VLSI interview questions

Metastability/ reset

What is metastability?

- When setup or hold window is violated in an flip flop then signal attains a unpredictable value or state known as metastability.

What is MTBF? What it signifies?

- MTBF-Mean Time Before Failure
- Average time to next failure

How chance of metastable state failure can be reduced?

- Lowering clock frequency
- Lowering data speed
- Using faster flip flop

What are the advantages of using synchronous reset ?

- No metastability problem with synchronous reset (provided recovery and removal time for reset is taken care).
- Simulation of synchronous reset is easy.

What are the disadvantages of using synchronous reset ?

- Synchronous reset is slow.
- Implementation of synchronous reset requires more number of gates compared to asynchronous reset design.
- An active clock is essential for a synchronous reset design. Hence you can expect more power consumption.

What are the advantages of using asynchronous reset ?

- Implementation of asynchronous reset requires less number of gates compared to synchronous reset design.
- Asynchronous reset is fast.
- Clocking scheme is not necessary for an asynchronous design. Hence design consumes less power. Asynchronous design style is also one of the latest design options to achieve low power. Design community is scrathing their head over asynchronous design possibilities.

What are the disadvantages of using asynchronous reset ?

- Metastability problems are main concerns of asynchronous reset scheme (design).
- Static timing analysis and DFT becomes difficult due to asynchronous reset.

What are the 3 fundamental operating conditions that determine the delay characteristics of gate?
How operating conditions affect gate delay?

- Process
- Voltage
- Temperature
- Click here to read more.

Is verilog/VHDL is a concurrent or sequential language?

- Verilog and VHDL both are concurrent languages.
- Any hardware descriptive language is concurrent in nature.

In a system with insufficient hold time, will slowing down the clock frequency help?

- No.
- Making data path slower can help hold time but it may result in setup violation.

In a system with insufficient setup time, will slowing down the clock frequency help?

- Yes.
- Making data path faster can also help setup time but it may result in hold violation.

Latch vs. Flip Flop

What is the difference between a latch and a flip-flop?

- Both latches and flip-flops are circuit elements whose output depends not only on the present inputs, but also on previous inputs and outputs.
- They both are hence referred as "sequential" elements.
- In electronics, a latch, is a kind of bistable multi vibrator, an electronic circuit which has two stable states and thereby can store one bit of of information. Today the word is mainly used for simple transparent storage elements, while slightly more advanced non-transparent (or clocked) devices are described as flip-flops. Informally, as this distinction is quite new, the two words are sometimes used interchangeably. [wiki]
- In digital circuits, a flip-flop is a kind of bistable multi vibrator, an electronic circuit which has two stable states and thereby is capable of serving as one bit of memory. Today, the term flip-flop has come to generally denote non-transparent (clocked or edge-triggered) devices, while the simpler transparent ones are often referred to as latches.[wiki]
- A flip-flop is controlled by (usually) one or two control signals and/or a gate or clock signal.

- Latches are level sensitive i.e. the output captures the input when the clock signal is high, so as long as the clock is logic 1, the output can change if the input also changes.
- Flip-Flops are edge sensitive i.e. flip flop will store the input only when there is a rising or falling edge of the clock.
- A positive level latch is transparent to the positive level(enable), and it latches the final input before it is changing its level(i.e. before enable goes to '0' or before the clock goes to -ve level.)
- A positive edge flop will have its output effective when the clock input changes from '0' to '1' state ('1' to '0' for negative edge flop) only.

- Latches are faster, flip flops are slower.
- Latch is sensitive to glitches on enable pin, whereas flip-flop is immune to glitches.
- Latches take less gates (less power) to implement than flip-flops.
- D-FF is built from two latches. They are in master slave configuration.
- Latch may be clocked or clock less. But flip flop is always clocked.
- For a transparent latch generally D to Q propagation delay is considered while for a flop clock to Q and setup and hold time are very important.

