1. মাইক্রোপ্রসেসর কী?
উত্তর: এমন একটি প্রোগ্রামেবল IC, যা ইনস্ট্রাকশন ফেচ, ডিকোড এবং এক্সিকিউট করে।

2. প্রথম মাইক্রোপ্রসেসর কোনটি?
উত্তর: Intel 4004, একটি ৪-বিট প্রসেসর যা ১৯৭১ সালে বের হয়।

3. মাইক্রোপ্রসেসরকে কম্পিউটারের CPU কেন বলা হয়?
উত্তর: কারণ এটি পুরো সিস্টেমের সব অপারেশন নিয়ন্ত্রণ ও প্রসেস করে।

4. ALU কীভাবে কাজ করে?
উত্তর: এটি সব অঙ্কগণিত (addition, subtraction ইত্যাদি) ও লজিক অপারেশন (AND, OR, XOR ইত্যাদি) সম্পন্ন করে।

5. কন্ট্রোল ইউনিটের কাজ কী?
উত্তর: এটি কন্ট্রোল সিগনাল তৈরি করে এবং সব অপারেশনকে সমন্বয় করে।

6. কোন কোন বাস CPU-কে মেমরির সাথে যুক্ত করে?
উত্তর: অ্যাড্রেস বাস, ডাটা বাস এবং কন্ট্রোল বাস।

7. প্রোগ্রাম কাউন্টার কেন প্রয়োজন?
উত্তর: এটি পরবর্তী ইনস্ট্রাকশনের অ্যাড্রেস ধরে রাখে।

8. স্ট্যাক কীভাবে সাহায্য করে?
উত্তর: স্ট্যাক সাময়িক ডাটা, যেমন রিটার্ন অ্যাড্রেস ইত্যাদি সংরক্ষণ করে।

9. 8085-এর ফ্ল্যাগগুলো কী কী?
উত্তর: Sign, Zero, Auxiliary Carry, Parity এবং Carry ফ্ল্যাগ।

10. ইনস্ট্রাকশন সেট কী?
উত্তর: এক সেট মেশিন-লেভেল ইনস্ট্রাকশন বা নির্দেশের সমষ্টি।

11. 8085-এ কোন কোন অ্যাড্রেসিং মোড আছে?
উত্তর: Immediate, Direct, Register, Register Indirect এবং Implied addressing mode।

12. অ্যাসেম্বলি ল্যাঙ্গুয়েজ কেন উপকারী?
উত্তর: কারণ এতে mnemonics ব্যবহার করে খুব নিচু-স্তরে হার্ডওয়্যার কন্ট্রোল করা যায়।

13. অ্যাসেম্বলার কীভাবে কাজ করে?
উত্তর: এটি mnemonics (assembly code) কে মেশিন কোডে ট্রান্সলেট করে।

14. অপকোড (opcode) কী?
উত্তর: ইনস্ট্রাকশনে কোন অপারেশনটি হবে তা নির্দেশকারী কোড।

15. 8086-এ কোন কোন রেজিস্টার আছে?
উত্তর: General-purpose, Segment, Pointer, Index এবং Flag রেজিস্টার।

16. 8086-এ segmentation কেন ব্যবহার করা হয়?
উত্তর: ২০-বিট অ্যাড্রেসিং ব্যবহার করে ১MB মেমরি অ্যাক্সেস করার জন্য।

17. 8085-এর ডাটা বাস কত বিটের?
উত্তর: এটি ৮-বিট ডাটা বাস।

18. পাইপলাইনিং (pipelining) কী?
উত্তর: ইনস্ট্রাকশনের বিভিন্ন পর্যায়কে ওভারল্যাপ করে একসাথে চালানো।

19. 8086-এ instruction queue কেন আছে?
উত্তর: ইনস্ট্রাকশন আগে থেকেই প্রি-ফেচ করে execution দ্রুত করার জন্য।

20. RISC আর CISC এর মধ্যে পার্থক্য কী?
উত্তর: RISC-এ কম এবং সহজ ইনস্ট্রাকশন, দ্রুত এক্সিকিউশন; CISC-এ জটিল, বেশি কাজ করা ইনস্ট্রাকশন থাকে।

21. রেজিস্টার অ্যাক্সেস আর মেমরি অ্যাক্সেস – কোনটা দ্রুত?
উত্তর: রেজিস্টার অ্যাক্সেস অনেক বেশি দ্রুত।

22. Instruction decoder কী করে?
উত্তর: অপকোডকে ইন্টারপ্রেট করে এবং কোন অপারেশন হবে তা নির্ধারণ করে সিগনাল তৈরি করে।

23. Interrupt কেন ব্যবহার করা হয়?
উত্তর: জরুরি ঘটনাগুলো polling না করে দ্রুত হ্যান্ডেল করার জন্য।

24. Maskable আর Non-maskable interrupt-এর পার্থক্য কী?
উত্তর: Maskable interrupt ডিসেবল করা যায়, Non-maskable interrupt ডিসেবল করা যায় না।

