UNIT-IOverview and Instructions
PART-A

1. **What are the eight ideas in computer architecture?**
Design for Moore’s Law
Use abstraction to simplify design
Make the common case fast
Performance via Parallelism
Performance via Pipelining
Performance via Prediction
Hierarchy of Memory
Dependability via Redundancy
2. **Define power wall.**
Old conventional wisdom:
Power is free
Transistors are expensive
New conventional wisdom: “Power wall”
Power expensive
Transistors“free” (Can put more on chip than can afford to turn on)
3. **What is uniprocessor?**
A **uniprocessor system** is defined as a computer system that has a single central processing unit that is used to execute computer tasks. As more and more modern software is able to make use of multiprocessing architectures, such as SMP and MPP, the term *uniprocessor* is therefore used to distinguish the class of computers where all processing tasks share a single CPU. Most desktop computers are now shipped with multiprocessing architectures
4. **What is multicore processor?**
A multi-core processor is a single computing component with two or more independent actual central processing units (called "cores"), which are the units that read and execute program instructions.[1] The instructions are ordinary CPU instructions such as add, move data, and branch, but the multiple cores can run multiple instructions at the same time, increasing overall speed for programs amenable to parallel computing.
5. **Write the basic functional units of computer?**
The basic functional units of a computer are input unit, output unit, memory unit, ALU unit and control unit.
6. **Define multiprocessing?**
Multiprocessing is the ability of an operating system to support more than one process at the same time.
7. **What is a super computer?**
A computer with high computational speed, very large memory and expive parallel structured hardware is known as a super computer.EX: CDC 6600
8. **What is mainframe computer?**
It is the large computer system containing thousands of IC’s. It is a room- sized machine placed in special computer centers and not directly accessible to average users. It serves as a central computing facility for an organization such as university, factory or bank.
9. **What is minicomputer?**
Minicomputers are small and low cost computers are characterized by
Short word size i.e. CPU word sizes of 8 or 16 bits.
Limited hardware and software facilities.
Physically smaller in size.
10. **Define microcomputer.**
Microcomputer is a smaller, slower and cheaper computer packing all the electronics of the computer in to a handful of IC’s, including the CPU and memory and IO chips.
11. **What is workstation?**
The more powerful desktop computers intended for scientific and engineering applications are referred as workstations.

12. **What is instruction register?**
    The instruction register (IR) holds the instruction that is currently being executed. Its Output is available to the control circuits which generate the timing signals that control the various processing elements involved in executing the instruction.
13. **What is program counter?**
    The program counter (PC) keeps track of the execution of a program. It contains the memory address of the next instruction to be fetched and executed.
14. **What is processor time?**
    The sum of the periods during which the processor is active is called the processor time.
15. **What are clock and clock cycles?**
    The timing signals that control the processor circuits are called as clocks. The clock defines regular time intervals called clock cycles.
16. **Give the CPU performance equation.**
    CPU execution time for a program = CPU clock cycle for a program **X** clock cycle time
17. **What is superscalar execution?**
    In this type of execution, multiple functional units are used to create parallel paths through which different instructions can be executed in parallel. so it is possible to start the execution of several instructions in every clock cycle. This mode of operation is called superscalar execution.
18. **What is RISC and CISC?**
    The processors with simple instructions are called as Reduced Instruction Set Computers(RISC). The processors with more complex instructions are called as Complex Instruction Set Computers (CISC).
19. **List out the methods used to improve system performance.**
    The methods used to improve system performance are

    Processor clock

    Basic Performance Equation

    Pipelining

    Clock rate

    Instruction set

    Compiler
20. **What is the assembly language notation? Give example.**
    To represent machine instructions and program. assembly Language format is used. For example: The statement specifies an instruction that causes the transfer described below, from memory location LOC to processor registerR1. Move LOC, R1. The contents of LOC are unchanged by the execution of this instruction, but the old contents of register R1 are overwritten.
21. **Define addressing modes and its various types.**
    The different ways in which the location of a operand is specified in an instruction is referred to as addressing modes:

    Register mode, absolute mode, immediate mode, indirect mode, index mode, relative mode, auto increment mode, auto decrement mode.
22. **Define register mode addressing.**
    In register mode addressing the operand is the contents of a process register. The name of the register is given in the instruction.
23. **Define absolute mode addressing.**
    In absolute mode addressing the operand is in a memory location. The address of this location is given explicitly in the instruction. This is also called direct mode addressing.
24. **Define immediate mode addressing.**
    In immediate mode addressing, the operand is given explicitly in the instruction.

    Eg. Move #200,R0.
25. **Define indirect mode addressing.**
    In indirect mode addressing the effective address of the operands is the content of a register or memory location whose address appears in the instruction. Eg: Add (R2), R0

PART-B

1. **Explain in detail about the eight ideas of computer architecture.**
   - Design for Moore's Law
   - Use abstraction to simplify design
   - Make the common case fast
   - Performance via Parallelism
   - Performance via Pipelining
   - Performance via Prediction
   - Hierarchy of Memory
   - Dependability via Redundancy

2. **Explain in detail the different Instruction types.Compare their merits and demerits. (Dec 2012) (May 2013)**

   A computer must have instructions capable of performing four types of operations.
   - Data transfers between the memory and the processor registers
   - Arithmetic and logic operations on data
   - Program sequencing and control
   - I/O transfers

   Basic Instructions Types:

Instruction Type	**Syntax**	**Eg**	**Description**
ThreeAddress	Operation Source1,Source2,Destination	Add A,B,C	Add values ofvariable A ,B &placethe result into c.
Two Address	Operation Source,Destination	Add A,B	Add thevalues ofA,B &placethe resultinto B.
OneAddress	Operation Operand	Add B	Content of accumulatoradd with content ofB.

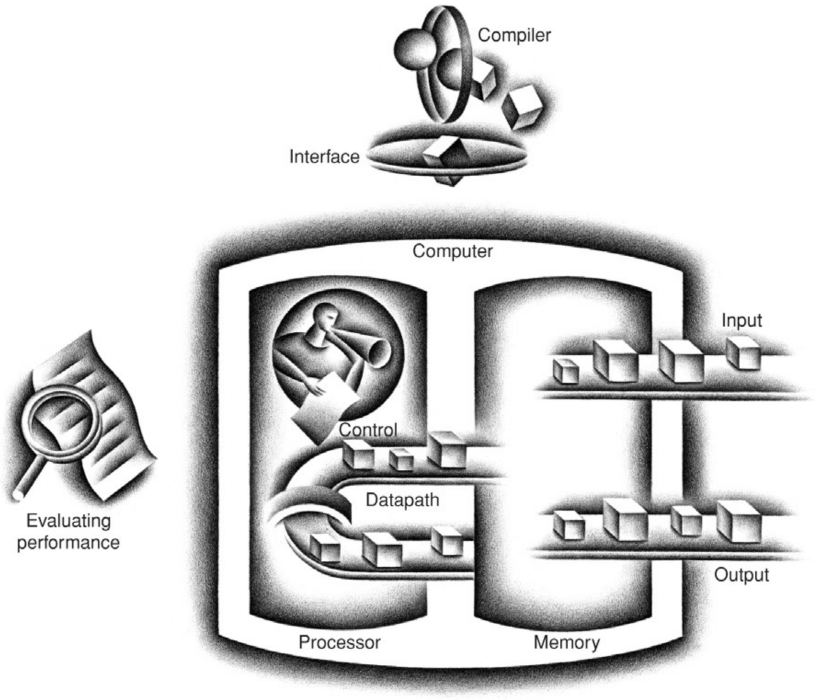
3. **Explain the different types of Addressing modes with suitable examples. (Dec 2012)**
   - The different ways in which the location of an operand is specified in an instruction are referred to as addressing modes.
   - **Register mode -** The operand is the contents of a processor register; the name (address) of the register is given in the instruction Eg: Move R1,R2
   - **Absolute mode –** The operand is in a memory location; the address of this location is given explicitly in the instruction. (In some assembly languages, this mode is called Direct).Eg: Move LOC, R2
   - **Immediate mode –** The operand is given explicitly in the instruction.

     For example, the instructionEg:Move #200, R0
   - **Indirect mode –** The effective address of the operand is the contents of a register or memory location whose address appears in the instruction.

     Eg: Add (R2), R0

- **Index mode –** the effective address of the operand is generated by adding a constant value to the contents of a register.The effective address of the operand is given byEA = X + [Rj]
- **Relative mode –** The effective address is determined by the Index mode using the program counter in place of the general-purpose register Ri.
- **Autoincrement mode –** the effective address of the operand is the contents of a register specified in the instruction. Eg. (Ri)+
- **Autodecrement mode –** the contents of a register specified in the instruction are first automatically decremented and are then used as the effective address of the operand.Eg. -(Ri)

4. Explain the architecture of a basic Computer.



- **Input Unit:**
    - This unit is used for entering data and programs into the computer system by the user for processing.
- **Memory Unit:**
    - The memory unit is used for storing data and instructions before and after processing.
- **Output Unit:**
    - The output unit is used for storing the result as output produced by the computer after processing.
- **Processing:**
    - The task of performing operations like arithmetic and logical operations is called processing. The Central Processing Unit (CPU) takes data and instructions from the storage unit and makes all sorts of calculations based on the instructions given and the type of data provided. It is then sent back to the storage unit. CPU includes Arithmetic logic unit (ALU) and control unit (CU)
- **Datapath:**

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

5. **Explain various performance measures of a computer.**
   - **Processor Clock**
     - Processor circuits are controlled by a timing signal called clock.
     - To execute a machine instruction the processor divides the action to be performed into a sequence of basic steps that each step can be completed in one clock cycle.
     - **Pipelining and superscalar operations**
       - The instructions are executed one after the other. Hence the value of S is the total number of basic steps, or clock cycles, required to execute one instruction.
       - A substantial improvement in performance can be achieved by overlapping the execution of successive instructions using a technique called pipelining.
     - **Clock rate**
       - These are two possibilities for increasing the clock rate 'R'.
       - Improving the IC technology makes logical circuit faster, which reduces the time of execution of basic steps. This allows the clock period P, to be reduced and the clock rate R to be increased.
       - Reducing the amount of processing done in one basic step also makes it possible to reduce the clock period P.
   - **Instruction set**
     - **Instruction set CISC & RISC**
     - Simple instructions require a small number of basic steps to execute.
     - Complex instructions involve a large number of steps. For a processor that has only simple instruction a large number of instructions may be needed to perform a given programming task.
   - **Compiler**
     - The SPEC rating is a measure of the combined effect of all factors affecting performance, including the compiler, the OS, the processor, the memory of comp being tested.

6. **Discuss in detail about Instruction performance and CPU performance of a computer**
   - The machine (or CPU) is said to be faster or has better performance running this program if the total execution time is shorter.
   - Thus the inverse of the total measured program execution time is a possible performance measure or metric: $\text{Performance}_A = 1 / \text{Execution Time}_A$
   - **CPU performance**

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

   - **Instruction performance**

     CPU clock cycles = Instructions for a program × Average clock cycles per instruction
   - **CPU Time**
     - doesn't count I/O or time spent running other programs
     - can be broken up into system time, and user time

       CPU time = Instruction Count × CPI × Clock Cycle Time

7. **Explain classes of computing applications and their characteristics.**

Computing applications

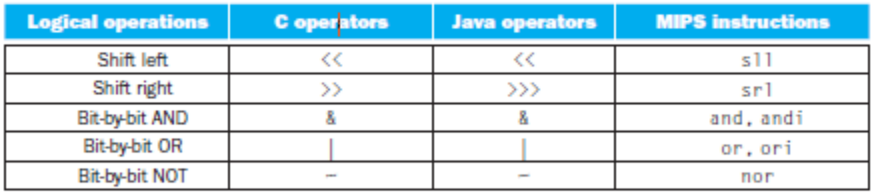
- Banking
- Insurance
- Education
- Marketing
- Health Care
- Engineering Design
- Military
- Communication
- Government Applications

Characteristics

- **Single user systems** are systems that only allow for one user to access the computer at a time
- **Multi User systems**are systems that allow continious access of multiple users at once and that are capable of running more than one process at a time in order to maximise the potential of all users
- Example: a server / computer where several people access a server / computer remotely
- **Single Tasking** is a reference to when a system is only doing one task at a time
- **Multi Tasking** allows a computer to run more than one task at a time by sceduling what tasks to do when in order to not have the computer make mistakes or do the wrong thing.

8. What are the various logical operations and explain the instructions supporting the logical operations.

Logical Operations:



Logical operations	C operators	Java operators	MIPS instructions
Shift left	<<	<<	sll
Shift right	>>	>>>	srl
Bit-by-bit AND	&	&	and, andi
Bit-by-bit OR	|	|	or, ori
Bit-by-bit NOT	~	~	nor

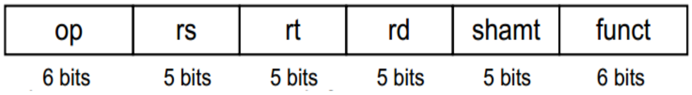
SHIFT

Shift Left Logical

Shift left and fill with 0 bits

Shift Right Logical

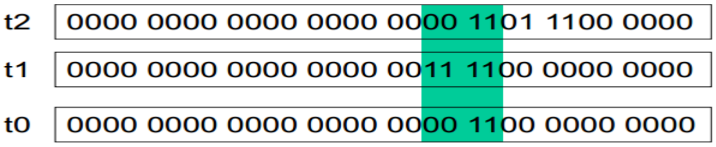
Shift right and fill with 0 bits



op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

AND

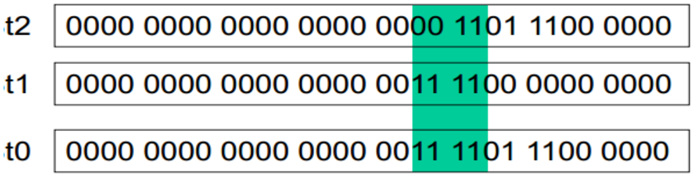
Used to mask bits in a work



```
t2  0000 0000 0000 0000 0000 1101 1100 0000
t1  0000 0000 0000 0000 0011 1100 0000 0000
t0  0000 0000 0000 0000 0000 1100 0000 0000
```

OR

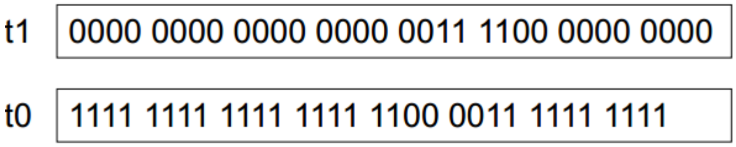
Used to include bit in a word



```
t2  0000 0000 0000 0000 0000 1101 1100 0000
t1  0000 0000 0000 0000 0011 1100 0000 0000
t0  0000 0000 0000 0000 0011 1101 1100 0000
```

NOT

Used to invert bits in a word



```
t1  0000 0000 0000 0000 0011 1100 0000 0000
t0  1111 1111 1111 1111 1100 0011 1111 1111
```

9. **Explain in detail about the instructions for control operation.**

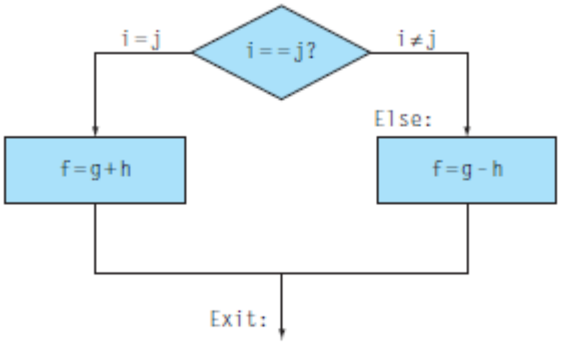
Conditional Branches

- **Branch if equal**

```
beq register1, register2, L1
```

- **Branch if not equal**

```
bne register1, register2, L1
```



Unconditional Branch& loops

- **Exit**

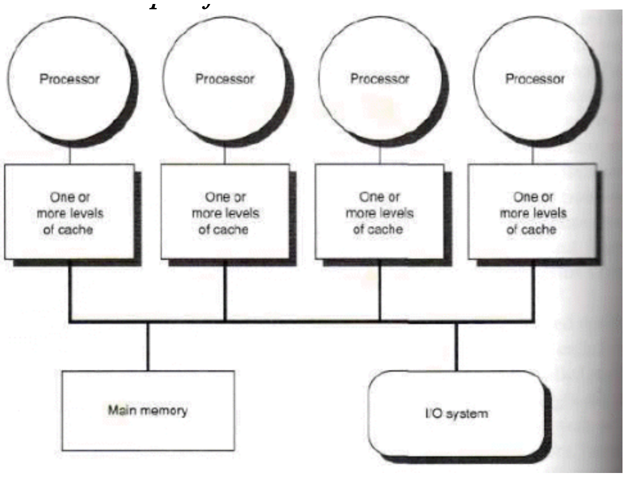
```
j Exit     # go to Exit
```

- **Loop**

10. **Briefly discuss about Uniprocessor and Multiprocessor.**

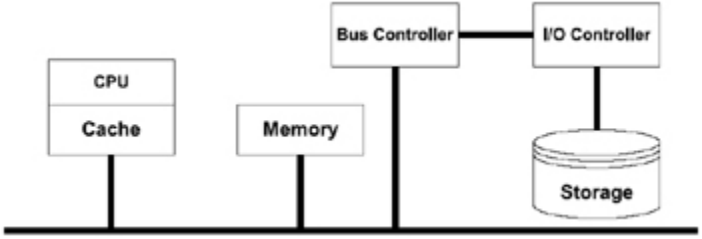
Multiprocessors

- Multiprocessors are parallel processors with a single shared address.
- Multiprocessors have the highest performance , it is higher than the fastest uni-processor.
- Multiprocessors have a lot of more effective applications than a uni-processor: search engines, web servers, databases….
- Multiprocessors communicate through shared variables in memory , all processors can access any memory location via loads and stores
- As processors operating in parallel, they normally share data. Only one processor at a
- time can acquire the lock and other processors interested in shared data have to wait until the original processor unlocks the variable so called Lock approach.