Synthesis perspective: Pros and Cons of Latches and Flip Flops

- In synthesis of HDL codes inappropriate coding can infer latches instead of flip flops. Eg.:"if" and "case" statements. This should be avoided sa latches are more prone to glitches.
- Latch takes less area, Flip-flop takes more area (as flip flop is made up of latches) .
- Latch facilitate time borrowing or cycle stealing whereas flip flops allow synchronous logic.
- Latches are not friendly with DFT tools. Minimize inferring of latches if your design has to be made testable. Since enable signal to latch is not a regular clock that is fed to the rest of the logic. To ensure testability, you need to use OR gate using "enable"� and "scan_enable" signals as input and feed the output to the enable port of the latch. [ref]
- Most EDA software tools have difficulty with latches. Static timing analyzers typically make assumptions about latch transparency. If one assumes the latch is transparent (i.e.triggered by the active time of clock,not triggered by just clock edge), then the tool may find a false timing path through the input data pin. If one assumes the latch is not transparent, then the tool may miss a critical path.

- If target technology supports a latch cell then race condition problems are minimized. If target technology does not support a latch then synthesis tool will infer it by basic gates which is prone to race condition. Then you need to add redundant logic to overcome this problem. But while optimization redundant logic can be removed by the synthesis tool ! This will create endless problems for the design team.

- Due to the transparency issue, latches are difficult to test. For scan testing, they are often replaced by a latch-flip-flop compatible with the scan-test shift-register. Under these conditions, a flip-flop would actually be less expensive than a latch. Read a good article on problems of latch published in eetimes long back !!

- Flip flops are friendly with DFT tools. Scan insertion for synchronous logic is hassle free.

Intel Questions:

1. Why power stripes routed in the top metal layers?
2. Why do you use alternate routing approach HVH/VHV (Horizontal-Vertical-Horizontal/ Vertical-Horizontal-Vertical)?
3. What are several factors to improve propagation delay of standard cell?
4. How do you compute net delay (interconnect delay) / decode RC values present in tech file?
5. What are various ways of timing optimization in synthesis tools?
6. What would you do in order to not use certain cells from the library?
7. How delays are characterized using WLM (Wire Load Model)?
8. What are various techniques to resolve congestion/noise?
9. Let's say there enough routing resources available, timing is fine, can you increase clock buffers in clock network? If so will there be any impact on other parameters?
10. How do you optimize skew/insertion delays in CTS (Clock Tree Synthesis)?
11. What are pros/cons of latch/FF (Flip Flop)?
12. How you go about fixing timing violations for latch- latch paths?
13. As an engineer, let's say your manager comes to you and asks for next project die size estimation/projection, giving data on RTL size, performance requirements. How do you go about the figuring out and come up with die size considering physical aspects?
14. How will you design inserting voltage island scheme between macro pins crossing core and are at different power wells? What is the optimal resource solution?
15. What are various formal verification issues you faced and how did you resolve?
16. How do you calculate maximum frequency given setup, hold, clock and clock skew?
17. What are effects of metastability?
18. Consider a timing path crossing from fast clock domain to slow clock domain. How do you design synchronizer circuit without knowing the source clock frequency?
19. How to solve cross clock timing path?
20. How to determine the depth of FIFO/ size of the FIFO?

Design Questions

1. If inverted output of D flip-flop is connected to its input how the flip-flop behaves?
2. Design a circuit to divide input frequency by 2?
3. Design a divide by two counter using D-Latch.
4. Design a divide-by-3 sequential circuit with 50% duty cycle.