25. 8085-এ সর্বোচ্চ প্রায়োরিটি ইন্টারাপ্ট কোনটি?
উত্তর: TRAP, এটি একটি non-maskable interrupt।

26. ALU-তে ফ্ল্যাগ কেন দরকার?
উত্তর: অপারেশনের ফলাফল সম্পর্কে তথ্য দেয়, যেমন result zero হল কি না, carry হল কি না ইত্যাদি।

27. 8085-এ কয়টি অ্যাড্রেস লাইন আছে?
উত্তর: ১৬টি অ্যাড্রেস লাইন, যা ৬৪KB মেমরি অ্যাড্রেস করতে পারে।

28. Stack pointer কী?
উত্তর: একটি ১৬-বিট রেজিস্টার যা স্ট্যাকের টপ অ্যাড্রেস ধরে রাখে।

29. Cache মেমরি বলতে কী বোঝায়?
উত্তর: CPU আর RAM-এর মাঝে থাকা দ্রুতগতির SRAM মেমরি।

30. Clock signal কেন দরকার?
উত্তর: CPU-র সব অপারেশনকে same timing অনুযায়ী সিঙ্ক্রোনাইজ করার জন্য।

31. Machine cycle কী?
উত্তর: একবার ফেচ/রিড/রাইট অপারেশন সম্পন্ন করতে যে সময় লাগে।

32. SRAM আর DRAM – কোনটি দ্রুত?
উত্তর: SRAM বেশি দ্রুত, তবে বেশি দামী।

33. 8085-এ ALE সিগনাল কেন আছে?
উত্তর: Multiplexed address/data লাইন থেকে অ্যাড্রেস অংশ আলাদা করতে।

34. 8085-এ মোট কয়টি ইনস্ট্রাকশন আছে?
উত্তর: ৭৪ ধরনের ইনস্ট্রাকশন, ২৪৬টি অপকোড।

35. CALL আর JUMP-এর পার্থক্য কী?
উত্তর: CALL রিটার্ন অ্যাড্রেস স্ট্যাকে সেভ করে; JUMP শুধুই কন্ট্রোল স্থানান্তর করে, ফিরে আসার তথ্য রাখে না।

36. কোন ইনস্ট্রাকশন accumulator ক্লিয়ার করে?
উত্তর: XRA A ইনস্ট্রাকশন অ্যাকিউমুলেটরকে ০ করে।

37. Macro কেন ব্যবহার করা হয়?
উত্তর: একই ধরনের রিপিটেটিভ কোড একবার ডিফাইন করে অনেকবার ব্যবহার করার জন্য।

38. Subroutine কীভাবে সংজ্ঞায়িত হয়?
উত্তর: ছোট ছোট reusable কোড ব্লক, যেগুলো CALL ইনস্ট্রাকশনের মাধ্যমে ডাকা হয়।

39. কোন ইনস্ট্রাকশন Zero flag চেক করে?
উত্তর: JZ ইনস্ট্রাকশন ফলাফল শূন্য হলে জাম্প করে।

40. SIM ইনস্ট্রাকশন কেন ব্যবহার করা হয়?
উত্তর: Interrupt mask করা এবং Serial output সেট করার জন্য।

41. IN/OUT ইনস্ট্রাকশন কী কাজ করে?
উত্তর: CPU আর I/O পোর্টের মধ্যে ডাটা আদান-প্রদান করে।

42. Multiplexed bus কেন ব্যবহার করা হয়?
উত্তর: IC-এর পিন সংখ্যা কমানোর জন্য একই লাইনে অ্যাড্রেস ও ডাটা বহন করা।

43. DMA কী?
উত্তর: Direct Memory Access – ডিভাইস সরাসরি মেমরির সাথে ডাটা ট্রান্সফার করে, CPU-কে বাইপাস করে।

44. কোন সিগনাল read নির্দেশ করে?
উত্তর: $\overline{RD}$ সিগনাল মেমরি বা I/O read অপারেশন নির্দেশ করে।

45. T-state কী?
উত্তর: একটি clock period, অর্থাৎ ক্লকের এক পিরিয়ডের সময়।

46. 8086-এর segment রেজিস্টারগুলো কী কী?
উত্তর: CS, DS, SS এবং ES।

47. 8085-এ কোন register pair মেমরি পয়েন্ট করে?
উত্তর: HL register pair।

48. Logical ইনস্ট্রাকশন কেন দরকার?
উত্তর: AND, OR, XOR, shift, complement ইত্যাদি লজিকাল অপারেশনের জন্য।

49. Immediate আর Direct addressing-এর পার্থক্য কী?
উত্তর: Immediate-এ অপারান্ড ভ্যালু ইনস্ট্রাকশনেই থাকে; Direct-এ মেমরি অ্যাড্রেস থাকে।

