi i, i, 1;
      j loop;
exit;
```

Here first use loop, for temporary & i value to store & more four address for each loop.

if bne t, j then, goto exit; else add i, i, 4;
add i, i then loop; else exit;

Basic Block: it a sequence of instruction without branches, except possibly at the end and without branch targets or branch labels, except possibly at beginning.

1.12.3. Comparison instruction:

* mips compiler use mainly 4 comparision instruction they are Equal, not equal, less than, greater than or equal.
* o/p of the comparison instruction are either 1 or 0.
* set less than : slt c, a, b; slti c, a, 10; c=1 if a<5;
* set greater than : sgt c, a, b; sgti c, a, 20; c=1 if a>5;

1.12.4. Switch / Case statement:

* most programming language having case / switch statement that allows the programmer to select one of many alternatives depending on single value.
* Switch statement can be apply using 2 ways:
  1. using chain if-then-else statement.
  2. using Jump address table:

Jump table: or Jump address table is a table of address of alternative instruction sequence. It is just array of words contain the address of corresponding label.

1.12.5. Unconditional Statement:

1. Jump (J):
2. Jump link (JL):
3. Jump register (JR):

1.13. Addressing & Addressing modes:

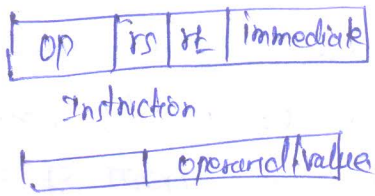
* def. of Addressing mode: is also called multiple form of addressing (or) addressing schemes.
* It is one of several addressing regimes delimited by their varied use of operand and/or address.
* def. of Effective Addressing: An address is computed by the processor when executing a memory access or branch instruction or when fetching next instruction is called effective Addressing (EA).
* Addressing mode specify how to calculate the effective address of operand by using elements of effective address.
* Element of effective address are 1. base 2. Index 3. displacement.

Types of addressing mode: register direct.

1. Immediate 2. Register (or) 3. Base or displacement.
4. pc relative 5. Absolute or direct 6. Indirect
7. Register Indirect. 8. Auto increment 9. Auto decrement
10. Pseudo direct.

1. Immediate addressing mode:

* Here the operand is given explicitly in instruction.
* In this constants are used as operand value.
* constants are short & fit within 16 bit field.
* EX: mov A, #20; (or) mov A, #20;
* This above instruction move or copy a value of 20 to register A.
* The #sign in front of value is indicating that value is an immediate operand.

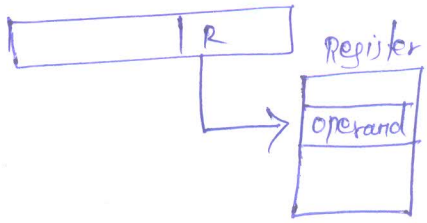


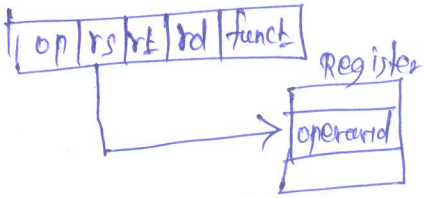
dig: for: Immediate: ↑

* * *

2. Register addressing mode:

* Here the operand is register.
* So we need to specify register name for operations.
* EX: mov R1, R2;
* The above instruction will copy the value of R2 into the R1 register.
* Either compiler or assembler must break large constants into pieces and reassemble them in register.



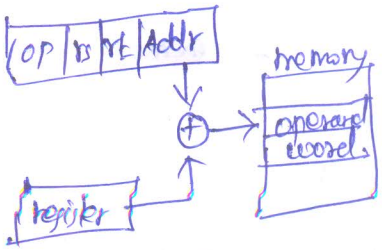


dig: for: register: ↑

* * *

3. Base or displacement addressing mode:

* Here the operand is at memory location whose address is the sum of register and a constant in instruction.
* EX: mov A, 2000;
* The above instruction will copy the content which present in memory address location 2000.
* So instead of operand we are mentioning memory address. of the variable is used.
* By using this we can easily calculate EA (effective address) with the help of (1) Base register (2) Indexing (3) displacement.