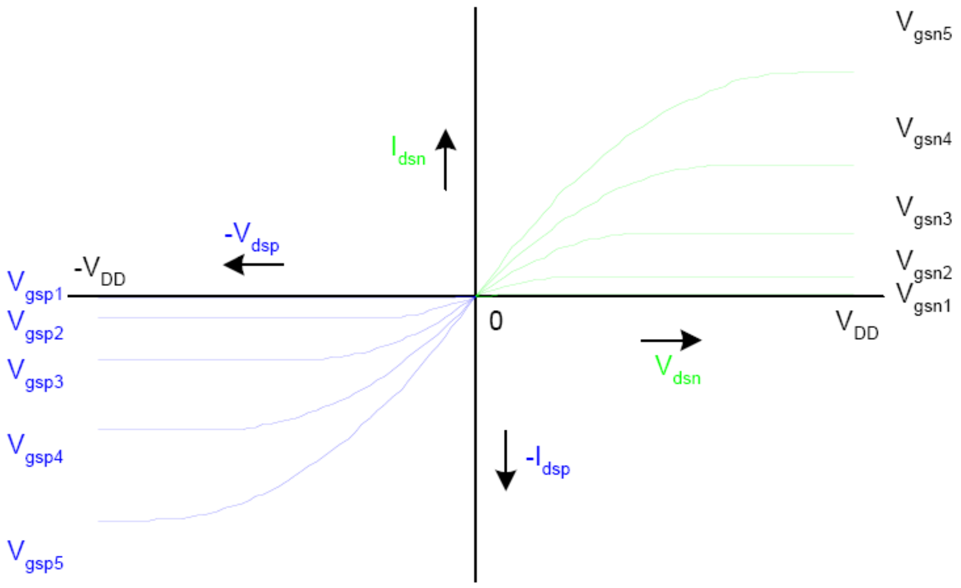
5. What are the different types of adder implementation?
6. Draw a Transmission Gate-based D-Latch?
7. Give the truth table for a Half Adder. Give a gate level implementation of the same.
8. Design an OR gate from 2:1 MUX.
9. What is the difference between a LATCH and a FLIP-FLOP?
10. Design a D Flip-Flop from two latches.
11. Design a 2 bit counter using D Flip-Flop.
12. What are the two types of delays in any digital system
13. Design a Transparent Latch using a 2:1 Mux.
14. Design a 4:1 Mux using 2:1 Mux's.
15. What is metastable state? How does it occur?
16. What is metastablity?
17. Design a 3:8 decoder
18. Design a FSM to detect sequence "101" in input sequence
19. Convert NAND gate into Inverter in two different ways.
20. Design a D and T flip flop using 2:1 mux only.
21. Design D Latch from SR flip-flop.
22. Define Clock Skew, Negative Clock Skew, Positive Clock Skew?
23. What is race condition? How it occurs? How to avoid it?
24. Design a 4 bit Gray Counter?
25. Design 4-bit synchronous counter, asynchronous counter?
26. Design a 16 byte asynchronous FIFO?
27. What is the difference between a EEPROM and FLASH?
28. What is the difference between a NAND-based Flash and NOR-based Flash?
29. Which one is good: asynchronous reset or synchronous reset? Why?
30. Design a simple circuit based on combinational logic to double the output frequency.
31. What is the difference between flip-flop and latch?
32. Implement comparator using combinational logic, that compares two 2-bit numbers A and B. The comparator should have 3 outputs: A > B, A < a =" B.">
33. Give two ways of converting a two input NAND gate to an inverter?
34. What is the difference between mealy and moore state-machines?
35. What is the difference between latch based design and flip-flop based design?
36. What is metastability and how to prevent it?
37. Design a four-input NAND gate using only two-input NAND gates.
38. Why are most interrupts active low?
39. How do you detect if two 8-bit signals are same?
40. 7 bit ring counter's initial state is 0100010. After how many clock cycles will it return to the initial state?
41. Design all the basic gates NOT, AND, OR, NAND, NOR, XOR, XNOR using 2:1 Multiplexer.
42. How will you implement a full subtractor from a full adder?
43. In a 3-bit Johnson's counter what are the unused states?
44. What is difference between RAM and FIFO?
45. What is an LFSR? List a few of its industry applications.
46. Implement the following circuits:
    (a) 3 input NAND gate using minimum number of 2 input NAND gates

(b) 3 input NOR gate using minimum number of 2 input NOR gates
(c) 3 input XNOR gate using minimum number of 2 input XNOR gates assuming 3 inputs A,B,C?

47. Design a D-latch using (a) using 2:1 Mux (b) from S-R Latch?
48. How to implement a Master Slave flip flop using a 2 to 1 mux?
49. How many 2 input xor's are needed to inplement 16 input parity generator?
50. Convert xor gate to buffer and inverter.
51. Difference between onehot and binary encoding?
52. What are different ways to synchronize between two clock domains?
53. How to calculate maximum operating frequency?
54. How to find out longest path?
55. How to achieve 180 degree exact phase shift?
56. What is significance of ras and cas in SDRAM?
57. Tell some of applications of buffer?
58. Implement an AND gate using mux?
59. What will happen if contents of register are shifter left, right?
60. What is the basic difference between analog and digital design?
61. What advantages do synchronous counters have over asynchronous counters?
62. What types of flip-flops can be used to implement the memory elements of a counter?
63. What are the advantages of using a microprocessor to implement a counter rather than the conventional method (flip-flop and logic gates)?
64. What is the principal advantage of Gray Code over straight (conventional) binary?
65. What does Pipelining do?
66. Design divide by 2, divide by 3 circuit with equal duty cycle.
67. How many 4:1 mux do you need to design a 8:1 mux?
68. What is D-Word, Q-word?
69. Define Moore, Mealy state machines. Which one is good for timing?
70. Design a FSM to detect 10110. What is the minimum number of flops required?
71. Design a simple circuit based on combinational logic to double the output frequency.
72. Design a 2bit up/down counter with clear using gates. (No verilog or vhdl)
73. Design a finite state machine to give a modulo 3 counter when x=0 and modulo 4 counter when x=1.
74. Minimize: S= A' + AB
75. What is the function of a D-flipflop, whose inverted outputs are connected to its input?
76. How to synchronize control signals and data between two different clock domains?
77. Describe a finite state machine that will detect three consecutive coin tosses (of one coin) that results in heads.
78. In what cases do you need to double clock a signal before presenting it to a synchronous state machine?
79. How many bit combinations are there in a byte?
80. What are the different Adder circuits you studied?
81. Give the truth table for a Half Adder. Give a gate level implementation of the same.
82. Convert 65(Hex) to Binary
83. Convert a number to its two's compliment and back.
84. What is the 1's and 2's complement of the decimal number 25.
85. If A?B=C and C?A=B then what is the boolean operator ?