50. কোন ইনস্ট্রাকশন CPU halt করে?
উত্তর: HLT ইনস্ট্রাকশন এক্সিকিউশন বন্ধ করে।

51. Segmentation কেন উপকারী?
উত্তর: মডুলার প্রোগ্রামিং সাপোর্ট করে এবং বড় মেমরি ব্যবস্থাপনায় সুবিধা দেয়।

52. Microprocessor আর Microcontroller-এর পার্থক্য কী?
উত্তর: মাইক্রোপ্রসেসর সাধারণত বাইরের RAM, ROM, I/O প্রয়োজন করে; মাইক্রোকন্ট্রোলারের ভিতরেই এগুলো বিল্ট-ইন থাকে।

53. Signed আর Unsigned সংখ্যার পার্থক্য কী?
উত্তর: Signed সংখ্যা পজিটিভ ও নেগেটিভ উভয়ই হতে পারে; Unsigned শুধু পজিটিভ।

54. Accumulator rotate করার ইনস্ট্রাকশন কোনগুলো?
উত্তর: RLC এবং RRC।

55. Assembler directive কেন ব্যবহার করা হয়?
উত্তর: এগুলো assembler-কে নির্দেশ দেয়, CPU এগুলো এক্সিকিউট করে না।

56. Loop কীভাবে ইমপ্লিমেন্ট করা হয় (8086-এ)?
উত্তর: LOOP ইনস্ট্রাকশন ও CX রেজিস্টার ব্যবহার করে।

57. Near আর Far jump-এর পার্থক্য কী?
উত্তর: Near jump শুধু IP পরিবর্তন করে; Far jump CS এবং IP দুটোই পরিবর্তন করে।

58. CALL ইনস্ট্রাকশন কেন শক্তিশালী?
উত্তর: Modular coding ও subroutine ব্যবহারে সহায়তা করে।

59. কোন ইনস্ট্রাকশন ডাটা swap করে?
উত্তর: XCHG রেজিস্টারের কনটেন্ট একে অপরের সাথে বদলায়।

60. Pseudo-instruction কেন ব্যবহার করা হয়?
উত্তর: Label, variable ইত্যাদি সংজ্ঞায়িত করতে – এগুলোও CPU এক্সিকিউট করে না, শুধু assembler-এর জন্য।

61. PUSH কীভাবে কাজ করে?
উত্তর: আগে SP decrement হয়, তারপর সেই নতুন SP-তে ডাটা লেখা হয়।

62. কোন ইনস্ট্রাকশন stack থেকে ডাটা রিস্টোর করে?
উত্তর: POP ইনস্ট্রাকশন রেজিস্টারে ডাটা ফিরিয়ে আনে।

63. 8086-এ segment override কেন ব্যবহার করা হয়?
উত্তর: ডিফল্ট segment ছাড়া অন্য segment থেকে ডাটা/কোড অ্যাক্সেস করার জন্য।

64. Software আর Hardware interrupt-এর পার্থক্য কী?
উত্তর: Software interrupt ইনস্ট্রাকশন (যেমন INT n) দ্বারা; Hardware interrupt বাইরের সিগনাল দ্বারা ঘটে।

65. কোন ইনস্ট্রাকশন software interrupt জেনারেট করে?
উত্তর: `INT n` ইনস্ট্রাকশন।

66. Interrupt-এ priority encoder কেন লাগে?
উত্তর: একাধিক interrupt একসাথে এলে কোনটা আগে সার্ভিস পাবে তা নির্ধারণ করতে।

67. 8086-এ instruction queue-এর সাইজ কত?
উত্তর: ৬ বাইট।

68. কোন সিগনাল memory আর I/O আলাদা করে?
উত্তর: IO/M̄ সিগনাল মেমরি নাকি I/O অ্যাক্সেস হচ্ছে তা নির্দেশ করে।

69. Parity flag কেন আছে?
উত্তর: ফলাফলের মধ্যে ১-বিটের সংখ্যা জোড় নাকি বিজোড় তা চেক করতে।

70. কোন ইনস্ট্রাকশন accumulator complement করে?
উত্তর: CMA – সব বিট উল্টে দেয় (0→1, 1→0)।

71. Shift instruction কেন ব্যবহার করা হয়?
উত্তর: খুব দ্রুত ২ দ্বারা গুণ বা ভাগসহ অন্যান্য bit-wise অপারেশন করার জন্য।

72. SHR আর SAR-এর পার্থক্য কী?
উত্তর: SHR Logical shift right (sign বিট ধরে না); SAR Arithmetic shift right (sign বিট ধরে রাখে)।

73. Interrupt polling-এর চেয়ে দ্রুত কেন?
উত্তর: কারণ CPU-কে বারবার ডিভাইস চেক করতে হয় না; ইভেন্ট হলে নিজে থেকে CPU-কে জানায়।

74. 8086-এ base pointer কোন রেজিস্টার?
উত্তর: BP রেজিস্টার।

75. 8086-এ segment overlap কেন করা হয়?
উত্তর: মেমরি ইফিশিয়েন্টভাবে ব্যবহার করতে এবং বিভিন্ন সেগমেন্টের ওভারল্যাপিং নিশ্চিত করতে।