Processor
Processor
Processor
Processor
One or more levels of cache
One or more levels of cache
One or more levels of cache
One or more levels of cache
Main memory
I/O system

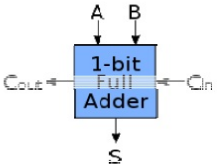
Uniprocessors

- Uniprocessor has only one CPU

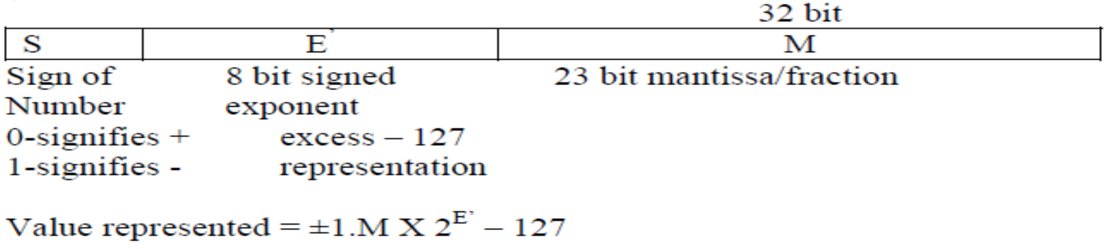


Unit – 2ARITHMETIC OPERATIONS
PART-A

1. **Define Full Adder (FA) with logic diagram**
   A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full adder adds three one-bit numbers, often written as *A*, *B*, and *C*in; *A* and *B* are the operands, and *C*in is a bit carried in (in theory from a past addition). The full-adder is usually a component in a cascade of adders, which add 8, 16, 32, etc.



2. **State the rule for floating point addition.**
   Choose the number with the smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents.
   Set the exponent of the result equal to the larger exponent.
   Perform the addition on the mantissa and determine the sign of the result.
   Normalize the resulting value if necessary.
3. **Draw the format of floating point number.**
   IEEE standard signal precision floating point number:
   Signal Precision: 32 bit



32 bit

S	E'	M
Sign of Number 0-signifies + 1-signifies -	8 bit signed exponent excess – 127 representation	23 bit mantissa/fraction

Value represented = $\pm 1.M \times 2^{E'} - 127$

Double precision representation contains 11 bits excess -1023 exponent E’ which has the range 1≤ E’ ≤ 2046 for normal values. This me that the actual exponent E is in range -1022 ≤ E ≤ 1023. The 53 bit mantissa provides a precision equivalent to about 16 decimal digits.

64 bit

S	E´	M
Sign bit	11 Bit excess -1023 Exponent	52 – bit mantissa fraction

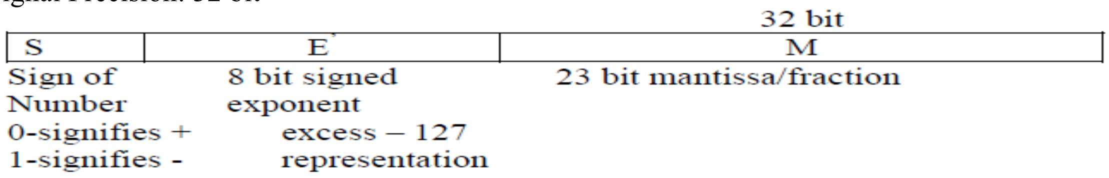
4. **What is guard bit . What are the ways to truncate the guard bits?**
Although the mantissa of initial operands is limited to 24 bits, it is important to retain extra bits, called as guard bits. There are several ways to truncate the guard bits:

    1)Chopping
    2)VonNeumannrounding
    3) Rounding

5. **What is overflow and underflow case in single precision?**
Underflow-The normalized representation requires an exponent less than -126
Overflow-The normalized representation requires an exponent greater than -126

6. **Why floating point number is more difficult to represent and process than integer?**
In floating point numbers we have to represent any number in three fields sign, exponent and mantissa. The IEEE 754 standard gibes the format for these fields and according to format the numbers are to be represented. In case of any process the mantissa and exponent are considered separately.

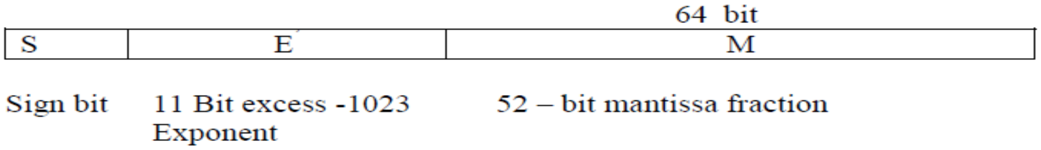
7. **Discuss the IEEE format used for representing single- precision floating point numbers.**
IEEE standard signal precision floating point number:
Signal Precision: 32 bit



Value represented = $\pm 1.M \text{ X } 2^{E'} - 127$

8. **When can you say that a number is normalized?**
When the decimal point is placed to the right of the first (nonzero) significant digit, the number is said to be normalized.

9. **Discuss the IEEE format used for Double precision floating point numbers.**
Double precision representation contains 11 bits excess -1023 exponent E' which has the range 1≤ E' ≤ 2046 for normal values. This me that the actual exponent E is in range -1022 ≤ E ≤ 1023. The 53 bit mantissa provides a precision equivalent to about 16 decimal digits.



10. **What is arithmetic overflow**
In a computer, the condition that occurs when a calculation produces a result that is greater in magnitude than which a given register or storage location can store or represent.
In a computer, the amount by which a calculated value is greater in magnitude than that which a given register or storage location can store or represent.

11. **Define Booth Algorithm.**
Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values A and S to a product P, then performing a rightward arithmetic shift on P.

12. **What is Carry Save addition?**
Using carry save addition, the delay can be reduced further still. The idea is to take 3 numbers that we want to add together, x+y+z, and convert it into 2 numbers c+s such that x+y+z=c+s, and do this in O (1) time. The reason why addition cannot be performed in O (1) time is because the carry information must be propagated. In carry save addition, we refrain from directly passing on the carry information until the very last step.

13. **Define Integer Division and give its rule.**
Integers are the set of whole numbers and their opposites. The sign of an integer is positive if the number is greater than zero, and the sign is negative if the number is less than zero. The set

of all integers represented by the set {... -4, -3, -2, -1, 0, 1, 2, 3, 4...}Negative integers: {. . . -4, -3, -2, -1}Positive integers: {1, 2, 3, 4 ...}
{0} is neither positive nor negative, neutral
DIVISION RULE: The quotient of two integers with same sign is positive. The quotient of two integers with opposite signs is negative.

14. **Write Restoring and Non-Restoring division algorithm**

Restoring Division Algorithm:
Shift A and Q left one binary position.
Subtract M from A, and place the answer back in A.
If the sign of A is 1, set q0 to 0 and add M back to A (that is, restore A); otherwise, set q0 to 1.
Non- Restoring Division Algorithm
Step 1: Do the following n times: If the sign of A is 0, shift A and Q left one bit position and subtract M from A; otherwise, shift A and Q left and add M to A. Now, if the sign of A is 0, set q0 to 1; otherwise,
set q0 to 0.
Step 2: If the Sign of A is 1, add M to A.

15. **Write the rules for add/sub operation on floating point numbers.**

Choose the number with the smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents.
Set the exponent of the result equal to the larger exponent
Perform addition / subtraction on the mantissa and determine the sign of the result
Normalize the resulting value, if necessary

16. **Write the rules for multiply operation on floating point numbers.**

Add the exponents and subtract 127.
Multiply the mantissa and determine the sign of the result.
Normalize the resulting value, if necessary.
Write the rules for divide operation on floating point numbers
Subtract the exponents and subtract 127.
Divide the mantissa and determine the sign of the result.
Normalize the resulting value, if necessary.

17. **Define Truncation.**

To retain maximum accuracy, all extra bits during operation (called *guard bits*) are kept (e.g., multiplication). If we assume $n = 3$bits are used in final representation of a number, $n = 3$extra guard bits are kept during operation. By the end of the operation, the resulting $2n = 6$bits need to be truncated to $n = 3$bits by one of the three methods.

18. **Define Chopping.**

There are several ways to truncate. The simplest way is to remove the guard bits and make no changes in the retained bits. This is called Chopping. Chopping discards the least significant bits and retains the 24 most significant digits
Easy to implement.
Biased, since all values are rounded to-wards a lower mantissa value.
Maximum rounding error 0≤e<+1 LSB.

19. **Define Von Neumann Rounding**

If at least one of the guard bits is 1, the least significant bit of the retained bits is set to 1 otherwise nothing is changed in retained bits and simply guard bits are dropped.

20. **What is Subword Parallelism?**

Subword parallelism is a technique that enables the full use of word-oriented data paths when dealing with lower precision data. It is a form of low-cost, small-scale SIMD parallelism.

PART-B

1. **Explain the Booth's algorithm for multiplication of signed two's complement numbers.**

- Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values A and S to a product P, then performing a rightward arithmetic shift on P. Let m and r be the multiplicand and multiplier, respectively; and let x and y represent the number of bits in m and r.

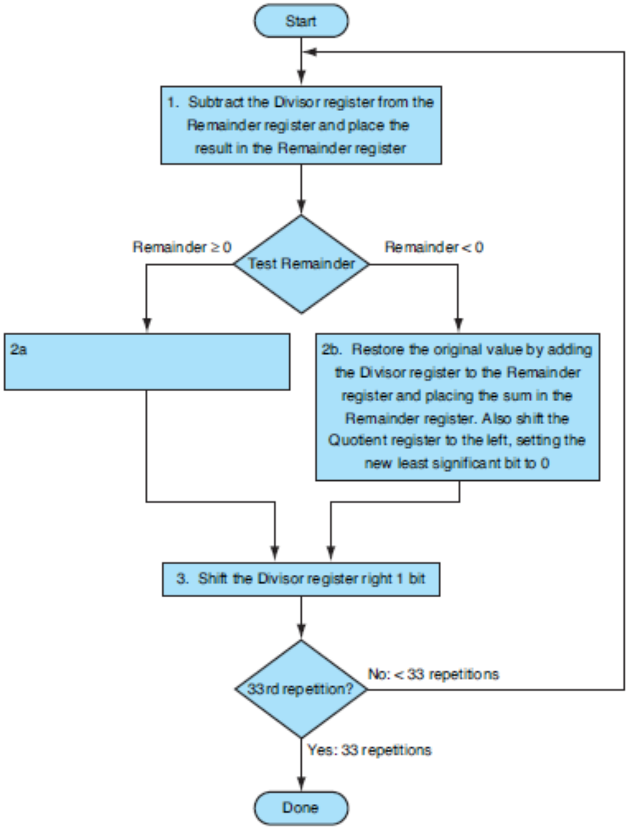
- Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to (x + y + 1).
    - A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining (y + 1) bits with zeros.
    - S: Fill the most significant bits with the value of (−m) in two's complement notation. Fill the remaining (y + 1) bits with zeros.
    - P: Fill the most significant x bits with zeros. To the right of this, append the value of r. Fill the least significant (rightmost) bit with a zero.
- Determine the two least significant (rightmost) bits of P.
    - If they are 01, find the value of P + A. Ignore any overflow.
    - If they are 10, find the value of P + S. Ignore any overflow.
    - If they are 00, do nothing. Use P directly in the next step.
    - If they are 11, do nothing. Use P directly in the next step.
- Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let P now equal this new value.
- Repeat steps 2 and 3 until they have been done y times.
- Drop the least significant (rightmost) bit from P. This is the product of m and r.

2. Explain the floating point addition and subtraction.

Step 1: Align the decimal point of the number that has the smaller exponent
Step 2: Add/Subract the significands
Step 3: The sum is adjust to normalized scientific notation
Step 4: Round the number to four digits

3. State the restoring division technique.

- If the remainder is positive, the divisor did go into the dividend, so step 2a generates a 1 in the quotient.
- A negative remainder after step 1 means that the divisor did not go into the dividend, so step 2b generates a 0 in the quotient and adds the divisor to the remainder, thereby reversing the subtraction of step 1.
- The final shift, in step 3, alignsthe divisor properly, relative to the dividend for the next iteration. These steps are repeated 33 times.


Start
1. Subtract the Divisor register from the Remainder register and place the result in the Remainder register
Remainder ≥ 0
Test Remainder
Remainder < 0
2a
2b. Restore the original value by adding the Divisor register to the Remainder register and placing the sum in the Remainder register. Also shift the Quotient register to the left, setting the new least significant bit to 0
3. Shift the Divisor register right 1 bit
33rd repetition?
No: < 33 repetitions
Yes: 33 repetitions
Done

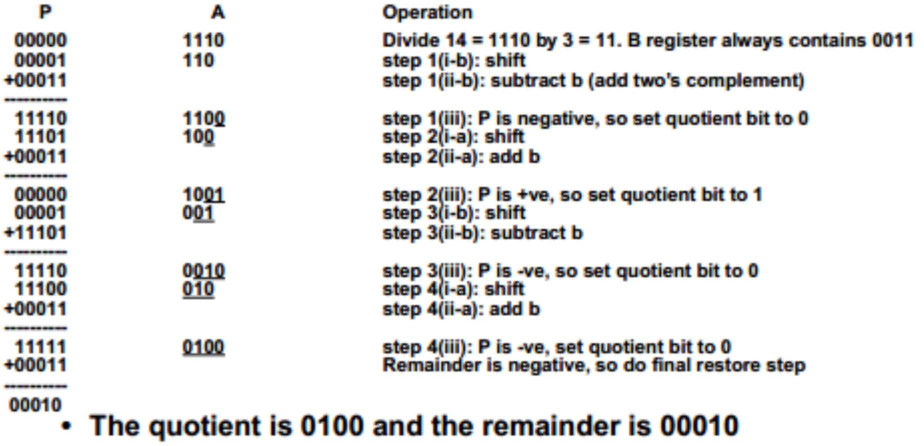
4. **State the Non – restoring division technique.**

A variant that skips the restoring step and instead workswith negative residuals

- If P is negative
  - (i-a) Shift the register pair (P,A) one bit left
  - (ii-a) Add the contents of register B to P
- If P is positive
  - (i-b) Shift the register pair (P,A) one bit left
  - (ii-b) Subtract the contents of register B from P
  - (iii) If P is negative, set the low-order bit of A to 0,

otherwise set it to 1

- After n cycles
  - The quotient is in A
- If P is positive, it is the remainder, otherwise it has to be restored add B to it to get the remainder.