* EA = address/value + R;

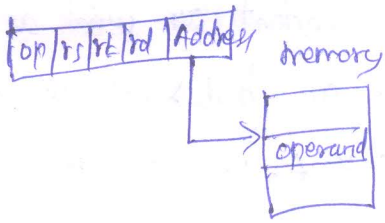


dig for: base or displacement ↑

* * *

4. PC-relative address:

* Here the referenced register is program counter.
* hence this addressing mode is called PC-relative.
* The EA (effective address) can be calculated by adding contents of PC to address.
* EA = PC + Address of instruction.
* EX: beq a, 0, again;

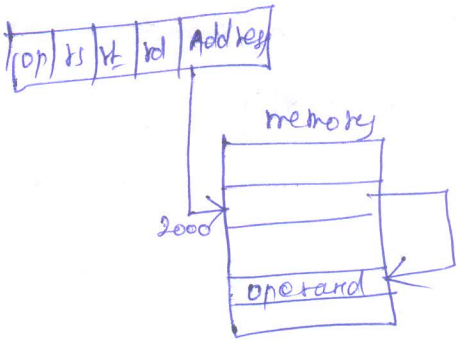


dig: for absolute (or) direct mode ↑

* * *

5. Absolute (or) direct addressing mode:

* The address of location of operand is given explicitly as a part of instruction.
* EX: mov A, 2000 :
* The above instruction will copy the contents that is present in 2000 memory location.
* The address of operand is given explicitly in the instruction.



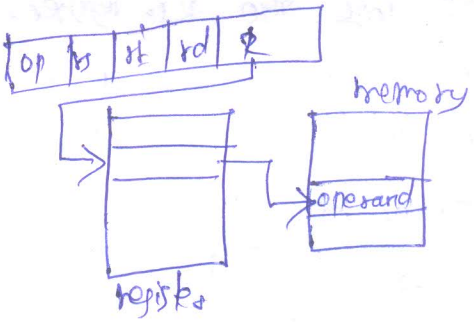
dig: for indirect mode ↑

* * *

6. Indirect addressing mode:

* In this mode, the instruction contain the address of memory which refers to address of operand.
* EX: mov 2000; (or) ~~load~~ ~~address~~.

  2000 → 2010;



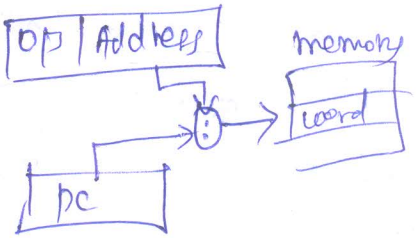
dig for register Indirect mode ↑

* * *

* The above will search for address 2000 that address 2000 will points to the operand (A) which is in another memory location.

7. Register Indirect mode:

* Here ~~content~~ effective address of the operand is content of register or main memory location whose address is given explicitly in the instruction.
* EX: mov A, (R0)
* This instruction copy the contents of memory address in the contents of register R0 into A.



dig for pseudodirect mode: ↑

* * *

8. Auto increment:

* The effective address of operand is contents of register specified in the instruction.
* After accessing the operand the contents of register are incremented to address the next location.
* EX: mov (R2)+, R0;
* The above instruction copy the contents of register R0 to register R2.
* After copy the content of R2 will be automatically incremented by 1.

9. Auto decrement:

* The effective address of operand is content of register specified in the instruction minus one.
* After accessing the operand the content of register are decremented to the address of next location.
* EX: mov (R2)−, R0
* The above instruction copy the contents of register R0 to register R2.
* After copy the content of R2 will be automatically decremented by 1

10. Pseudo direct addressing mode:

* Here the instruction like Jump & branch equall, branch not equall, branch on less than were used to Jump to specific location without any reason.
* So the pseudodirect addressing mode mainly helpfull when we use conditional statement/operation in our instructions.
* EX: BLE a, I, 1000;
* The above equation check if value of a is less than I then branch will go to address 1000. without any instruction.
* Hence it is called pseudodirect addressing mode.

Assignment: Carry Look AHEAD ADDER (or) FAST ADDER.

University Questions [8/16 marks]

1. Addressing modes with ex: [Dec 11/may 12/ Dec 12/ may 14/ may 15/ Dec 15/ Dec 16.
2. components of computer s/m: [Dec 14/ Dec 15/ Dec 16.
3. cpu performance equ: [Dec 14/
4. various techniques to represent instruction [may 15/