76. Trap flag কী?
উত্তর: একে সেট করলে প্রসেসর single-step mode-এ চলে যায়, অর্থাৎ একেকটি ইনস্ট্রাকশনের পর ইন্টারাপ্ট হয় – ডিবাগিংয়ের জন্য।

77. Conditional jump কেন দরকার?
উত্তর: ফ্ল্যাগের অবস্থা অনুযায়ী প্রোগ্রামের নিয়ন্ত্রণ এক জায়গা থেকে আরেক জায়গায় সরাতে।

78. কোন ইনস্ট্রাকশন signed সংখ্যার গুণ করে?
উত্তর: IMUL।

79. 8086-এ minimum এবং maximum mode কেন আছে?
উত্তর: Minimum mode একক প্রসেসর সিস্টেমের জন্য, Maximum mode মাল্টিপ্রসেসর বা কোপ্রসেসর সিস্টেমের জন্য।

80. কোন ইনস্ট্রাকশন unsigned division করে?
উত্তর: DIV।

81. Assembler এর pass-1 কী করে?
উত্তর: Symbol table তৈরি করে, লেবেল রেজলভ করে এবং সিনট্যাক্স চেক করে।

82. Pass-2 কী করে?
উত্তর: চূড়ান্ত মেশিন কোড জেনারেট করে।

83. Assembly-তে label কেন ব্যবহার করা হয়?
উত্তর: নির্দিষ্ট ইনস্ট্রাকশনের মেমরি অ্যাড্রেস সহজে রেফার করার জন্য।

84. কোন রেজিস্টার instruction pointer ধরে রাখে?
উত্তর: IP রেজিস্টার (8086-এ)।

85. MOV আর MVI-এর পার্থক্য কী (8085-এ)?
উত্তর: MOV রেজিস্টার/মেমরি থেকে রেজিস্টারে কপি করে; MVI রেজিস্টার বা মেমরিতে immediate ডাটা লোড করে।

86. Segment:offset addressing কেন ব্যবহার করা হয় (8086-এ)?
উত্তর: Segment ও offset মিলিয়ে ২০-বিট ফিজিক্যাল অ্যাড্রেস তৈরি করার জন্য।

87. 8086-এর predefined interrupt কোনগুলো?
উত্তর: INT 0–4 (divide error, debug, NMI ইত্যাদির জন্য)।

88. কোন ফ্ল্যাগ overflow দেখায়?
উত্তর: Overflow flag (OF)।

89. Conditional CALL কেন দরকার?
উত্তর: নির্দিষ্ট ফ্ল্যাগ সেট/ক্লিয়ার থাকলে তবেই সাবরুটিন কল করার জন্য।

90. Assembler আর Linker-এর পার্থক্য কী?
উত্তর: Assembler source code থেকে object code বানায়; Linker একাধিক object module ও লাইব্রেরি একত্র করে executable বানায়।

91. Carry flag ক্লিয়ার করার ইনস্ট্রাকশন কোনটি?
উত্তর: CLC।

92. Stack নিচের দিকে (downward) কেন গ্রো করে?
উত্তর: যেন কোড আর ডাটা সেগমেন্টের সাথে সংঘর্ষ কম হয় এবং মেমরি ব্যবস্থাপনা সহজ হয়।

93. ORG আর EQU-এর পার্থক্য কী?
উত্তর: ORG origin (location counter) সেট করে; EQU কোনো কনস্ট্যান্ট ভ্যালুকে নাম দেয়।

94. No-operation ইনস্ট্রাকশন কোনটি?
উত্তর: NOP।

95. LOOP ইনস্ট্রাকশন কেন উপকারী?
উত্তর: বারবার ডিক্রিমেন্ট ও জাম্প লেখার ঝামেলা কমায়, কোড ছোট ও পরিষ্কার হয়।

96. 8086-এ CS রেজিস্টারের ভূমিকা কী?
উত্তর: Code segment-এর বেস অ্যাড্রেস ধরে রাখে।

97. কোন ইনস্ট্রাকশন effective address লোড করে?
উত্তর: LEA।

98. TEST ইনস্ট্রাকশন কেন ব্যবহার করা হয়?
উত্তর: দুটি অপারান্ডের AND করে ফ্ল্যাগ সেট/ক্লিয়ার করতে, কিন্তু অপারান্ড দুইটির মান পরিবর্তন না করে।

99. XLAT এর ব্যবহার কী?
উত্তর: একটি lookup table থেকে ডাটাকে translate করতে (AL-এর মানকে টেবিল থেকে নতুন মান দিয়ে বদলায়)।

100. কোন ইনস্ট্রাকশন string কপি করে?
উত্তর: MOVS (MOVSB/MOVSW) – byte বা word কপি করে।