P	A	Operation
00000	1110	Divide 14 = 1110 by 3 = 11. B register always contains 0011
00001	110	step 1(i-b): shift
+00011		step 1(ii-b): subtract b (add two's complement)

11110	1100	step 1(iii): P is negative, so set quotient bit to 0
11101	100	step 2(i-a): shift
+00011		step 2(ii-a): add b

00000	1001	step 2(iii): P is +ve, so set quotient bit to 1
00001	001	step 3(i-b): shift
+11101		step 3(ii-b): subtract b

11110	0010	step 3(iii): P is -ve, so set quotient bit to 0
11100	010	step 4(i-a): shift
+00011		step 4(ii-a): add b

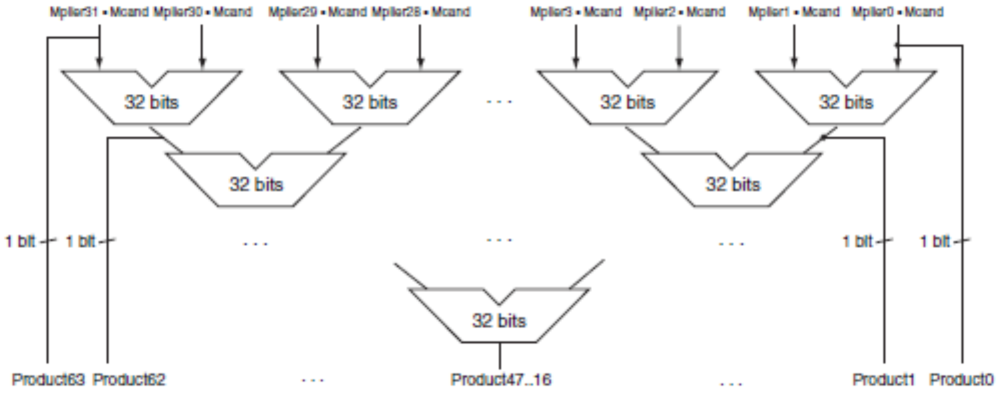
11111	0100	step 4(iii): P is -ve, set quotient bit to 0
+00011		Remainder is negative, so do final restore step

00010		

- **The quotient is 0100 and the remainder is 00010**

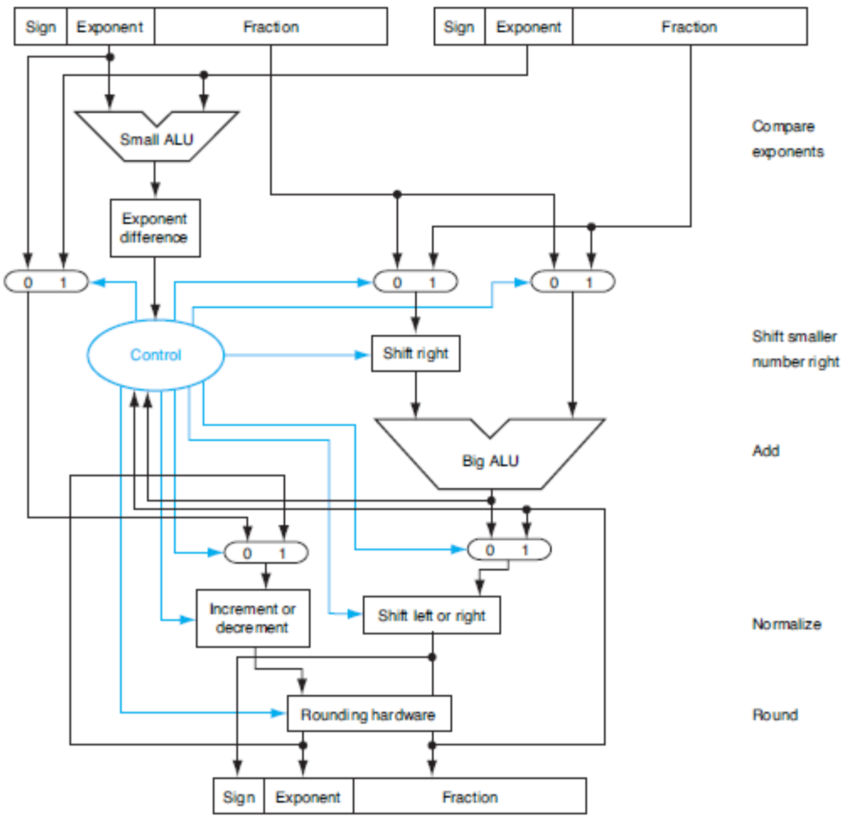
5. **Explain with a diagram the design of a fast multiplier using carry save adder circuit.**

- Faster multiplications are possible by essentially providingone 32-bit adder for each bit of the multiplier: one input is the multiplicand ANDedwith a multiplier bit, and the other is the output of a prior adder.
- A straightforward approach would be to connect the outputs of adders on the right to the inputs of adders on the left, making a stack of adders 32 high.
- Analternative way to organize these 32 additions is in a parallel treeshows. Instead of waiting for 32 add times, we wait just the log2 (32) or five 32-bit add times.

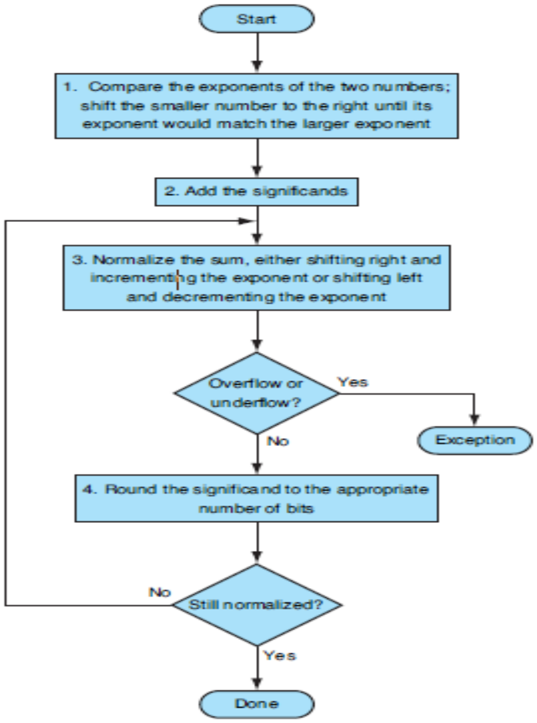


6. **Give the block diagram for a floating point adder and subtractor unit and discuss its operation.**

- First, the exponent of one operand is subtracted from the other using the small ALU to determine which islarger and by how much.
- This difference controls the three multiplexors; from left to right, they select the larger exponent, the significand of thesmaller number, and the significand of the larger number. The smaller significand is shifted right, and then the significands are added togetherusing the big ALU. The normalization step then shifts the sum left or right and increments or decrements the exponent.
- Rounding then createsthe final result, which may require normalizing again to produce the final result.



7. **Draw and explain the flowchart of floating point addition process.**


Start
1. Compare the exponents of the two numbers; shift the smaller number to the right until its exponent would match the larger exponent
2. Add the significands
3. Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent
Overflow or underflow?
Yes
Exception
No
4. Round the significand to the appropriate number of bits
No
Still normalized?
Yes
Done

8. **Explain in detail about the carry look ahead adder.**

- The CLA is used in most ALU designs
- It is faster compared to ripple carry logic adders or full adders especially when adding a large number of bits.
- The Carry Look Ahead Adder is able to generate carries before the sum is produced using the

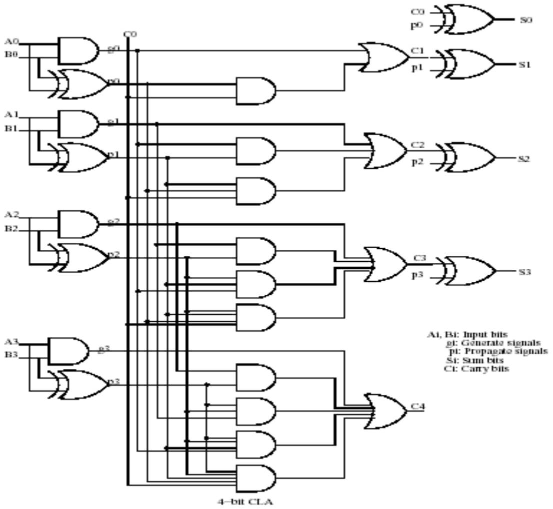
propagate and generate logic to make addition much faster.

$$G_i = A_i.B_i \qquad Pi = (A_i \oplus B_i)$$

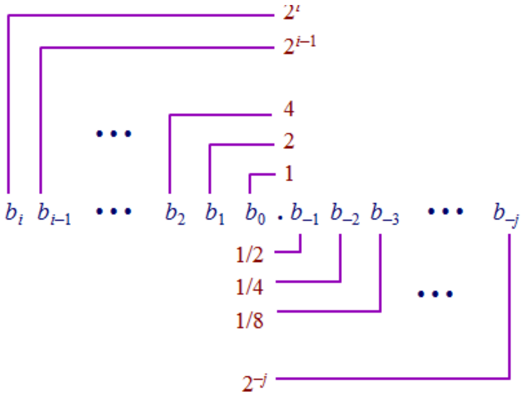
$$C_1 = G_0 + P_0.C_0$$
$$C_2 = G_1 + P_1.C_1 = G_1 + P1.G_0 + P_1.P_0.C_0$$
$$C_3 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.C_0$$
$$C_4 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3P_2P_1.G_0 + P_3P_2P_1.P_0.C_0$$

$$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i.$$

in Fig. 6.

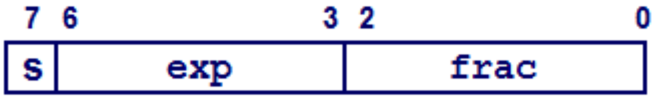


9. **Discuss about IEEE standards for Binary Arithmetic operations for floating point numbers IEEE Standard 754**
    - Established in 1985 as uniform standard for floating point arithmetic
        - Before that, many idiosyncratic formats
    - Supported by all major CPUs
    - **Fractional Binary number**



2^i
2^{i-1}
4
2
1
• • •
b_i b_{i-1} • • • b_2 b_1 b_0 . b_{-1} b_{-2} b_{-3} • • • b_{-j}
1/2
1/4
1/8
• • •
2^{-j}

- **8-bit Floating Point Representation**
    - the sign bit is in the most significant bit.
    - the next four bits are the exponent, with a bias of 7.
    - the last three bits are the frac
- Same General Form as IEEE Format
    - normalized, denormalized
    - representation of 0, NaN, infinity

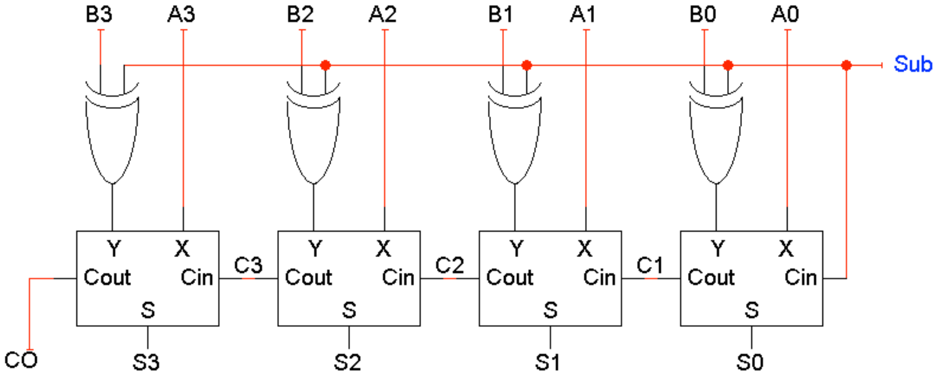


10. Draw and explain the block diagram of n-bit two's complement adder – sub tractor with an example.

- .XOR gates let us selectively complement the B input.

  $X \oplus 0 = X \qquad X \oplus 1 = X'$

- When Sub = 0, the XOR gates output B3 B2 B1 B0 and the carry in is 0. The adder output will be A + B + 0, or just A + B.
- When Sub = 1, the XOR gates output B3' B2' B1' B0' and the carry in is 1. Thus, the adder output will be a two's complement subtraction, A - B.



UNIT III PROCESSOR AND CONTROL UNIT
PART-A

1. **Whatis pipelining?**
   The technique of overlapping the execution of successive instruction for substantial improvement in performance is called pipelining.
2. **What is precise exception?**
   A precise exception is one in which all instructions prior to the faulting instruction are complete and instruction following the faulting instruction, including the faulty instruction; do not change the state of the machine.
3. **Define processor cycle in pipelining.**
   The time required between moving an instruction one step down the pipeline is a processor cycle.
4. **What is meant by pipeline bubble?**
   To resolve the hazard the pipeline is stall for 1 clock cycle. A stall is commonly called a pipeline bubble, since it floats through the pipeline taking space but carrying no useful work.
5. **What is pipeline register delay?**
   Adding registers between pipeline stages me adding logic between stages and setup and hold times for proper operations. This delay is known as pipeline register delay.
6. **What are the major characteristics of a pipeline?**
   The major characteristics of a pipeline are:
   1. Pipelining cannot be implemented on a single task, as it works by splitting multiple tasks into a number of subtasks and operating on them simultaneously.
   2. The speedup or efficiency achieved by suing a pipeline depends on the number of pipe stages and the number of available tasks that can be subdivided.
7. **What is data path?**

As instruction execution progress data are transferred from one instruction to another, often passing through the ALU to perform some arithmetic or logical operations. The registers, ALU, and the interconnecting bus are collectively referred as the data path.

8. **What is a pipeline hazard and what are its types?**
   Any condition that causes the pipeline to stall is called hazard. They are also called as stalls or bubbles. The various pipeline hazards are:
   Data hazard
   Structural Hazard
   Control Hazard.
9. **What is meant by data hazard in pipelining? (Nov/Dec 2013)**
   Any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline is called data hazard.
10. **What is Instruction or control hazard?**
    The pipeline may be stalled because of a delay in the availability of an instruction. For example, this may be a result of a miss in the cache, requiring the instruction to be fetched from the main memory. Such hazards are often called control hazards or instruction hazard.
11. **Define structural hazards.**
    This is the situation when two instruction require the use of a given hardware resource at the same time. The most common case in which this hazard may arise is in access to memory.
12. **What is side effect?**
    When a location other than one explicitly named in an instruction as a destination operand is affected, the instruction is said to have a side effect.
13. **What do you mean by branch penalty?**
    The time lost as a result of a branch instruction is often referred to as branch penalty.
14. **What is branch folding?**
    When the instruction fetch unit executes the branch instruction concurrently with the execution of the other instruction, then this technique is called branch folding.
15. **What do you mean by delayed branching?**
    Delayed branching is used to minimize the penalty incurred as a result of conditional branch instruction. The location following the branch instruction is called delay slot. The instructions in the delay slots are always fetched and they are arranged such that they are fully executed whether or not branch is taken. That is branching takes place one instruction later than where the branch instruction appears in the instruction sequence in the memory hence the name delayed branching.
16. **What are the two types of branch prediction techniques available?**
    The two types of branch prediction techniques are static branch prediction and dynamic branch prediction.
17. **What is a hazard? (Dec 2012) (May 2013)**
    Any condition that causes the pipeline to stall is called hazard. They are also called as stalls or bubbles. The various pipeline hazards are:
    Data hazard
    Structural Hazard
    Control Hazard.
18. **Define exception. (Dec 2012)**
    The term exception is used to refer to any event that causes an interruption.
19. **Why is branch prediction algorithm needed? Differentiate between the static and dynamic techniques. (May 2013)**
    The branch instruction will introduce branch penalty which would reduce the gain in performance expected from pipelining. Branch instructions can be handled in several ways to reduce their negative impact on the rate of execution of instructions. Thus the branch prediction algorithm is needed.
    Static Branch prediction
    The static branch prediction, assumes that the branch will not take place and to continue to fetch instructions in sequential address order.
    Dynamic Branch prediction

The idea is that the processor hardware assesses the likelihood of a given branch being taken by keeping track of branch decisions every time that instruction is executed. The execution history used in predicting the outcome of a given branch instruction is the result of the most recent execution of that instruction.