VLSI Design:

1) Explain why & how a MOSFET works
2) Draw Vds-Ids curve for a MOSFET.



   i. Now, show how this curve changes (a) with increasing Vgs (b) with increasing transistor width (c) considering Channel Length Modulation

3) Explain the various MOSFET Capacitances & their significance

4) Draw a CMOS Inverter. Explain its transfer characteristics
5) Explain sizing of the inverter

- Use equivalent circuits for MOS transistors
- Ideal switch + capacitance and ON resistance
  - Unit nMOS has resistance R, capacitance C
  - Unit pMOS has resistance 2R, capacitance C
  - Capacitance proportional to width
  - Resistance inversely proportional to width

6) How do you size NMOS and PMOS transistors to increase the threshold voltage?
7) What is Noise Margin? Explain the procedure to determine Noise Margin
8) Give the expression for CMOS switching power dissipation
9) What is Body Effect?
10) Describe the various effects of scaling
11) Give the expression for calculating Delay in CMOS circuit
12) What happens to delay if you increase load capacitance?
13) What happens to delay if we include a resistance at the output of a CMOS circuit?
14) What are the limitations in increasing the power supply to reduce delay?
15) How does Resistance of the metal lines vary with increasing thickness and increasing length?

16) You have three adjacent parallel metal lines. Two out of phase signals pass through the outer two metal lines. Draw the waveforms in the center metal line due to interference. Now, draw the signals if the signals in outer metal lines are in phase with each other
17) What happens if we increase the number of contacts or via from one metal layer to the next?
18) Draw a transistor level two input NAND gate. Explain its sizing (a) considering Vth (b) for equal rise and fall times
19) Let A & B be two inputs of the NAND gate. Say signal A arrives at the NAND gate later than signal B. To optimize delay, of the two series NMOS inputs A & B, which one would you place near the output?
20) Draw the stick diagram of a NOR gate. Optimize it
21) For CMOS logic, give the various techniques you know to minimize power consumption
22) What is Charge Sharing? Explain the Charge Sharing problem while sampling data from a Bus
23) Why do we gradually increase the size of inverters in buffer design? Why not give the output of a circuit to one large inverter?
24) In the design of a large inverter, why do we prefer to connect small transistors in parallel (thus increasing effective width) rather than lay out one transistor with large width?
25) Given a layout, draw its transistor level circuit. (I was given a 3 input AND gate and a 2 input Multiplexer. You can expect any simple 2 or 3 input gates)
26) Give the logic expression for an AOI gate. Draw its transistor level equivalent. Draw its stick diagram
27) Why don't we use just one NMOS or PMOS transistor as a transmission gate?
28) For a NMOS transistor acting as a pass transistor, say the gate is connected to VDD, give the output for a square pulse input going from 0 to VDD
29) Draw a 6-T SRAM Cell and explain the Read and Write operations
30) Draw the Differential Sense Amplifier and explain its working. Any idea how to size this circuit? (Consider Channel Length Modulation)
31) What happens if we use an Inverter instead of the Differential Sense Amplifier?
32) Draw the SRAM Write Circuitry
33) Approximately, what were the sizes of your transistors in the SRAM cell? How did you arrive at those sizes?
34) How does the size of PMOS Pull Up transistors (for bit & bit- lines) affect SRAM's performance?
35) What's the critical path in a SRAM?
36) Draw the timing diagram for a SRAM Read. What happens if we delay the enabling of Clock signal?
37) Give a big picture of the entire SRAM Layout showing your placements of SRAM Cells, Row Decoders, Column Decoders, Read Circuit, Write Circuit and Buffers
38) In a SRAM layout, which metal layers would you prefer for Word Lines and Bit Lines? Why?
39) How can you model a SRAM at RTL Level?
40) What's the difference between Testing & Verification?