101. Direction flag কেন গুরুত্বপূর্ণ?
উত্তর: String অপারেশনগুলো বামে (address decrement) না ডানে (address increment) এগোবে তা নির্ধারণ করে।

102. কোন ইনস্ট্রাকশন string স্ক্যান করে?
উত্তর: SCAS – নির্দিষ্ট byte/word খোঁজার জন্য string স্ক্যান করে।

103. Repeat prefix কেন ব্যবহার করা হয়?
উত্তর: String ইনস্ট্রাকশন (MOVS, SCAS, CMPS ইত্যাদি) নির্দিষ্ট সংখ্যক বার বা নির্দিষ্ট কন্ডিশন পর্যন্ত রিপিট করাতে।

104. REP আর REPE-এর পার্থক্য কী?
উত্তর: REP কেবল count (CX) শেষ হওয়া পর্যন্ত রিপিট করে; REPE/REPZ একই সাথে Zero flag সেট থাকা পর্যন্ত রিপিট করে।

105. কোন ইনস্ট্রাকশন string compare করে?
উত্তর: CMPS – দুটি string-এর byte/word তুলনা করে।

106. Conditional assembly কেন ব্যবহার করা হয়?
উত্তর: নির্দিষ্ট কন্ডিশনের উপর ভিত্তি করে কিছু অংশ assemble করা বা বাদ দেওয়ার জন্য।

107. Macro আর Procedure-এর পার্থক্য কী?
উত্তর: Macro inline এক্সপ্যান্ড হয় (কোড কপি হয়); Procedure CALL/RET দিয়ে কল হয়, এক জায়গায় থাকে।

108. Modular programming কেন গুরুত্বপূর্ণ?
উত্তর: কোডকে ভাগে ভাগ করে লেখা যায়, এতে readability, debugging ও reuse সহজ হয়।

109. Interrupt থেকে ফেরার ইনস্ট্রাকশন কোনটি?
উত্তর: IRET।

110. CLI আর STI-এর পার্থক্য কী?
উত্তর: CLI interrupt disable করে; STI interrupt enable করে।

111. WAIT state কেন ব্যবহার করা হয়?
উত্তর: Slow memory বা ডিভাইসের সাথে CPU-এর স্পিড মিলিয়ে কাজ করাতে।

112. 8086-এ READY সিগনালের ভূমিকা কী?
উত্তর: মেমরি/ডিভাইস ডাটা রেডি আছে কি না তা CPU-কে জানায়, প্রয়োজন হলে WAIT state তৈরি হয়।

113. 8085-এ পাওয়ার সাপ্লাই কোন পিনে?
উত্তর: Vcc = +5V, Vss = GND।

114. Address/data line multiplex কেন করা হয়?
উত্তর: IC-তে পিন সংখ্যা কমানোর জন্য একই পিনে কখনো অ্যাড্রেস, কখনো ডাটা পাঠানো হয়।

115. 8085-এ SID/SOD-এর কাজ কী?
উত্তর: SID (Serial Input Data) – সিরিয়াল ইনপুট, SOD (Serial Output Data) – সিরিয়াল আউটপুট লাইন।

116. 8086-এ কোন রেজিস্টার মেমরি পয়েন্টার হিসেবে কাজ করে?
উত্তর: SI (Source Index) এবং DI (Destination Index)।

117. SS রেজিস্টার কেন দরকার?
উত্তর: Stack segment-এর বেস অ্যাড্রেস ধরে রাখার জন্য।

118. JMP আর LCALL-এর পার্থক্য কী?
উত্তর: JMP শুধু কন্ট্রোল ট্রান্সফার করে; LCALL (long call) কন্ট্রোল ট্রান্সফার করার পাশাপাশি রিটার্ন অ্যাড্রেস সেভ করে।

119. PUSHF আর POPF কেন ব্যবহার করা হয়?
উত্তর: Flags register স্ট্যাকে সেভ ও পরে রিস্টোর করার জন্য।

120. কোন ইনস্ট্রাকশন carry সহ shift করে?
উত্তর: RCL এবং RCR।

121. NOP-এর ভূমিকা কী?
উত্তর: কোনো কাজ না করে শুধু একটি ক্লক/সাইকেল খরচ করে – timing adjust বা delay করার জন্য।

122. Assembler listing file কেন তৈরি করে?
উত্তর: Debugging ও reference-এর জন্য – সোর্স লাইনের সাথে মেশিন কোড দেখায়।

123. END directive-এর ব্যবহার কী?
উত্তর: সোর্স প্রোগ্রামের শেষ নির্দেশ করতে।

124. প্রসেসরে microcode কী?
উত্তর: কন্ট্রোল মেমরিতে রাখা খুব নিচু-স্তরের control instruction, যা হার্ডওয়্যার সিগনাল জেনারেট করে।

125. মাইক্রোপ্রসেসরে কোন কোন pipeline hazard হয়?
উত্তর: Structural, Data এবং Control hazards।