20. What is branch Target Address?

The address specified in a branch, which becomes the new program counter, if the branch is taken. In MIPS the branch target address is given by the sum of the offset field of the instruction and the address of the instruction following the branch.

21. What is an interrupt?

An exception that comes from outside of the processor. There are two types of interrupt.
1.Imprecise interrupt and 2.Precise interrupt

22. Define Pipeline speedup. (Nov/Dec 2013)

The ideal speedup from a pipeline is equal to the number of stages in the pipeline.

$$\frac{\text{Time per instruction on unpipelined machine}}{\text{Number of pipe stages}}$$

23. What is meant by vectored interrupt? (Nov/Dec 2013)

An interrupt for which the address to which control is transferred is determined by the cause of the exception.

PART B

1. Explain in detail about instruction execution characteristics.

To execute bus instruction it is necessary to perform following actions:

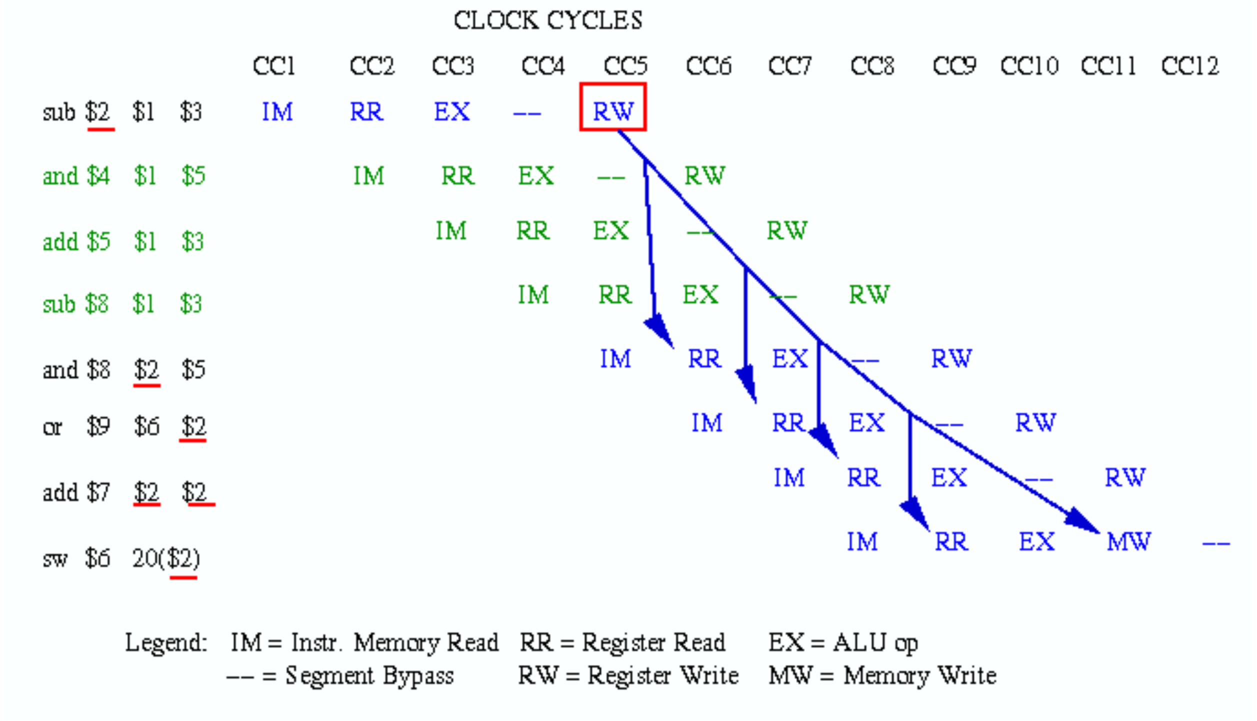
1. Fetch the instruction
2. Fetch the operand from memory location pointed by.
3. Perform the operation
4. Store the results

Add R1,(R2) for the single bus processor.

The sequence of control steps required to perform these operations for the single bus architecture are as follows;

1.PCout, MARin Yin, select C, Add, Zin
2. Zout, PCin, MARout , MARinM, Read
3. MDRout P,MARin
4. R2out , MARin
5. R2out , Yin,MARout , MARinM, Read
6. MDRout P, select Y, Add, Zin
7. Zout, R1in

2. Explain the function of a six segment pipeline showing the time it takes to process eight tasks.Give an account on branch penalty, branch folding, side effects, speculative execution.



CLOCK CYCLES

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12
sub $2 $1 $3	IM	RR	EX	--	RW							
and $4 $1 $5		IM	RR	EX	--	RW						
add $5 $1 $3			IM	RR	EX	--	RW					
sub $8 $1 $3				IM	RR	EX	--	RW				
and $8 $2 $5					IM	RR	EX	--	RW			
or $9 $6 $2						IM	RR	EX	--	RW		
add $7 $2 $2							IM	RR	EX	--	RW	
sw $6 20($2)								IM	RR	EX	MW	--

Legend: IM = Instr. Memory Read RR = Register Read EX = ALU op
-- = Segment Bypass RW = Register Write MW = Memory Write

3. **Discuss the role of cache in pipelining.**

Role of Cache Memory

Each pipeline stage is expected to complete in one clock cycle.

- ❖ The clock period should be long enough to let the slowest pipeline stage to complete.
- ❖ Faster stages can only wait for the slowest one to complete.
- ❖ Pipeline Most effective –if task performed in different stages require same amount of time.
- ❖ Since main memory is very slow compared to the execution, if each instruction needs to be fetched from main memory, pipeline is almost useless.
  Use of cache solves the memory access problem.

Pipeline Performance

- The potential increase in performance resulting from pipelining is proportional to the number of pipeline stages.
- However, this increase would be achieved only if all pipeline stages require the same time to complete, and there is no interruption throughout program execution.
- Unfortunately, this is not true.

Performance considerations

- The execution time T of a program that has a dynamic instruction count N is given by:
- where S is the average number of clock cycles it takes to fetch and execute one instruction, and R is the clock rate.
- Instruction throughput is defined as the number of instructions executed per second.
- An *n*-stage pipeline has the potential to increase the throughput by *n* times.
- However, the only real measure of performance is the total execution time of a program.
- Higher instruction throughput will not necessarily lead to higher performance.

4. **What is data hazard? Explain the ways and means of handing it. (Dec 2012)**

Data hazards occur when data is modified. Ignoring potential data hazards can result in race conditions (sometimes known as race hazards). There are three situations a data hazard can occur in:

1. **Read after Write** (RAW) or **True dependency**: An operand is modified and read soon after. Because the first instruction may not have finished writing to the operand, the second instruction may use incorrect data.

2. **Write after Read** (WAR) or **Anti dependency**: Read an operand and write soon after to that same operand. Because the write may have finished before the read, the read instruction may incorrectly get the new written value.

3. **Write after Write** (WAW) or **Output dependency**: Two instructions that write to the same operand are performed. The first one issued may finish second, and therefore leave the operand with an incorrect data value.

Eliminating data hazards

Forwarding

Forwarding involves feeding output data into a previous stage of the pipeline.

5. **Explain 4-stage instruction pipeline. Also explain the issues affecting pipeline performance.**

6. Explain the dynamic branch prediction technique. (May 2013)

The idea is that the processor hardware assesses the likelihood of a given branch being taken by keeping track of branch decisions every time that instruction is executed. The execution history used in predicting the outcome of a given branch instruction is the result of the most recent execution of that instruction. The processor assumes that the next time the instruction is executed; the result is likely to be the same. Hence, the algorithm may be described by the **two-state machine.**

The two states are:

LT: Branch is likely to be taken

LNT: Branch is likely not to be taken

Suppose that the algorithm is started in state LNT. When the branch instruction is executed and if the branch is taken, the machine moves to state LT. Otherwise, it remains in state LNT. the next time the same instruction is encountered, the branch is predicted as taken if the corresponding state machine is in state LT. Otherwise it is predicted as not taken. In this simple scheme, it requires one bit (0 for branch not taken and 1 for branch taken) of history information for each branch instruction.

Better performance can be achieved by keeping more information about execution history. An algorithm that uses 4 states, Thus requiring two bits of history information for each branch instruction.

The four states are:

ST: Strongly likely to be taken
LT: Likely to be taken
LNT: Likely not to be taken
SNT: Strongly likely not to be taken

Again assume that the state of the algorithm is initially set to LNT. After the branch instruction has been executed, and if the branch is actually taken, the state is changed to ST; otherwise, it is changed to SNT. As program execution progresses and the same instruction is encountered again, the state of the branch prediction algorithm continues to change as shown. When a branch instruction is encountered, the instruction fetch unit predicts that the branch will be taken if the state is either LT or ST, and it begins to fetch instructions at the branch target address. Otherwise, it continues to fetch instructions in sequential address order.

7. Describe the techniques for handling control hazard in pipelining. (May 2013)

- When a branch is executed it may or it may not change the PC (to other value than its value + 4)
– If a branch is changing the PC to its target address, than it is a *taken* branch
– If a branch doesn't change the PC to its target address, than it is a *not taken* branch
- If instruction i is a taken branch, than the value of PC will not change until the end MEM stage of the instruction execution in the pipeline
– A simple method to deal with branches is to stall the pipe as soon as we detect a branch until we know the result of the branch

Branch Instruction	IF	ID	EX	MEM	WB			
Branch Successor		IF	stall	stall	IF	ID	EX	MEM
Branch Successor +1						IF	ID	EX
Branch Successor +2							IF	ID

- A branch causes three cycle stall in our example processor pipeline
– One cycle is a repeated IF – necessary if the branch would be taken. If the branch is not taken, this IF is redundant
– Two idle cycles

8. Explain the data path and control consideration of a pipeline organization and explain.

The processor can perform the following operations independently,

1. Reading an instruction from the instruction cache.
2. Incrementing the PC.
3. Decoding an instruction by instruction decoder.
4. Reading from and writing into the data cache.
5. Reading the contents of up to two registers from the register file.
6. Writing into one register in the register file.

7. Performing an ALU operation.

Data path modified for pipelined execution are

- Separate instruction and data caches
- PC is connected to IMAR
- DMAR
- Separate MDR
- Buffers for ALU
- Instruction queue
- Instruction decoder output

9. Briefly explain the speedup performance models for pipelining.(Dec 2012)(Dec 2013)

Performance of Pipelines with Stalls

A stall causes the pipeline performance to degrade the ideal performance.

$$\textbf{Speedup from pipelining} = \frac{\textbf{Average instruction time unpipelined}}{\textbf{Average instruction time pipelined}}$$

$$= \frac{\textbf{CPI unpipelined * Clock Cycle Time unpipelined}}{\textbf{CPI pipelined * Clock Cycle Time pipelined}}$$

The ideal CPI on a pipelined machine is almost always 1. Hence, the pipelined CPI is

CPIpipelined = Ideal CPI + Pipeline stall clock cycles per instruction

= 1 + Pipeline stall clock cycles per instruction

If we ignore the cycle time overhead of pipelining and assume the stages are all perfectly balanced, then the cycle time of the two machines are equal and

$$\textbf{Speedup} = \frac{\textbf{CPI unpipelined}}{\textbf{1+ Pipeline stall cycles per instruction}}$$

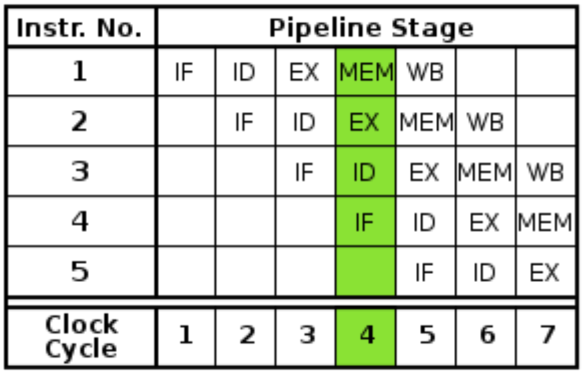
If all instructions take the same number of cycles, which must also equal the number of pipeline stages (the depth of the pipeline) then unpipelined CPI is equal to the depth of the pipeline, leading to

$$\textbf{Speedup} = \frac{\textbf{Pipeline depth}}{\textbf{1 + Pipeline stall cycles per instruction}}$$

If there are no pipeline stalls, this leads to the intuitive result that pipelining can improve performance by the depth of pipeline.

10. What is instruction hazards? Explain in detail how to handle the instruction hazards in pipelining with relevant examples .

An instruction pipeline is a technique used in the design of computers to increase their instruction throughput (the number of instructions that can be executed in a unit of time). The basic instruction cycle is broken up into a series called a pipeline. Rather than processing each instruction sequentially (one at a time, finishing one instruction before starting the next), each instruction is split up into a sequence of steps – different steps can be executed concurrently (by different circuitry) and indeed in parallel (at the same time).



Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Pipelining increases instruction throughput by performing multiple operations at the same time (in parallel), but does not reduce instruction latency (the time to complete a single instruction from start to finish) as it still must go through all steps. Indeed, it may increase latency due to additional overhead from breaking the computation into separate steps and worse, the pipeline may stall (or even need to be flushed), further increasing latency. Pipelining thus increases throughput at the cost of latency, and is frequently used in CPUs, but avoided in realtime systems, where latency is a hard constraint.

Each instruction is split into a sequence of dependent steps. The first step is always to fetch the instruction from memory; the final step is usually writing the results of the instruction to processor registers or to memory. Pipelining seeks to let the processor work on as many instructions as there are dependent steps, just as an assembly line builds many vehicles at once, rather than waiting until one vehicle has passed through the line before admitting the next one. Just as the goal of the assembly line is to keep each assembler productive at all times, pipelining seeks to keep every portion of the processor busy with some instruction. Pipelining lets the computer's cycle time be the time of the slowest step, and ideally lets one instruction complete in every cycle.

11. Write note on exception handling. .(Nov/Dec 2013)

Out-of-order Execution

The dispatch unit dispatches the instructions in the order in which they appear in the program. But their execution may be completed in the different order.

There are two causes of exceptions,

- Imprecise exceptions
- Precise exceptions

Imprecise exception

If instruction I1 causes an exception and succeeding instructions are permitted to complete execution, then the processor is said to have imprecise exceptions. Because of the exception by I1, program execution is in an inconsistent state.

Precise exception

In precise exception, if the exception occurs during an instruction, the succeeding instructions, may have been partially executed are discarded.

If an external interrupt is received, the dispatch unit stops reading new instructions from the instruction queue. The instructions which are placed in the instruction queue are discarded.

The processor first completes the pending execution of the instructions to completion.

The consistent state of the processor and all its registers is achieved.

Then the processor starts the interrupt handling process.

Deadlock

Consider 2 units U1 and U2 are using shared resources.

U2 needs completion of the task assign to unit U1 to complete its task.

If unit U2 is using a resource which is also required to unit U1, both units cannot complete the tasks assigned to them.

Both the units remain waiting for the need resource.

Also, unit U2 is waiting for the completion of task by unit U1 before it can release that resource. Such a situation is called a deadlock.