41) For an AND-OR implementation of a two input Mux, how do you test for Stuck-At-0 and Stuck-At-1 faults at the internal nodes? (You can expect a circuit with some redundant logic)
42) What is Latch Up? Explain Latch Up with cross section of a CMOS Inverter. How do you avoid Latch Up?

Digital Design:

1. Give two ways of converting a two input NAND gate to an inverter
2. Given a circuit, draw its exact timing response. (I was given a Pseudo Random Signal Generator; you can expect any sequential ckt)
3. What are set up time & hold time constraints? What do they signify? Which one is critical for estimating maximum clock frequency of a circuit?

Setup violations are related to two edges of clock, i mean you can vary the clock frequency to correct setup violation. But for hold time, you are only concerned with one edge and does not basically depend on clock frequency.

set up time: - the amount of time the data should be stable before the application of the clock signal, where as the hold time is the amount of time the data should be stable after the application of the clock. Setup time signifies maximum delay constraints; hold time is for minimum delay constraints. Setup time is critical for establishing the maximum clock frequency.

4. Give a circuit to divide frequency of clock cycle by two
5. Design a divide-by-3 sequential circuit with 50% duty circle. (Hint: Double the Clock)
6. Suppose you have a combinational circuit between two registers driven by a clock. What will you do if the delay of the combinational circuit is greater than your clock signal? (You can't resize the combinational circuit transistors)
7. The answer to the above question is breaking the combinational circuit and pipelining it. What will be affected if you do this?
8. What are the different Adder circuits you studied?
9. Give the truth table for a Half Adder. Give a gate level implementation of the same.
10. Draw a Transmission Gate-based D-Latch.
11. Design a Transmission Gate based XOR. Now, how do you convert it to XNOR? (Without inverting the output)
12. How do you detect if two 8-bit signals are same?
    XOR each bits of A with B (for e.g. A[0] xor B[0]) and so on.the o/p of 8 xor gates are then given as i/p to an 8-i/p nor gate. if o/p is 1 then A=B.
13. How do you detect a sequence of "1101" arriving serially from a signal line?
14. Design any FSM in VHDL or Verilog.
15. **Given only two xor gates one must function as buffer and another as inverter?**
    Tie one of xor gates input to 1 it will act as inverter.
    Tie one of xor gates input to 0 it will act as buffer.
16. **What is difference between latch and flipflop?**
    The main difference between latch and FF is that latches are level sensitive while FF are edge sensitive. They both require the use of clock signal and are used in sequential logic. For a latch,

the output tracks the input when the clock signal is high, so as long as the clock is logic 1, the output can change if the input also changes. FF on the other hand, will store the input only when there is a rising/falling edge of the clock.

17. **Implement an AND gate using mux?**
This is the basic question that many interviewers ask. For an AND gate, give one input as select line, in case if u r giving b as select line, connect one input to logic '0' and other input to a.

18. **What will happen if contents of register are shifter left, right?**
It is well known that in left shift all bits will be shifted left and LSB will be appended with 0 and in right shift all bits will be shifted right and MSB will be appended with 0 this is a straightforward answer
What is expected is in a left shift value gets Multiplied by 2 eg:consider 0000_1110=14 a left shift will make it 0001_110=28, it the same fashion right shift will Divide the value by 2.
19. **Design all the basic gates(NOT,AND,OR,NAND,NOR,XOR,XNOR) using 2:1 Multiplexer?**
Using 2:1 Mux, (2 inputs, 1 output and a select line)
(a) NOT
Give the input at the select line and connect I0 to 1 & I1 to 0. So if A is 1, we will get I1 that is 0 at the O/P.
(b) AND
Give input A at the select line and 0 to I0 and B to I1. O/p is A & B
(c) OR
Give input A at the select line and 1 to I1 and B to I0. O/p will be A | B
(d) NAND
AND + NOT implementations together
(e) NOR
OR + NOT implementations together
(f) XOR
A at the select line B at I0 and ~B at I1. ~B can be obtained from (a) (g) XNOR
A at the select line B at I1 and ~B at I0
20. **Design a circuit that calculates the square of a number?** It should not use any multiplier circuits. It should use Multiplexers and other logic?