126. Cache coherence কেন গুরুত্বপূর্ণ?
উত্তর: একাধিক cache-এ কোনো ডাটার কপি থাকলে সেগুলো যেন সঙ্গতিপূর্ণ (consistent) থাকে তা নিশ্চিত করতে।

127. Virtual memory CPU-কে কীভাবে সাহায্য করে?
উত্তর: ডিস্কের অংশকে মেমরি হিসেবে ব্যবহার করে প্রোগ্রামকে বড় address space দেয়।

128. Paging কী?
উত্তর: মেমরিকে সমান fixed-size ব্লকে (page) ভাগ করার পদ্ধতি।

129. Segmentation fault কেন হয়?
উত্তর: যখন প্রোগ্রাম অবৈধ বা অনুমতিহীন মেমরি অ্যাড্রেসে এক্সেস করতে চায়।

130. MMU কীভাবে কাজ করে?
উত্তর: Virtual address-কে physical address-এ ট্রান্সলেট করে।

131. কোন টেকনিক instruction throughput বাড়ায়?
উত্তর: Pipelining।

132. Branch prediction কেন গুরুত্বপূর্ণ?
উত্তর: ভুল branch নিলে pipeline stall হয়; prediction ঠিক হলে stall কমে।

133. Microprogrammed control কী?
উত্তর: Control signal গুলো microcode থেকে জেনারেট হয় – হার্ডওয়্যার logic-এর বদলে ছোট প্রোগ্রাম দ্বারা নিয়ন্ত্রিত।

134. কোন addressing mode সবচেয়ে দ্রুত?
উত্তর: Register addressing।

135. Harvard architecture কেন ব্যবহার করা হয়?
উত্তর: Instruction ও data-র জন্য আলাদা bus ব্যবহার করে ফেচ ও ডাটা অ্যাক্সেস parallel করা যায়।

136. Superscalar architecture কীভাবে performance বাড়ায়?
উত্তর: প্রতি ক্লক সাইকেলে একাধিক ইনস্ট্রাকশন parallelভাবে execute করে।

137. Macro instruction আর Micro instruction-এর পার্থক্য কী?
উত্তর: Macro উচ্চ-স্তরের assembly লেভেলের; Micro instruction প্রসেসরের control unit-এর হার্ডওয়্যার লেভেলের নির্দেশ।

138. Stack overflow কেন হয়?
উত্তর: Stack-এর জন্য নির্ধারিত মেমরি সীমা অতিক্রম করলে।

139. Interrupt vector table কীভাবে কাজ করে?
উত্তর: প্রতিটি interrupt-এর ISR-এর অ্যাড্রেস এখানে সংরক্ষিত থাকে; interrupt এলে টেবিল থেকে ঠিকানা নিয়ে ISR চালানো হয়।

140. Pipeline-এ কোন ধরনের scheduling ব্যবহার করা হয়?
উত্তর: Dynamic instruction scheduling।

141. Out-of-order execution কেন উপকারী?
উত্তর: ডিপেন্ডেন্সি না থাকা ইনস্ট্রাকশন আগে execute করে stall কমায়।

142. Register renaming কী?
উত্তর: ফালতু (false) ডিপেন্ডেন্সি দূর করতে লজিক্যাল রেজিস্টারের জন্য আলাদা ফিজিক্যাল রেজিস্টার ব্যবহার করা।

143. CISC এখনো কেন ব্যবহার হয়?
উত্তর: ভালো code density, পুরনো সফটওয়্যার ও আর্কিটেকচারের সাথে সামঞ্জস্যতা (legacy support) জন্য।

144. Instruction latency আর throughput-এর পার্থক্য কী?
উত্তর: Latency = এক ইনস্ট্রাকশন শেষ হতে যত সময় লাগে; Throughput = প্রতি ইউনিট সময়ে কত ইনস্ট্রাকশন শেষ হচ্ছে।

145. Speculative execution কী?
উত্তর: ব্রাঞ্চের ফল নিশ্চিত হওয়ার আগেই অনুমিত পথ ধরে ইনস্ট্রাকশন execute করা।

146. 8086-এ flags কোন রেজিস্টারে থাকে?
উত্তর: FLAGS register।

147. Instruction pipelining সীমাবদ্ধ কেন?
উত্তর: Hazards ও ডিপেন্ডেন্সির কারণে সব স্টেজ সবসময় ব্যস্ত রাখা যায় না।

148. Control signal আর Status signal-এর পার্থক্য কী?
উত্তর: Control signal অপারেশন নির্দেশ করে; Status signal বর্তমান অবস্থা নির্দেশ করে।

149. DMA CPU-র চেয়ে দ্রুত কেন?
উত্তর: কারণ ডিভাইস সরাসরি মেমরিতে ডাটা ট্রান্সফার করে, CPU-কে বারবার জড়াতে হয় না।

150. Vectored interrupt আর Non-vectored interrupt-এর পার্থক্য কী?
উত্তর: Vectored interrupt-এর ISR অ্যাড্রেস নির্দিষ্ট (predefined); Non-vectored-এ বাহ্যিকভাবে ISR অ্যাড্রেস প্রদান করতে হয়।