UNIT IV PARALLELISM
PART-A

1. **What is Instruction level parallelism?**
ILP is a measure of how many of the operations in a computer program can be performed simultaneously. The potential overlap among instructions is called instruction level parallelism.
2. **Define Static multiple issue.**
An approach to implementing a multiple-issue processor where many decisions are made by the compiler before execution.
3. **Define Dynamic multiple issue.**
An approach to implementing a multiple-issue processor where many decisions are made during execution by the processor.
4. **What is Speculation?**
An approach whereby the compiler or processor guesses the outcome of an instruction to remove it as a dependence in executing other instructions.
5. **Define Use latency.**
Number of clock cycles between a load instruction and an instruction that can use the result of the load with-out stalling the pipeline.
6. **What is Loop unrolling?**
A technique to get more performance from loops that access arrays, in which multiple copies of the loop body are made and instructions from different iterations are scheduled together.
7. **Define Register renaming.**
The renaming of registers by the compiler or hardware to remove anti-dependences.
8. **What is Superscalar?**
An advanced pipelining technique that enables the processor to execute more than one instruction per clock cycle by selecting them during execution.
9. **What is Dynamic pipeline schedule?**
Hardware support for reordering the order of instruction execution so as to avoid stalls.
10. **Define Commit unit.**
The unit in a dynamic or out-of-order execution pipeline that decides when it is safe to release the result of an operation to programmer visible registers and memory.
11. **What is Reservation station?**
A buffer within a functional unit that holds the operands and the operation.
12. **Define Reorder buffer?**
The buffer that holds results in a dynamically scheduled processor until it is safe to store the results to memory or a register.
13. **Define Out of order execution.**
A situation in pipelined execution when an instruction blocked from executing does not cause the following instructions to wait.
14. **What is In order commit?**
A commit in which the results of pipelined execution are written to the programmer visible state in the same order that instructions are fetched.
15. **Define Strong scaling.**
Speed-up achieved on a multi-processor without increasing the size of the problem.
16. **Define weak scaling.**
Speed-up achieved on a multi-processor while increasing the size of the problem proportionally to the increase in the number of processors.

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

18. **Define Single Instruction, Multiple Data streams (SIMD)**
A computer which exploits multiple data streams against a single instruction stream to perform operations which may be naturally parallelized. For example, an array processor or GPU.

19. **Define Multiple Instruction, Single Data stream (MISD)**
Multiple instructions operate on a single data stream. Uncommon architecture which is generally used for fault tolerance. Heterogeneous systems operate on the same data stream and must agree on the result. Examples include the Space Shuttle flight control computer.

20. **Define Multiple Instruction, Multiple Data streams (MIMD)**
Multiple autonomous processors simultaneously executing different instructions on different data. Distributed systems are generally recognized to be MIMD architectures; either exploiting a single shared memory space or a distributed memory space. A multi-coresuperscalar processor is an MIMD processor.

21. **What is Single program multiple data streams?**
Multiple autonomous processors simultaneously executing the same program on different data.

22. **Define multithreading.**
Multiple threads to share the functional units of 1 processor via overlapping

23. processor must duplicate independent state of each thread e.g., a separate copy of register file, a separate PC, and for running independent programs, a separate page tablememory shared through the virtual memory mechanisms, which already support multiple processes

24. **What is Fine grained multithreading?**
Switches between threads on each instruction, causing the execution of multiples threads to be interleaved,
    - Usually done in a round-robin fashion, skipping any stalled threads
    - CPU must be able to switch threads every clock

25. **What is Coarse grained multithreading?**
Switches threads only on costly stalls, such as L2 cache misses

26. **Define Multicore processors.**
A multi-core processor is a processing system composed of two or more independent cores. The cores are typically integrated onto a single integrated circuit die or they may be integrated onto multiple dies in a single chip package.

27. **What is symmetric multi-core processor?**
A symmetric multi-core processoris one that has multiple cores on a single chip, and all of those cores are identical.Example: Intel Core 2.

28. **What is asymmetric multi-core processor?**
In an asymmetric multi-core processor, the chip has multiple cores onboard, but the cores might be different designs.Each core will have different capabilities

PART-B

1. **Explain Instruction level parallelism.**
Architectural technique that allows the overlap of individual machine operations (add, mul, load, store …)
Multiple operations will execute in parallel (simultaneously)
Goal: Speed Up the execution
Example:

```
load R1 ← R2              add  R3 ← R3, “1”
add  R3 ← R3, “1”         add  R4 ← R3, R2
add  R4 ← R4, R2          store [R4] ← R0
```

Sequential execution (Without ILP)
Add r1, r2 → r84 cycles
Add r3, r4 → r74 cycles 8 cycles

ILP execution (overlap execution)
Add r1, r2 → r8
Add r3, r4 → r7
Total of 5 cycles

ILP Architectures

Sequential Architectures: the program is not expected to convey any explicit information regarding parallelism. (Superscalar processors)

Dependence Architectures: the program explicitly indicates the dependences that exist between operations (Dataflow processors)

Independence Architectures: the program provides information as to which operations are independent of one another. (VLIW processors)

ILP Scheduling

2. Explain challenges in parallel processing.

Parallel Processing encompasses a wide variety of different things:

Intel Core Duo, Quad, Cell multiprocessors, networked and distributed computer systems, SETI@Home, Folding@Home, neural nets are all examples

Connecting your CPUs.

Challenges in parallel processing

Dynamic vs Static—connections can change from one communication to next
Blocking vs Nonblocking—can simultaneous connections be present?
Connections can be complete, linear, star, grid, tree, hypercube, etc.
Bus-based routing
Crossbar switching—impractical for all but the most expensive super-computers
2X2 switch—can route inputs to different destinations

3. Explain in detail about Hardware multithreading.

Coarse grained multithreading

- Single thread runs until a costly stall
  - E.g. 2nd level cache miss
- Another thread starts during stall for first
  - Pipeline fill time requires several cycles!
- Does not cover short stalls
- Less likely to slow execution of a single thread (smaller latency)
- Needs hardware support
  - PC and register file for each thread
  - little other hardware

Fine Grained Multithreading

- Two or more threads interleave instructions
  - Round-robin fashion
  - Skip stalled threads
- Needs hardware support
  - Separate PC and register file for each thread
  - Hardware to control alternating pattern
- Naturally hides delays
  - Data hazards, Cache misses
  - Pipeline runs with rare stalls

Simultaneous Multithreading

- Instructions from multiple threads issued on same cycle
  - Uses register renaming and dynamic scheduling facility of multi-issue architecture
- Needs more hardware support
  - Register files, PC's for each thread
  - Temporary result registers before commit
  - Support to sort out which threads get results from which instructions
- Maximizes utilization of execution units

4. Discuss in detail about Flynn's classification.

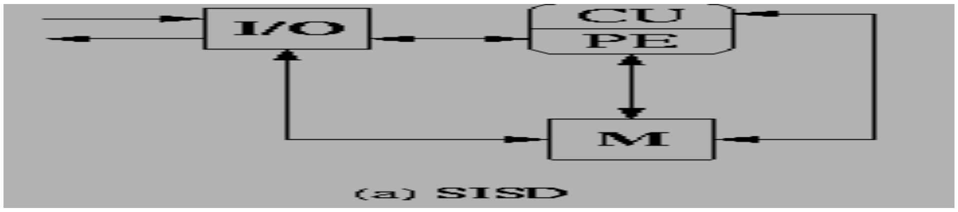
- Michael Flynn proposed a classification for computer architectures based on the number of instruction steams and data streams (Flynn's Taxonomy).
- Flynn uses the stream concept for describing a machine's structure
- A stream simply means a sequence of items (data or instructions).
- The classification of computer architectures based on the number of instruction steams and data streams (Flynn's Taxonomy).

Flynn's Taxonomy

SISD: Single instruction single data
 – Classical von Neumann architecture
SIMD: Single instruction multiple data
MISD: Multiple instructions single data
 – Non existent, just listed for completeness
MIMD: Multiple instructions multiple data
 – Most common and general parallel machine

5. Explain SISD and SIMD with an example.

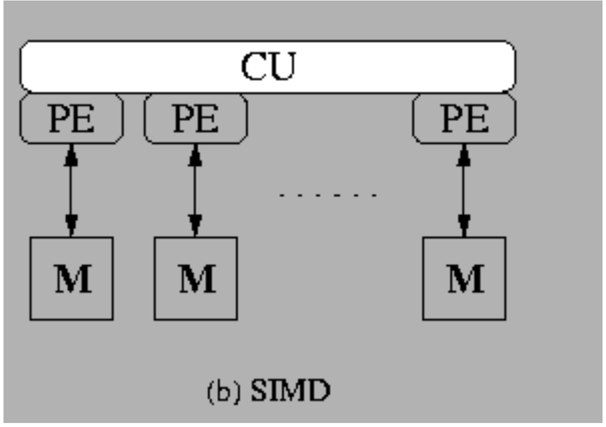
- SISD (Singe-Instruction stream, Singe-Data stream)
- SISD corresponds to the traditional mono-processor (von Neumann computer). A single data stream is being processed by one instruction stream OR
- A single-processor computer (uni-processor) in which a single stream of instructions is generated from the program.



(a) SISD

where CU= Control Unit, PE= Processing Element, M= Memory

- SIMD (Single-Instruction stream, Multiple-Data streams)
- Each instruction is executed on a different set of data by different processors i.e multiple processing units of the same type process on multiple-data streams.
- This group is dedicated to array processing machines.
- Sometimes, vector processors can also be seen as a part of this group.

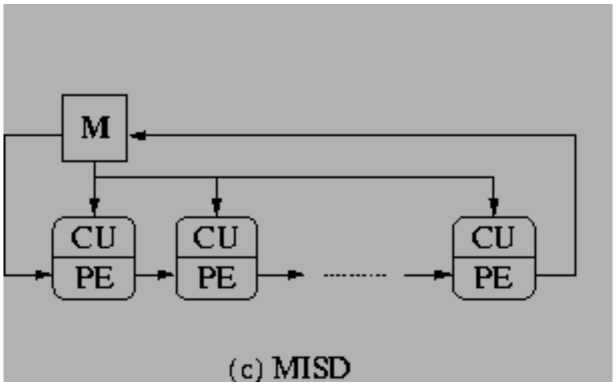


(b) SIMD

w here CU= Control Unit, PE= Processing Element, M= Memory

6. **Explain MISD and MIMD with an example.**

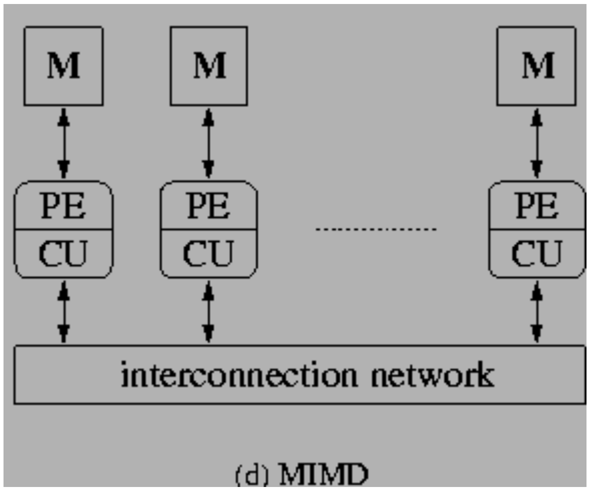
- MISD (Multiple-Instruction streams, Singe-Data stream)
- Each processor executes a different sequence of instructions.
- In case of MISD computers, multiple processing units operate on one single-data stream .
- In practice, this kind of organization has never been used



(c) MISD

where CU= Control Unit, PE= Processing Element, M= Memory

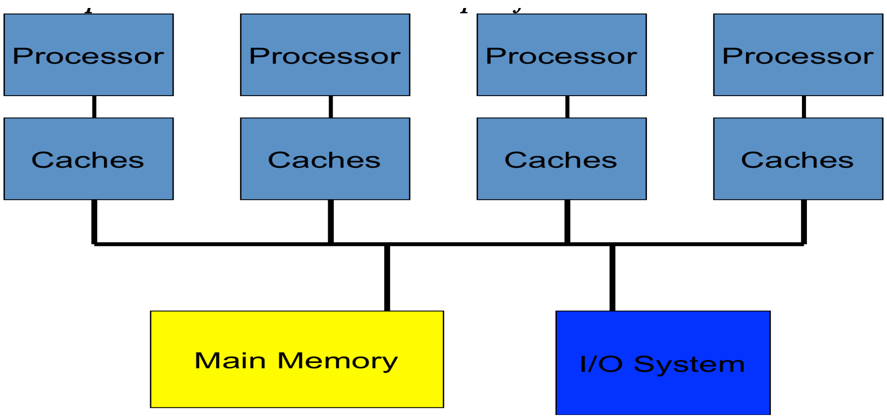
- **MIMD** (**M**ultiple-**I**nstruction streams, **M**ultiple-**D**ata streams)
- Each processor has a separate program.
- An instruction stream is generated from each program.
- Each instruction operates on different data.
- This last machine type builds the group for the traditional multi-processors. Several processing units operate on multiple-data streams.



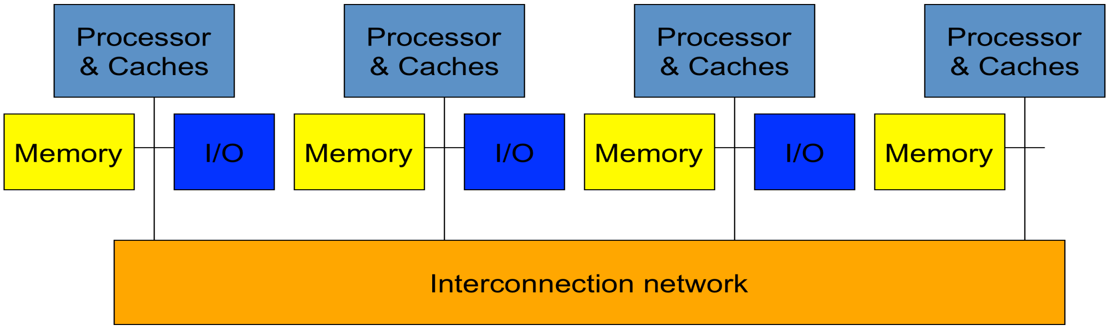
(d) MIMD

7. **Discuss Shared memory multiprocessor with a neat diagram.**

- **Centralized shared-memory multiprocessor or Symmetric shared-memory multiprocessor (SMP)**
- **Multiple processors connected to a single centralized memory – since all processors see the same memory organization → uniform memory access (UMA)**
- **Shared-memory because all processors can access the entire memory address space**

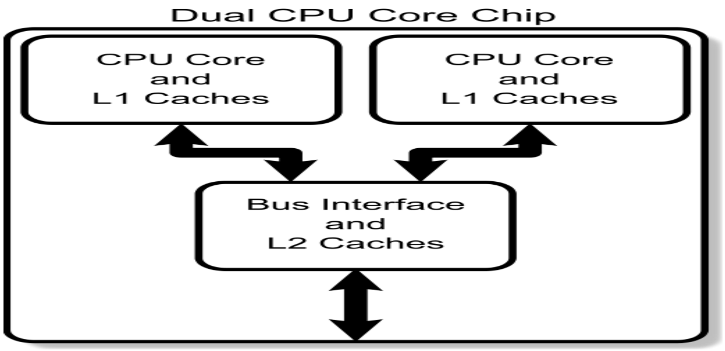


- **For higher scalability, memory is distributed among processors → distributed memory multiprocessors**
- **If one processor can directly address the memory local to another processor, the address space is shared → distributed shared-memory (DSM) multiprocessor**
- **If memories are strictly local, we need messages to communicate data → cluster of computers or multicomputers**
- **Non-uniform memory architecture (NUMA) since local memory has lower latency than remote memory**



8. Discuss briefly about the motivation of Multi-core computing.

- A multi-core processor is a processing system composed of two or more independent cores (or CPUs). The cores are typically integrated onto a single integrated circuit die (known as a chip multiprocessor or CMP), or they may be integrated onto multiple dies in a single chip package.
- A many-core processor is one in which the number of cores is large enough that traditional multi-processor techniques are no longer efficient - this threshold is somewhere in the range of several tens of cores - and likely requires a network on chip.