This is interesting....
1^2=0+1=1
2^2=1+3=4
3^2=4+5=9
4^2=9+7=16
5^2=16+9=25
and so on
See a pattern yet?To get the next square, all you have to do is add the next odd number to the previous square that you found.See how 1,3,5,7 and finally 9 are added.Wouldn't this be a possible solution to your question since it only will use a counter,multiplexer and a couple of adders?It seems it would take n clock cycles to calculate square of n.
21. **Implement the following circuits:**
(a) 3 input NAND gate using min no of 2 input NAND Gates
(b) 3 input NOR gate using min no of 2 inpur NOR Gates
(c) 3 input XNOR gate using min no of 2 inpur XNOR Gates
Assuming 3 inputs A,B,C?

3 input NAND:
Connect :
a) A and B to the first NAND gate
b) Output of first Nand gate is given to the two inputs of the second NAND gate (this basically realizes the inverter functionality)
c) Output of second NAND gate is given to the input of the third NAND gate, whose other input is C
((A NAND B) NAND (A NAND B)) NAND C Thus, can be implemented using '3' 2-input NAND gates. I guess this is the minimum number of gates that need to be used.
3 input NOR:
Same as above just interchange NAND with NOR ((A NOR B) NOR (A NOR B)) NOR C
3 input XNOR:
Same as above except the inputs for the second XNOR gate, Output of the first XNOR gate is one of the inputs and connect the second input to ground or logical '0'
((A XNOR B) XNOR 0)) XNOR C

Computer Architecture:

1) What is pipelining?

2) What are the five stages in a DLX pipeline?

3) For a pipeline with 'n' stages, whatï¿½s the ideal throughput? What prevents us from achieving this ideal throughput?

4) What are the different hazards? How do you avoid them?

In computer architecture, a hazard is a potential problem that can happen in a pipelined processor. It refers to the possibility of erroneous computation when a CPU tries to simultaneously execute multiple instructions which exhibit data dependence. There are typically three types of hazards: data hazards, branching hazards (control hazards), and structural hazards.

Instructions in a pipelined processor are performed in several stages, so that at any given time several instructions are being executed, and instructions may not be completed in the desired order.

A hazard occurs when two or more of these simultaneous (possibly out of order) instructions conflict.

Types of hazard:

1. Data Hazard: RAW, WAR and WAR

    Forwarding – feed output data in an earlier stage of the pipeline.

2. Structural Hazard - i/o is needed by two or more applications simulatneosely.

3. Branch hazard - don't know what the branch result would be

Bubbling – can always issue NOPs.

Tomasulo – use a common data bus to communicate between the processors/ stages of pipeline.

5) Instead of just 5-8 pipe stages why not have, say, a pipeline with 50 pipe stages?

6) What are Branch Prediction and Branch Target Buffers?

In computer architecture, a branch predictor is the part of a processor that determines whether a conditional branch in the instruction flow of a program is likely to be taken or not. This is called branch prediction. Branch predictors are crucial in today's modern, superscalar processors for achieving high performance. They allow processors to fetch and execute instructions without waiting for a branch to be resolved.

Almost all pipelined processors do branch prediction of some form, because they must guess the address of the next instruction to fetch before the current instruction has been executed. Many earlier microprogrammed CPUs did not do branch prediction because there was little or no performance penalty for altering the flow of the instruction stream.

Branch prediction is not the same as branch target prediction. Branch prediction attempts to guess whether a conditional branch will be taken or not. Branch target prediction attempts to guess the target of the branch or unconditional jump before it is computed by parsing the instruction itself.

7) How do you handle precise exceptions or interrupts?

8) What is a cache?

9) What's the difference between Write-Through and Write-Back Caches? Explain advantages and disadvantages of each.

10) Cache Size is 64KB, Block size is 32B and the cache is Two-Way Set Associative. For a 32-bit physical address, give the division between Block Offset, Index and Tag.

11) What is Virtual Memory?

Virtual memory is a computer system technique which gives an application program the impression that it has contiguous working memory, while in fact it may be physically fragmented and may even overflow on to disk storage. Systems that use this technique make programming of large applications easier and use real physical memory (e.g. RAM) more efficiently than those without virtual memory.