151. Shadow register কী?
উত্তর: দ্রুত context switch-এর জন্য মূল রেজিস্টারের ব্যাকআপ কপি।

152. Prefetch buffer কেন ব্যবহার করা হয়?
উত্তর: Instruction আগেই ফেচ করে রেখে execution বিলম্ব কমাতে।

153. Pipeline flushing কী?
উত্তর: ভুল ব্রাঞ্চ ধরা পড়লে বা ব্যতিক্রম ঘটলে pipeline-এর সব আংশিক execute ইনস্ট্রাকশন ফেলে দেওয়া।

154. Arithmetic overflow কোন ফ্ল্যাগ দিয়ে ধরা হয়?
উত্তর: Overflow flag (OF)।

155. CISC-এ instruction length ভিন্ন হয় কেন?
উত্তর: বেশি flexibility ও জটিল অপারেশন সাপোর্ট করার জন্য ভ্যারিয়েবল লেন্থ ইনস্ট্রাকশন।

156. Stack frame কী?
উত্তর: একেকটা ফাংশন/প্রসিডিউরের জন্য stack-এ রাখা লোকাল ভ্যারিয়েবল, রিটার্ন অ্যাড্রেস, সেভড রেজিস্টার ইত্যাদির ব্লক।

157. Call stack কেন গুরুত্বপূর্ণ?
উত্তর: Nested ও recursive ফাংশন কলের অবস্থা ট্র্যাক করে।

158. Priority interrupt কীভাবে কাজ করে?
উত্তর: প্রতিটি interrupt-কে নির্দিষ্ট প্রায়োরিটি দেওয়া থাকে, উচ্চ প্রায়োরিটি আগে সার্ভিস পায়।

159. কোন unit floating-point অপারেশন করে?
উত্তর: FPU (Floating-Point Unit)।

160. মাইক্রোপ্রসেসরে clock অভ্যন্তরীণভাবে দ্বিগুণ কেন করা হয় কখনো?
উত্তর: Instruction-এর বিভিন্ন internal phase দ্রুত সম্পন্ন করার জন্য।

161. Pipeline stall কীভাবে সমাধান করা হয়?
উত্তর: Forwarding (data path আগেই সরবরাহ) বা bubble (ফাঁকা সাইকেল ঢুকিয়ে) ব্যবহার করে।

162. Hyper-threading কী?
উত্তর: Simultaneous multi-threading – একই কোরে একাধিক থ্রেড parallelভাবে চালানো।

163. Instruction alignment কেন গুরুত্বপূর্ণ?
উত্তর: Misaligned ডাটা/ইনস্ট্রাকশন অ্যাক্সেস করলে বাড়তি সাইকেল লাগে।

164. Watchdog timer কী?
উত্তর: সিস্টেম হ্যাং করলে বা নির্দিষ্ট সময় ইনস্ট্রাকশন না চললে রিসেট করার জন্য ব্যবহার্য টাইমার।

165. Memory-mapped I/O কেন ব্যবহার করা হয়?
উত্তর: I/O ডিভাইসকে মেমরি অ্যাড্রেস স্পেসে ম্যাপ করে একই load/store ইনস্ট্রাকশন দিয়ে অ্যাক্সেস করা যায়।

166. Pipeline-এ Instruction Fetch Unit কী করে?
উত্তর: Program counter অনুযায়ী ইনস্ট্রাকশন ফেচ করে।

167. Cache associativity কেন ব্যবহার করা হয়?
উত্তর: Cache conflict miss কমানোর জন্য।

168. Write-through আর Write-back cache-এর পার্থক্য কী?
উত্তর: Write-through-এ cache আপডেট হলে সঙ্গে সঙ্গে main memory-ও আপডেট হয়; Write-back-এ পরে (evict-এর সময়) আপডেট হয়।

169. Hazard detection unit কেন প্রয়োজন?
উত্তর: Pipeline-এ data/control hazard সনাক্ত করে সঠিকভাবে stall বা forwarding করতে।

170. Clock skew CPU-কে কীভাবে প্রভাবিত করে?
উত্তর: বিভিন্ন অংশে clock ভিন্ন সময়ে পৌঁছালে synchronization সমস্যা ও ভুল অপারেশন হতে পারে।

171. Speculative branch execution কী?
উত্তর: Branch-এর সম্ভাব্য পথ ধরে আগেই ইনস্ট্রাকশন execute করা।

172. Assembly-তে loop unrolling কেন করা হয়?
উত্তর: branch/jump-এর overhead কমাতে ও বেশি parallelism পেতে একই লুপের body কয়েকবার কপি করা।

173. Instruction window কী?
উত্তর: Decode হওয়া কিন্তু এখনও execute/commit না হওয়া ইনস্ট্রাকশনের buffer।

174. Instruction set orthogonal হওয়া কেন ভালো?
উত্তর: একই addressing mode ও অপারেশন নিয়মিতভাবে সব ইনস্ট্রাকশনে প্রয়োগ করা যায় – ডিজাইন সহজ ও consistent হয়।