- Dual-core processor contains two independent microprocessors.
- A dual core set-up is somewhat comparable to having multiple, separate processors installed in the same computer, but because the two processors are actually plugged into the same socket, the connection between them is faster. Ideally, a dual core processor is nearly twice as

powerful as a single core processor. In practice, performance gains are said to be about fifty percent: a dual core processor is likely to be about one-and-a-half times as powerful as a single core processor.

9. **Explain how to overcome data hazard with dynamic scheduling.**
   Dynamic Scheduling is when the hardware rearranges the order of instruction execution to reduce stalls.

- It handles cases when dependences unknown at compile time
    - it allows the processor to tolerate unpredictable delays such as cache misses, by executing other code while waiting for the miss to resolve
- It allows code that compiled for one pipeline to run efficiently on a different pipeline
- It simplifies the compiler
- Hardware speculation, a technique with significant performance advantages, builds on dynamic scheduling (next lectures)

Advantages:

- Dependencies unknown at compile time can be handled by the hardware.
- Code compiled for one type of pipeline can be efficiently run on another.

Disadvantages:

- Hardware much more complex.

10. **Explain in detail how branch penalties are reduced with dynamic hardware prediction.**

- static approaches to deal with branch penalties (assuming taken or not taken branches, using the branch delay slot)
- Here, we will examine dynamic hardware oriented approaches:
    - branch-prediction buffer
    - two-bit prediction scheme
    - branch-target buffer
    - branch folding
- Consider that if we know for a particular branch that it is usually taken, then we can predict it will be taken this time
- The buffer is a small cache indexed by the low-order bits of the address of the branch instruction
- This buffer contains a bit as to whether the branch was last taken or not (1 time history)
- Upon fetching the instruction, we also fetch the prediction bit (from a special cache), and use it to determine whether to start the branch process or ignore the branch process
- Consider a prediction accuracy of 90% and a hit rate of 90%, if 60% of all branches are actually taken, what is the average branch penalty using this scheme?
    - Branch penalty = hit rate * percent incorrect predictions * 2 cycles + (1 - hit rate) * taken branches * 2 cycles = (90% * 10% * 2) + (1 - 90%) * 60% * 2 = .3 cycles
    - Using delayed branches we had an average branch penalty of .5 cycles, so we improve with this approach

UNIT-IV MEMORY AND I/O SYSTEMS

PART-A

1. **What is principle of locality?**
   The principle of locality states that programs access a relatively small portion of their address space at any instant of time.
2. **Define temporal locality.**
   The principle stating that a data location is referenced then it will tend to be referenced again soon.
3. **Define spatial locality.**
   The locality principle stating that if a data location is referenced, data locations with nearby addresses will tend to be referenced soon.

4. **Define Memory Hierarchy.**

   A structure that uses multiple levels of memory with different speeds and sizes. The faster memories are more expensive per bit than the slower memories.

5. **Define Hit and Miss? (DEC 2013)**

   The performance of cache memory is frequently measured in terms of a quantity called hit ratio. When the CPU refers to memory and finds the word in cache, it is said to produce a hit. If the word is not found in cache, then it is in main memory and it counts as a miss.

6. **What is cache memory?**

   It is a fast memory that is inserted between the larger slower main memory and the processor. It holds the currently active segments of a program and their data.

7. **What is Direct mapped cache?**

A cache structure in which each memory location is mapped to exactly one location in the cache.

8. **Define write through.**

   A scheme in which writes always update both the cache and the next lower level of the memory hierarchy, ensuring the data is always consistent between the two.

9. **Define write buffer.**

   A queue that holds data while the data is waiting to be written to memory.

10. **What is write-back?**

    A scheme that handles writes by updating values only to the block in the cache, then writing the modified block to the lower level of the hierarchy when the block is replaced.

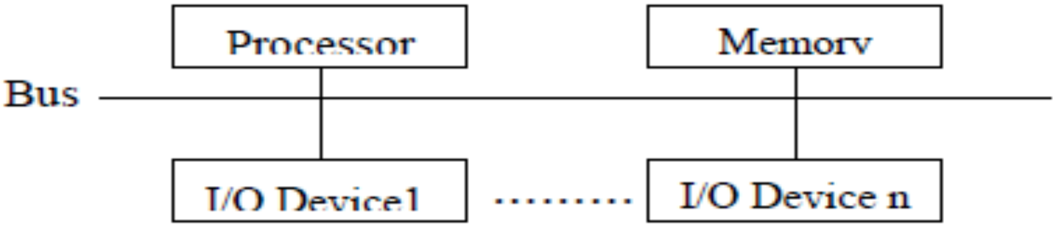
11. **Define virtual memory.**

    The data is to be stored in physical memory locations that have addresses different from those specified by the program. The memory control circuitry translates the address specified by the program into an address that can be used to access the physical memory

12. **Distinguish between memory mapped I/O and I/O mapped I/O.**

    Memory mapped I/O:

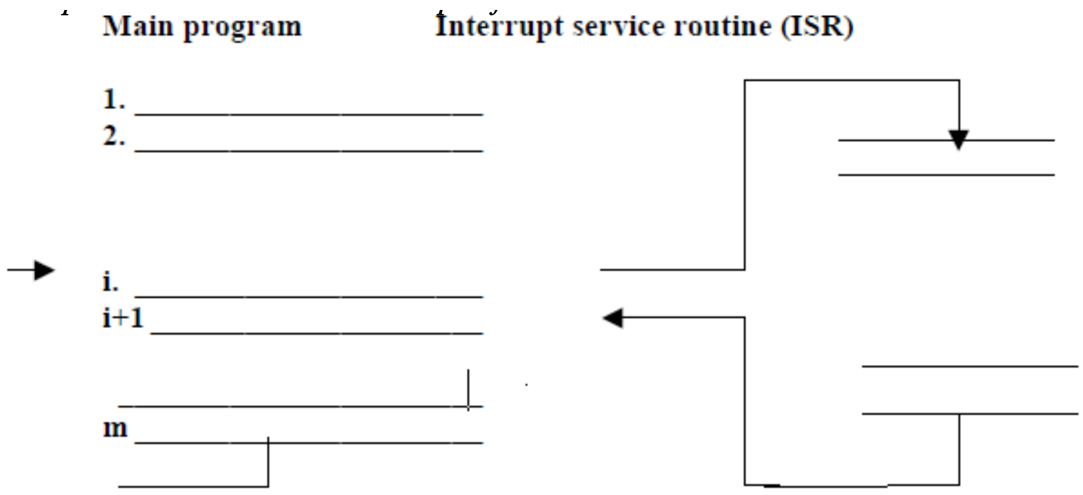
    When I/O devices and the memory share the same address space, the arrangement is called memory mapped I/O. The machine instructions that can access memory is used to trfer data to or from an I/O device.



    I/O mapped I/O:

    Here the I/O devices the memories have different address space. It has special I/O instructions. The advantage of a separate I/O address space is that I/O devices deals with fewer address lines.

13. **How does a processor handle an interrupt?**



Assume that an interrupt request arises during execution of instruction i. steps to handle interrupt by the processor is as follow:

1. Processor completes execution of instruction i
2. Processor saves the PC value, program status on to stack.
3. It loads the PC with starting address of ISR
4. After ISR is executed, the processor resumes the main program execution by reloading PC with (i+1)th instruction address.

14. What is SCSI?

Small **C**omputer System Interface, a parallel interfacestandard. SCSI interfaces provide for faster data transmission rates (up to 80 megabytes per second) than standard serial and parallel ports. In addition, you can attach many devices to a single SCSI port, so that SCSI is really an I/Obus rather than simply an interface.

15. Define USB.

Universal Serial Bus, an external bus standard that supports data transfer rates of 12 Mbps. A single USB port can be used to connect up to 127 peripheral devices, such as mice, modems, and keyboards. USB also supports Plug-and-Play installation and hot plugging.

16. What is the use of DMA? (Dec 2012)(Dec 2013)

DMA (Direct Memory Access) provides I/O transfer of data directly to and from the memory unit and the peripheral.

17. What are the units of an interface? (Dec 2012)

DATAIN, DATAOUT, SIN, SOUT

18. Distinguish between isolated and memory mapped I/O? (May 2013)

The **isolated I/O** method isolates memory and I/O addresses so that memory address values are not affected by interface address assignment since each has its own address space.

In **memory mapped I/O**, there are no specific input or output instructions. The CPU can manipulate I/O data residing in interface registers with the same instructions that are used to manipulate memory words.

19. Mention the advantages of USB. (May 2013)

The Universal Serial Bus (USB) is an industry standard developed to provide two speed of operation called low-speed and full-speed. They provide simple, low cost and easy to use interconnect system.

20. What is meant by vectored interrupt?(Dec 2013)

Vectored Interrupts are type of I/O interrupts in which the device that generates the interrupt request (also called IRQ in some text books) identifies itself directly to the processor.

21. Compare Static RAM and Dynamic RAM.(Dec 2013)

Static RAM is more expensive, requires four times the amount of space for a given amount of data than dynamic RAM, but, unlike dynamic RAM, does not need to be power-refreshed and is therefore faster to access. One source gives a typical access time as 25 nanoseconds in contrast to a typical access time of 60 nanoseconds for dynamic RAM. (More recent advances in dynamic RAM have improved access time.) Static RAM is used mainly for the level-1 and level-2 caches that the microprocessor looks in first before looking in dynamic RAM.

Dynamic RAM uses a kind of capacitor that needs frequent power refreshing to retain its charge. Because reading a DRAM discharges its contents, a power refresh is required after each read. Apart from reading, just to maintain the charge that holds its content in place, DRAM must be refreshed about every 15 microseconds. DRAM is the least expensive kind of RAM.

PART-B

1. Discuss DMA controller with block diagram.

Direct memory access (**DMA**) is a feature of computers that allows certain hardware subsystems within the computer to access system memory independently of the central processing unit (CPU).

The main idea of Direct Memory Access (DMA) is to enable peripheral devices to cut out the "middle man" role of the CPU in data transfer.

• It allows peripheral devices to transfer data directly from and to memory without the intervention of the CPU.

• Having peripheral devices access memory directly would allow the CPU to do other work, which would lead to improved performance, especially in the cases of large transfers.

• The DMA controller is a piece of hardware that controls one or more peripheral devices.

– It allows devices to transfer data to or from the system's memory without the help of the processor.

The following steps summarize the DMA Operations:

– DMA Controller initiates data transfer.

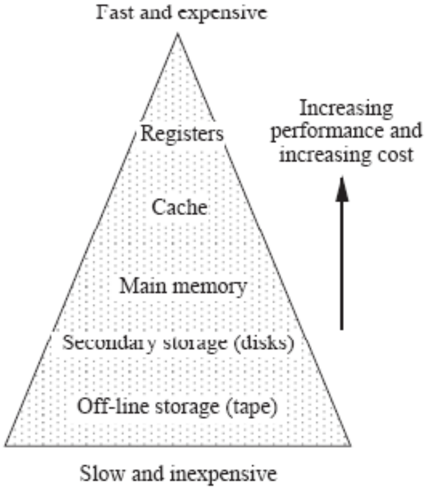
– Data is moved (increasing the address in memory, and reducing the count of words to be moved).

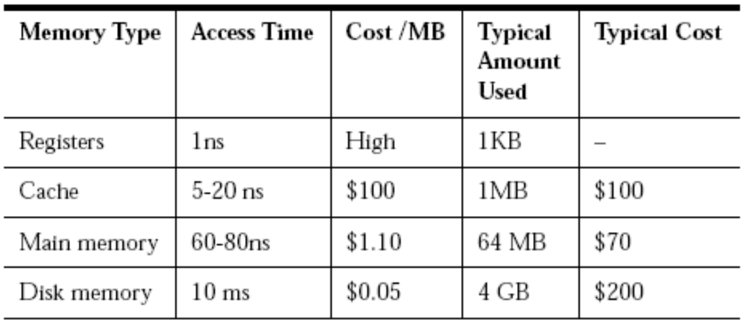
– When word count reaches zero, the DMA informs the CPU of the termination by means of an interrupt.

– The CPU regains access to the memory bus.

2. Explain the need for cache memory and discuss the different types of mapping functions with necessary block diagram. (Dec 2012) (May 2013)(Dec 2013)

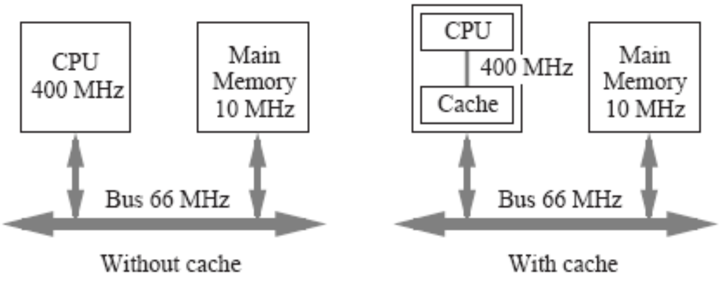
Cache memory has the fastest access time after registers.





Memory Type	Access Time	Cost /MB	Typical Amount Used	Typical Cost
Registers	1ns	High	1KB	–
Cache	5-20 ns	$100	1MB	$100
Main memory	60-80ns	$1.10	64 MB	$70
Disk memory	10 ms	$0.05	4 GB	$200

- When a program references a memory location, it is likely to reference that same memory location again soon.
- A memory location that is near a recently referenced location is more likely to be referenced then a memory location that is farther away.
- A small but fast cache memory, in which the contents of the most commonly accessed locations are maintained, can be placed between the CPU and the main memory.
- When a program executes, the cache memory is searched first.
- A cache memory has fewer locations than a main memory, which reduces the access time
- The cache is placed both physically closer and logically closer the the CPU than the main memory
- This cache-less computer usually needs a few bus cycles to synchronize the CPU with the bus.
- A cache memory can be positioned closer to the CPU.



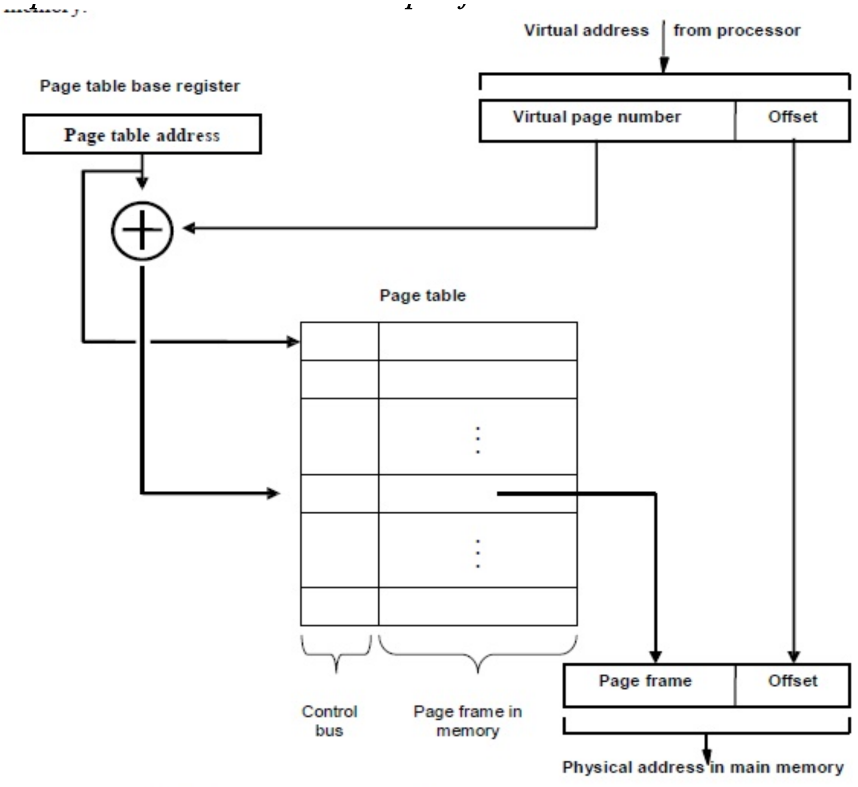
Types of Mapping

The three different types of mapping used for the purpose of cache memory are as follow, Associative mapping, Direct mapping and Set-Associative mapping.