Note that "virtual memory" is not just "using disk space to extend physical memory size". Extending memory is a normal consequence of using virtual memory techniques, but can be done by other means such as overlays or swapping programs and their data completely out to disk while they are inactive. The definition of "virtual memory" is based on tricking programs into thinking they are using large blocks of *contiguous* addresses.

All modern general-purpose computer operating systems use virtual memory techniques for ordinary applications, such as word processors, spreadsheets, multimedia players, accounting, etc. Embedded systems and other special-purpose computer systems which require very fast, very consistent response time do not generally use virtual memory.

12) What is Cache Coherency?

In computing, cache coherence refers to the integrity of data stored in local caches of a shared resource.

When clients in a system, particularly CPUs in a multiprocessing system, maintain caches of a common memory resource, problems arise. If the client A has a copy of a memory block from a previous read and client B changes that memory block, client A could be left with an invalid cache of memory without any notification of the change. Cache coherence is intended to manage such conflicts and maintain consistency between cache and memory.

13) What is MESI?

The **MESI protocol** (known also as *Illinois protocol*) is a widely used cache coherency and memory coherence protocol. It is the most common protocol which supports write-back cache. Its use in personal computers became widespread with the introduction of Intel's Pentium processor to "*support the more efficient write-back cache in addition to the write-through cache previously used by the Intel 486 processor*"[1].

Every cache line is marked with one of the four following states (coded in two additional bits):

Modifed

The cache line is present only in the current cache, and is *dirty*; it has been modified from the value in main memory. The cache is required to write the data back to main memory at some time in the future, before permitting any other read of the (no longer valid) main memory state.

Exclusive

The cache line is present only in the current cache, but is *clean*; it matches main memory.

Shared

Indicates that this cache may be stored in other caches of the machine.

Invalid

Indicates that this cache line is invalid.

14) What is a Snooping cache?

15) What are the components in a Microprocessor?

16) What is ACBF(Hex) divided by 16?

17) Convert 65(Hex) to Binary

18) Convert a number to its two's complement and back\

- 1's Complement: A negative number is represented by changing all zeroes to ones and all ones to zeroes. Example for an 8 bit representation of 1 and -1:

Binary	Signed	Unsigned
0000 0001	1	1
1111 1110	-1	254

Supported Range: -127 to 127. Disadvantage: We have two representations for 0: 0000 0000 and 1111 1111.

- 2's Complement: Similar to 1's complement but adding 1 for negative numbers:

Binary	Signed	Unsigned
0000 0001	1	1
1111 1111	-1	255
0111 1111	127	127
1000 0001	-127	129
1000 0000	-128	128

Supported Range: -128 to 127. Disadvantage: There is no corresponding positive number to the smallest negative integer value.

19) The CPU is busy but you want to stop and do some other task. How do you do it?

Context

```
Binary Division by Repeated Subtraction
#
Set quotient to zero
#
Repeat while dividend is greater than or equal to divisor

   * Subtract divisor from dividend
   * Add 1 to quotient
#
End of repeat block
#
quotient is correct, dividend is remainder
#
STOP
```

Binary Division by Shift and Subtract

Basically the reverse of the mutliply by shift and add.

```
#
Set quotient to 0
#
Align leftmost digits in dividend and divisor
#
Repeat

   * If that portion of the dividend above the divisor is greater than or equal to the divisor
         o Then subtract divisor from that portion of the dividend and
         o Concatentate 1 to the right hand end of the quotient
         o Else concatentate 0 to the right hand end of the quotient
   * Shift the divisor one place right

#
Until dividend is less than the divisor
#
quotient is correct, dividend is remainder
#
STOP
```

Binary Multiply - Repeated Shift and Add

Repeated shift and add - starting with a result of 0, shift the second multiplicand to correspond with each 1 in the first multiplicand and add to the result. Shifting each position left is equivalent to multiplying by 2, just as in decimal representation a shift left is equivalent to multiplying by 10.

```
#
Set result to 0
#
Repeat
   * Shift 2nd multiplicand left until rightmost digit is lined up with leftmost 1 in first multiplicand
   * Add 2nd multiplicand in that position to result
   * Remove that 1 from 1st multiplicand
#
Until 1st multiplicand is zero
#
Result is correct
#
STOP
```