3. Discuss the steps involved in the address translation of virtual memory with necessary block diagram. (Dec 2012)(Dec 2013)

- The process of translating a virtual address into physical address is known as address translation. It can be done with the help of MMU.
- A simple method for translating virtual addresses into physical addresses is to assume that all programs and data are composed of fixed-length units called *pages*, each of which consists of a block of words that occupy contiguous locations in the main memory.
- Pages commonly range from 2K to 16K bytes in length. They constitute the basic unit of information that is moved between the main memory and the disk whenever the translation mechanism determines that a move is required. Pages should not be too small, because the access time of a magnetic disk is much longer (several milliseconds) than the access time of the main memory.

    - A small cache, usually called the *Translation Lookaside Buffer* (TLB) is incorporated into the MMU. The operation of the TLB with respect to the page table in the main memory is essentially the same as the operation of cache memory; the TLB must also include the virtual address of the entry. When a program generates an access request to a page that is not in the main memory, a *page fault* is said to have occurred. It is essential to ensure that the interrupted task can continue correctly when it resumes execution. A page fault occurs when some instruction accesses a memory operand that is not in the main memory, resulting in an interruption before the execution of this instruction is completed. Hence, when the task resumes, either the execution of the interrupted instruction must continue from the point of interruption, or the instruction must be restarted.



4.Draw the block diagrams of two types of DRAMs and explain. (May 2013)

Cells that do not retain their state indefinitely; hence, they are called dynamic RAMs (DRAMs).Information is stored in a dynamic memory cell in the form of a charge on a capacitor, and this charge can be maintained for only tens of milliseconds. Since the cell is required to store information for a much longer time, its contents must be periodically refreshed by restoring the capacitor charge to its full value. An example of a dynamic memory cell that consists of a capacitor, C, and a transistor, T, is shown in Figure 4.6. In order to store information in this cell, transistor T is turned on and an appropriate voltage is applied to the bit line. This causes a known amount of charge to be stored in the capacitor.

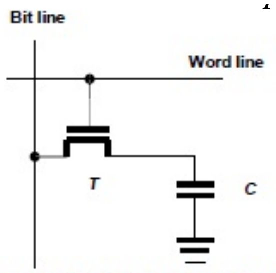


Figure 4.6 A single transistor dynamic memory cell

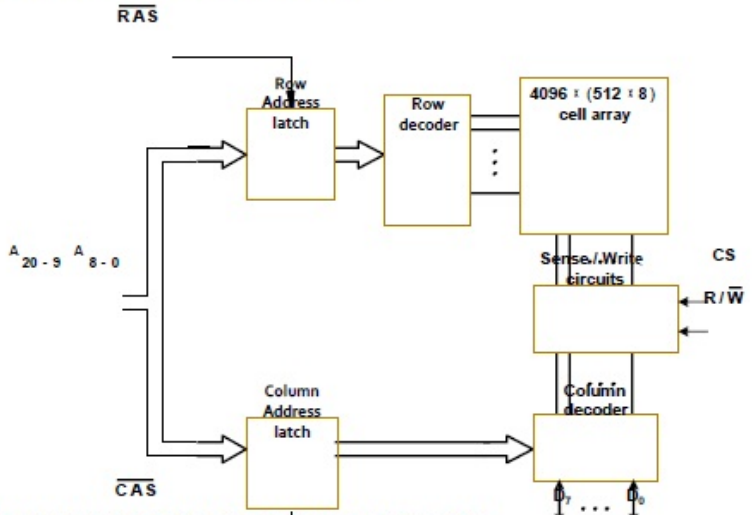


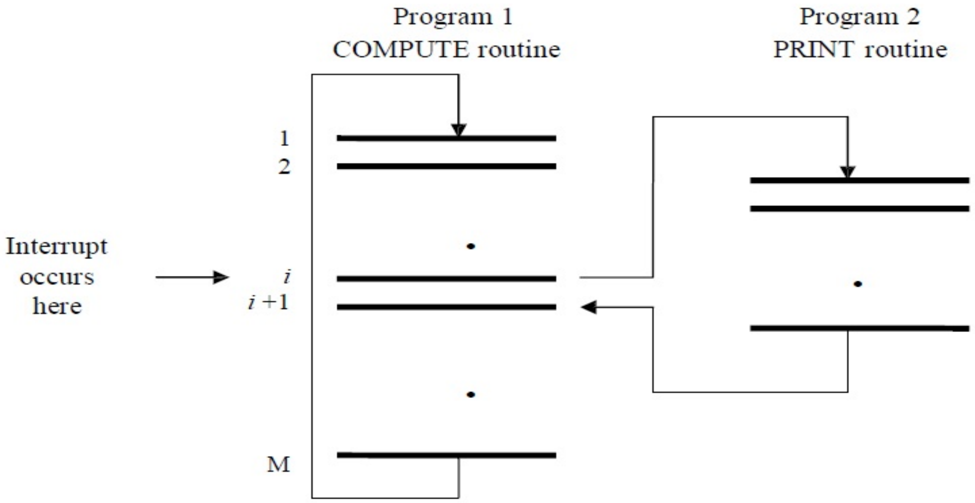
Figure 4.7 Internal Organization of a 2M x 8 dynamic memory chip

A 16-megabit DRAM chip, configured as 2M x 8, is shown in Figure 4.7. The cells are organized in the form of a 4 K x 4 K array. The 4096 cells in each row are divided into 512 groups of 8, so that a row can store 512 bytes of data. Therefore, 12 address bits are needed to select a row. Another 9 bits are needed to specify a group of 8 bits in the selected row. Thus, a 21-bit address is needed to access a byte in this memory. The high-order 12 bits and the low-order 9 bits of the address constitute the row and column addresses of a byte, respectively. To reduce the number of pins needed for external connections, the row and column addresses are multiplexed on 12 pins. During a Read or a Write operation, the row address is applied first. It is loaded into the row address latch in response to a signal pulse on the Row Address Strobe (RAS) input of the chip.

A refresh circuit usually performs this function automatically. A specialized memory controller circuit provides the necessary control signals, RAS and CAS, that govern the timing. The processor must take into account the delay in the response of the memory. Such memories are referred to as *asynchronous DRAMs*.

5. What is an interrupt? Explain the different types of interrupts and the different ways of handling the interrupts. (Dec 2012)

Interrupt is a hardware signal to the processor from I/O devices through one of the control line called interrupt-request line. The routine executed in response to an interrupt request is called the interrupt-service routine, Interrupts bear considerable resemblance to subroutine calls.

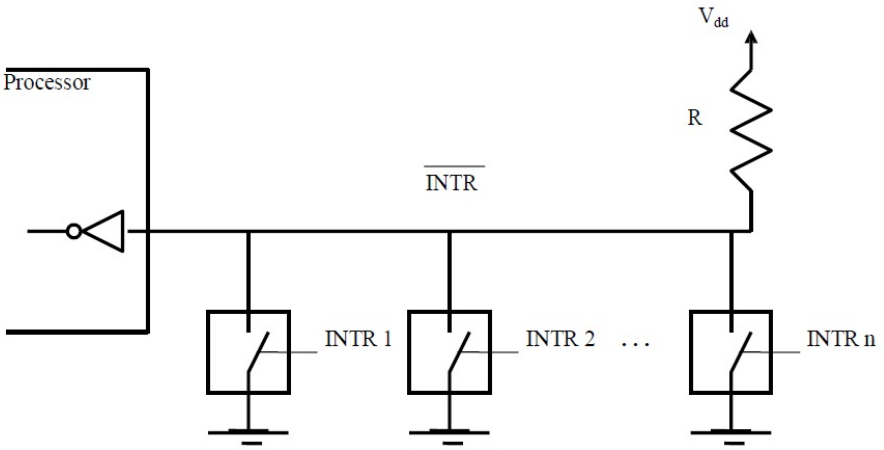


: **5.5** Transfer of control through the use of interrupts

INTERRUPT HARDWARE

A single interrupt-request line may be used to serve n devices as depicted in Figure 5.6. All devices are connected to the line via switches to ground. that is,

INTR = INTR 1 +... + INTRn



gure 5.6 An equivalent circuit for an open-drain bus used to implement a common interrupt- request ıe.

ENABLING AND DISABLING INTERRUPTS

A simple way is to provide machine instructions, such as Interrupt-enable and Interrupt- disable. The processor hardware ignores the interrupt-request line until the execution of the first instruction of the interrupt-service routine has been completed. Then, by using an Interrupt-disable instruction as the first instruction in the interrupt-service routine, the programmer can ensure that no further interruptions will occur until an Interrupt-enable instruction is executed..

Interrupts can be categorized into these different types:

- Maskable interrupt (IRQ): a hardware interrupt that may be ignored by setting a bit in an interrupt mask register's (IMR) bit-mask.
- Non-maskable interrupt (NMI): a hardware interrupt that lacks an associated bit-mask, so that it can never be ignored. NMIs are used for the highest priority tasks such as timers, especially watchdog timers.
- Inter-processor interrupt (IPI): a special case of interrupt that is generated by one processor to interrupt another processor in a multiprocessor system.
- Software interrupt: an interrupt generated within a processor by executing an instruction. Software interrupts are often used to implement system calls because they result in a subroutine call with a CPU ring level change.
- Spurious interrupt: a hardware interrupt that is unwanted. They are typically generated by system conditions such as electrical interference on an interrupt line or through incorrectly designed hardware.

6. **Explain USB interface, PCI buses..(DEC 2013)(May 2013)**

Universal Serial Bus (USB) is a set of interface specifications for high speed wired communication between electronics systems peripherals and devices with or without PC/computer. The USB was originally developed in 1995 by many of the industry leading companies like Intel, Compaq, Microsoft, Digital, IBM, and Northern Telecom.

The major goal of USB was to define an external expansion bus to add peripherals to a PC in easy and simple manner. The new external expansion architecture, highlights,

1. PC host controller hardware and software
2. Robust connectors and cable assemblies
3. Peripheral friendly master-slave protocols
4. Expandable through multi-port hubs.

The benefits of USB are low cost, expandability, auto-configuration, hot-plugging and outstanding performance. It also provides power to the bus, enabling many peripherals to operate without the added need for an AC power adapter.

Various versions USB:

As USB technology advanced the new version of USB are unveiled with time. Let us now try to understand more about the different versions of the USB.

USB1.0,USB1.1,USB2.0,USB3.0

USB can support four data transfer types or transfer mode, which are listed below.

1. Control
2. Isochronous

3. Bulk
4. Interrupt

USB packets may consist of the following fields:
1. Sync field
2. PID field
3. ADDR field
4. ENDP field
5. CRC field
6. EOP field

PCI (Peripheral Component Interconnect)

PCI (Peripheral Component Interconnect) is an interconnection system between a microprocessor and attached devices in which expansion slots are spaced closely for high speed operation. Using PCI, a computer can support both new PCI cards while continuing to support Industry Standard Architecture (ISA) expansion cards, an older standard. Designed by Intel, the original PCI was similar to the VESA Local Bus. However, PCI 2.0 is no longer a local bus and is designed to be independent of microprocessor design. PCI is designed to be synchronized with the clock speed of the microprocessor.

Today's computers and motherboards have replaced PCI with PCI Express (PCIe) slots.

Examples of PCI devices

- Modem
- Network card
- Sound card
- Video card

PCI device drivers

If you are looking for PCI drivers, you most likely need to get the drivers for the installed PCI device. For example, if you need a PCI Ethernet adapter driver you need the drivers for the PCI network card.

7.Explain the need for memory hierarchy technology with a four-level memory(Dec 2013)

Storage devices such as registers, caches, main memory, disk devices, and tape units are often organized as a hierarchy as depicted in the Figure 5.4.1(a). The memory technology and storage organization at each level are characterized by five parameters: the access time (t i), memory size (s i), cost per byte (c i), transfer bandwidth (b i), and unit of transfer (x i).

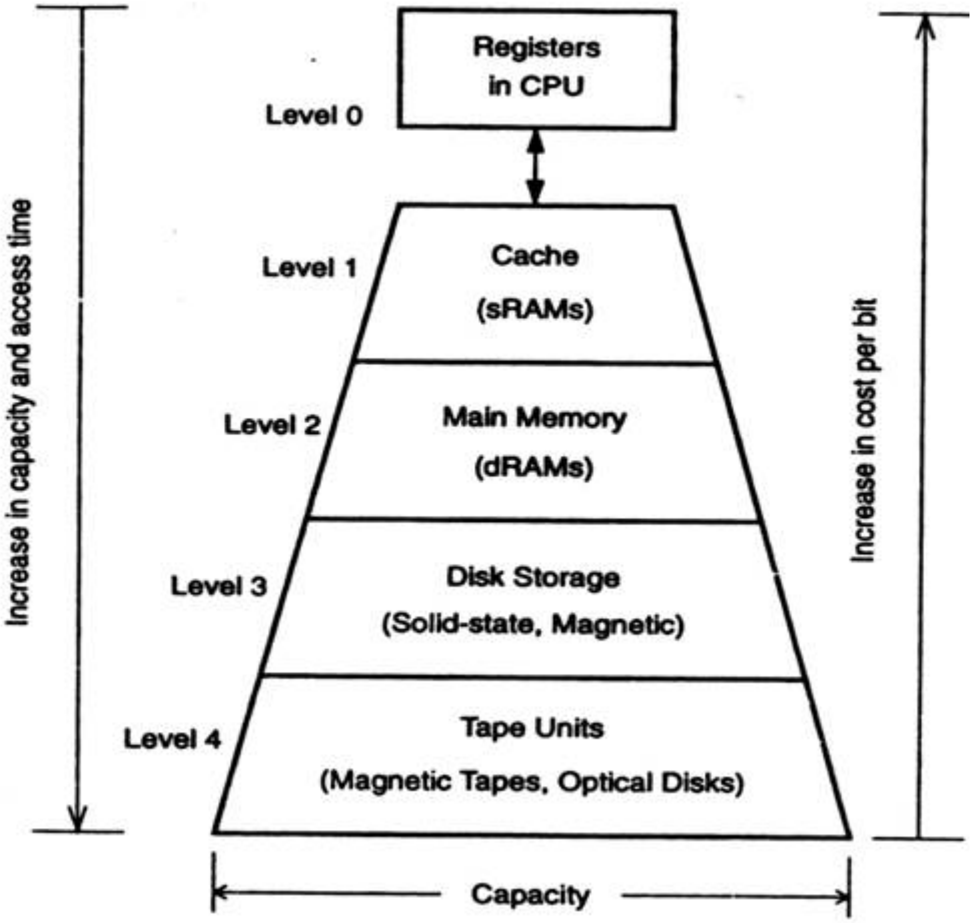
Registers and Caches:

The register and the cache are parts of the processor complex, built either on the processor chip or on the processor board. Register assignment is often made by the compiler. Register transfer operations are directly controlled by the processor after instructions are decoded. Register transfer is conducted at processor speed, usually in one clock cycle.

Main Memory:

The main memory is sometimes called the primary memory of a computer system. It is usually much larger than the cache and often implemented by the most cost-effective RAM chips, such as the 4-Mbit DRAMs used in 1991 and the 64-Mbit DRAMs projected in 1995.

The main memory is managed by a MMU in cooperation with the operating system. Options are often provided to extend the main memory by adding more memory boards to a system. Sometimes, the main memory is itself divided into two sublevels using different memory technologies.



Disk Drives and Tape Units:

Disk drives and tape units are handled by the OS with limited user intervention. The disk storage is considered the highest level of on-line memory. It holds the system programs such as the OS and compilers and some user programs and their data sets. The magnetic tape units are off-line memory for use as backup storage. They hold copies of present and past user programs and processed results and files.

Peripheral Technology:

Besides disk drives and tape units, peripheral devices include printers, plotters, terminals, monitors, graphics displays, optical scanners, image digitizers, output microfilm devices etc. **Inclusion, Coherence, and Locality**

Information stored in a memory hierarchy (M 1, M 2, …, M n) satisfies three important properties: inclusion, coherence, and locality.

Hit Ratios:

Hit ratio is a concept defined for any two adjacent levels of a memory hierarchy. When an information item is found in M i, we call it a hit, otherwise, a miss. The access frequency to M i is defined as f i = (1-h 1) (1-h 2)…(1-h i-1)h i.

Effective Access Time

Every time a miss occurs, a penalty must be paid to access the next higher level of memory. The misses have been called block misses in the cache and page faults in the main memory because blocks and pages are the units of transfer between these levels.

The Design of a Memory Hierarchy:

Consider the design of a three-level memory hierarchy with the following specifications for memory characteristics:

Memory level	Access time	Capacity	Cost/Kbyte
Cache	t 1 = 25 ns	s 1 = 512 Kbytes	c 1 = $1.25
Main memory	t 2 = unknown	s 2 = 32 Mbytes	c 2 = $0.2
Disk array	t 3 = 4 ms	s 3 = unknown	c 3 = $0.0002

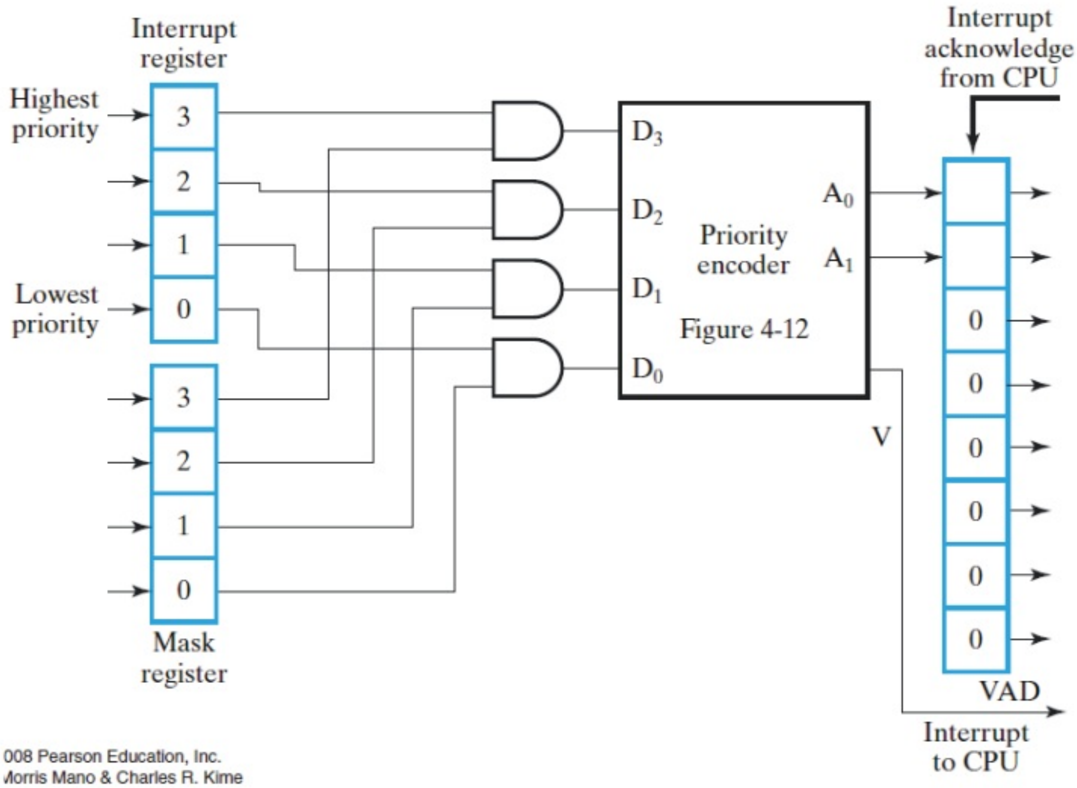
8.Design a parallel priority interrupt hardware for a system with eight interrupt sources and explain.

Parallel Priority Interrupt: The parallel priority interrupt method uses a register whose bits are set separately by the interrupt signal from each device. Priority is established according to the position of the bits in the register. The circuit may include a mask register whose purpose is to control the status of each interrupt request in addition to the interrupt register. The mask register can be programmed to disable lower-priority interrupts while a higher priority device is being serviced. It also provides a facility that allows a high priority device to interrupt the CPU while a lower-priority device is being serviced.

The Priority logic for a system of four interrupt sources is demonstrated in the previous figure. It consists of an interrupt register whose individual bits are set by external conditions and cleared by program instructions. The magnetic disk being a high-speed device, is given the highest priority. The printer has the next priority followed by a character reader and a keyboard. The mask register has the same number of bits as the interrupt register.

By means of program instructions, it is possible to set or reset any bit in the mask register. Each interrupt bit and its corresponding mask bit are applied to an AND gate to produce the four inputs to a priority encoder.

In this way an interrupt is recognized only if its corresponding mask bit is set to 1 by the program. The priority encoder generates two bits of the vector address, which is transferred to the CPU.



008 Pearson Education, Inc.
Morris Mano & Charles R. Kime

Figure 12-17 Parallel Priority Interrupt Hardware

The parallel priority interrupt method employs two registers to manage interrupts.

•The Interrupt Register is used to record which devices require service. Devices are prioritized by the position of the bits in the register used to record a request for service. Bit position 3 identifies the device having the highest priority and bit position 0 is set by the device having the lowest priority.

•The Mask Register can be programmed to disable lower priority devices while a higher priority device is being serviced. Service for a lower priority device may be interrupted to serve a higher priority device by employing the Mask Register.

•The Priority Encoder produces two outputs, signal Vand the least significant two bits of the VAD. Signal Vis asserted when service is requested

. The least significant two bits of the VAD designate the location of the service routine associated with the device making the request.

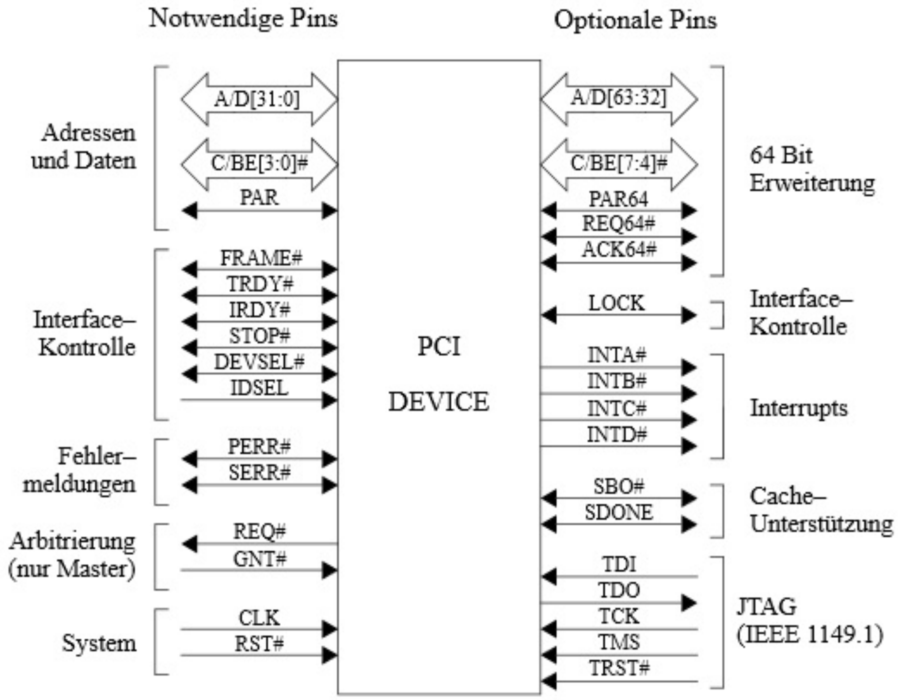
9.Write and explain the working of Peripheral Component Interface Bus. (Dec 2012)

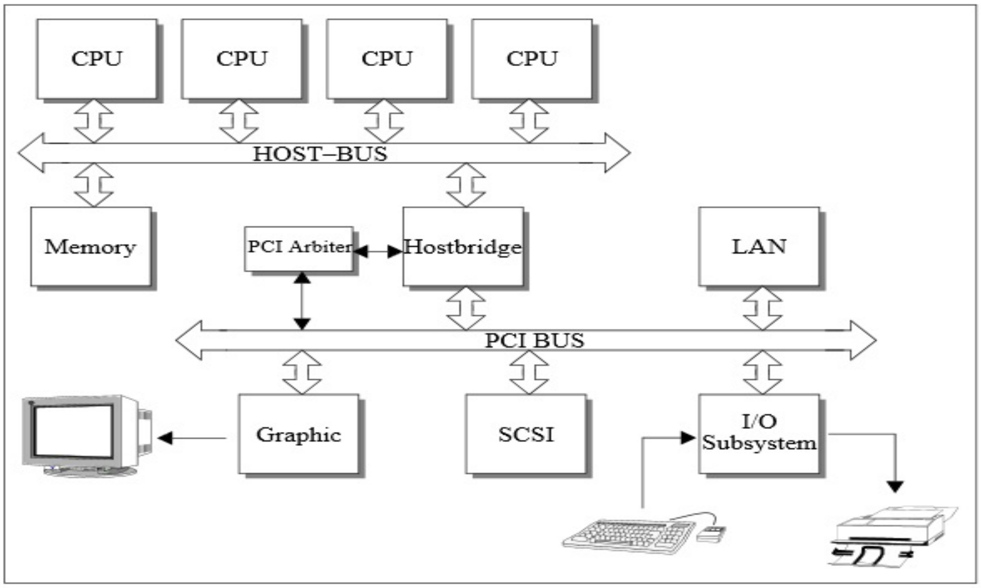
PCI - Peripheral Component Interface

Peripheral Component Interconnect is a computer bus for attaching hardware devices in a computer. These devices can take either the form of an integrated circuit fitted onto the motherboard itself, called a planar device in the PCI specification, or an expansion card that fits into a slot. The PCI specification covers the physical size of the bus (including the size and spacing of the circuit board edge electrical contacts), electrical characteristics, bus timing, and protocols. The specification can be purchased from the PCI Special Interest Group (PCI-SIG).

Typical PCI cards used in PCs include: network cards, sound cards, modems, extra ports such as USB or serial, TV tuner cards and disk controllers.

PCI - Peripheral Component Interface





10. Write short note on I/O processor. What is the need for an I/O interface? Describe the functions of SCSI interface with a neat diagram.(DEC 2013)

The IOP attaches to the system I/O bus and one or more input/output adapters (IOAs). The IOP processes instructions from the system and works with the IOAs to control the I/O devices.

There are many different kinds of IOPs.

- Some IOPs can only support one type of I/O device. In this case the IOA is embedded in the IOP so you can not remove the IOA or change it.
- Some IOPs can support multiple device types, but only one at a time. The type of IOA that is attached determines what device can be used. IOAs on these IOPs can be changed with another IOA to support a different I/O device.
- Some IOPs can support multiple types of I/O devices at the same time. These are known as MFIOPs or CFIOPs (this depends on the type of IOP).
- There are several important I/O devices in the system. These include the load source disk unit, the alternate IPL device, the console, and the electronic customer support hardware. The system needs to know where to locate these special devices on secondary partitions. When you create a logical partition, you need to identify the IOPs that control these important devices:
- The IOP that controls the disk unit that will be the load source.
- The IOP that controls the console.
- The IOP that controls the alternate IPL device.
- The IOP that controls the electronic customer support line.

I/O processor (IOP) A specialized computer that permits autonomous handling of data between I/O devices and a central computer or the central memory of the computer. It can be a programmable computer in its own right; in earlier forms, as a wired-program computer, it was called a channel controller.

The **input/output processor** or **io processor** is a processor that is separate from the main Processor or CPU designed to handle only input/output processes for a device or the computer.

INPUT/OUTPUT PROCESSOR For those computers that have an I/O processor, the physical organization of I/O is similar to the other major functional areas: CPU and memory. I/O processors can vary from many pcb's that makeup a module/unit to a single pcb.

Larger mainframe computers use the modular arrangement: multiple components on multiple pcb's that comprise one or more modules or units. Mini- and microcomputers use chassis or assemblies, cages or racks, and motherboard/backplane arrangements.

INTERFACE

An I/O interface is required whenever the I/O device is driven by the processor. The interface must have necessary logic to interpret the device address generated by the processor.

Handshaking should be implemented by the interface using appropriate commands (like BUSY, READY, and WAIT), and the processor can communicate with an I/O device through the interface. If different data formats are being exchanged, the interface must be able to convert serial data to parallel form and vice-versa. There must be provision for generating interrupts and the corresponding type numbers for further processing by the processor if required.

Input Output Interface:

Input-output interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit. The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral. The major differences are:

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.
2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently, a synchronization mechanism may be needed.
3. Data codes and formats in peripherals differ from the word format in the CPU and memory.

4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

To resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface between the processor bus and the peripheral device.

SCSI INTERFACE

SCSI is available in a variety of interfaces. The first, still very common, was parallel SCSI (now also called SPI), which uses a parallel bus design.

SCSI interfaces have often been included on computers from various manufacturers for use under Microsoft Windows, Mac OS, Unix, Commodore Amiga and Linux operating systems, either implemented on the motherboard or by the means of plug-in adaptors.

Short for Small Computer System Interface, SCSI is pronounced as "Scuzzy" and is one of the most commonly used interface for disk drives that was first completed in 1982.

SCSI-1 is the original SCSI standard developed back in 1986 as ANSI X3.131-1986. SCSI-1 is capable of transferring up to eight bits a second.

SCSI-2 was approved in 1990, added new features such as Fast and Wide SCSI, and support for additional devices.

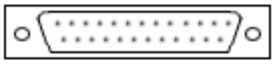
SCSI-3 was approved in 1996 as ANSI X3.270-1996.

SCSI is a standard for parallel interfaces that transfers information at a rate of eight bits per second and faster, which is faster than the average parallel interface. SCSI-2 and above supports up to seven peripheral devices, such as a hard drive, CD-ROM, and scanner, that can attach to a single SCSI port on a system's bus. SCSI ports were designed for Apple Macintosh and Unix computers, but also can be used with PCs. Although SCSI has been popular in the past, today many users are switching over to SATA drives.

SCSI connectors

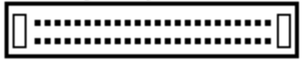
The below illustrations are examples of some of the most commonly found and used SCSI connectors on computers and devices and illustrations of each of these connections.

DB-25 Male External



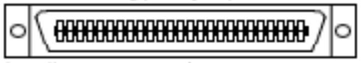
http://www.computerhope.com

Low-Density, 50-pin, Male Internal



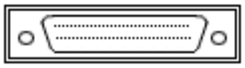
http://www.computerhope.com

Low-Density, 50-pin, Male External

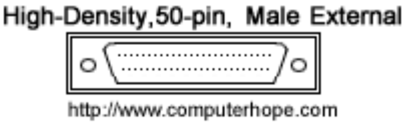


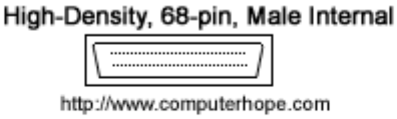
http://www.computerhope.com

High-Density, 68-pin, Male External



http://www.computerhope.com


High-Density,50-pin, Male External
http://www.computerhope.com


High-Density, 68-pin, Male Internal
http://www.computerhope.com