175. Micro-operation কী?
উত্তর: একটি instruction execute করতে প্রয়োজনীয় ক্ষুদ্র হার্ডওয়্যার স্তরের অপারেশন, যেমন register transfer।

176. Paging-এর সাথে TLB কেন গুরুত্বপূর্ণ?
উত্তর: Recent address translation cache করে virtual→physical mapping দ্রুত করে।

177. Bus arbitration কী?
উত্তর: যখন একাধিক ডিভাইস বাস ব্যবহার করতে চায়, তখন কাকে বাস দেওয়া হবে তা নির্ধারণের প্রক্রিয়া।

178. Priority inversion সমস্যা কেন হয়?
উত্তর: নিচু প্রায়োরিটির টাস্ক কোনো রিসোর্স ধরে আছে, আর উচ্চ প্রায়োরিটি টাস্ক সেই রিসোর্সের জন্য অপেক্ষা করছে – মাঝখানে মাঝারি প্রায়োরিটির টাস্ক CPU দখল করে রাখলে inversion ঘটে।

179. Cache miss penalty কীভাবে কমানো হয়?
উত্তর: Multi-level cache (L1, L2, L3) ব্যবহার করে।

180. Maskable interrupt আর Vectored interrupt-এর মধ্যে পার্থক্য কী?
উত্তর: Maskable interrupt disable করা যায়; Vectored interrupt-এর ISR address predefined – এখানে দুই ধারণা আলাদা প্রপার্টি, অনেক interrupt-ই দুটো বৈশিষ্ট্য একসাথে রাখতে পারে।

181. Instruction pre-decoding কেন করা হয়?
উত্তর: সামনে pipeline স্টেজগুলো দ্রুত চালাতে ইনস্ট্রাকশন আগে থেকেই আংশিক ডিকোড করে রাখা।

182. Barrel shifter কী?
উত্তর: এক ক্লকেই একাধিক বিট স্থানান্তর (shift/rotate) করতে সক্ষম সার্কিট।

183. Instruction cycle আর Machine cycle-এর পার্থক্য কী?
উত্তর: Instruction cycle = পূর্ণ ইনস্ট্রাকশন ফেচ, ডিকোড, এক্সিকিউট ইত্যাদির সমষ্টি; Machine cycle = একেকটা ফেচ/রিড/রাইট ধাপ, একাধিক machine cycle মিলে এক instruction cycle।

184. Harvard আর Von Neumann আর্কিটেকচারের পার্থক্য কী?
উত্তর: Harvard-এ instruction ও data-এর জন্য আলাদা মেমরি/বাস; Von Neumann-এ একটাই শেয়ার্ড মেমরি/বাস।

185. Stack-based machine সহজ কেন?
উত্তর: Explicit register addressing লাগে না, সব অপারেশন stack-এর top এলিমেন্টের উপর হয়।

186. Pipeline depth কী?
উত্তর: Pipeline-এর মোট স্টেজের সংখ্যা।

187. Hazard forwarding কেন ব্যবহার করা হয়?
উত্তর: Data hazard দ্রুত সমাধান করে পরের ইনস্ট্রাকশনকে অপেক্ষা না করিয়ে ডাটা সরাসরি আগের স্টেজ থেকে পৌঁছে দিতে।

188. Branch delay slot কী?
উত্তর: Branch ইনস্ট্রাকশনের পরের একটি ইনস্ট্রাকশন, যা branch নেওয়া হোক বা না হোক execute হয়।

189. Dynamic branch prediction ভালো কেন?
উত্তর: প্রোগ্রামের বাস্তব আচরণ থেকে শেখে এবং বেশি accurate হয়।

190. Bus cycle কী?
উত্তর: বাসে একবার ডাটা ট্রান্সফারের পুরো প্রক্রিয়া।

191. Instruction-level parallelism (ILP) কেন গুরুত্বপূর্ণ?
উত্তর: একাধিক ইনস্ট্রাকশন একই সময়ে execute করে throughput বাড়াতে।

192. Superscalar execution কী?
উত্তর: একাধিক pipeline ব্যবহার করে এক ক্লকে একের বেশি ইনস্ট্রাকশন parallelভাবে execute করা।

193. 8255-এ control word কেন ব্যবহার হয়?
উত্তর: পোর্টগুলোর mode ও দিক (input/output) কনফিগার করতে।

194. Tri-state buffer কী?
উত্তর: তিনটি অবস্থা থাকে – Logic 0, Logic 1 এবং High-impedance (Z)।

195. Instruction profiling কেন করা হয়?
উত্তর: প্রোগ্রামের কোন অংশ বেশি সময় নিচ্ছে তা জেনে সেগুলো অপ্টিমাইজ করতে।

196. Hardwired control আর Microprogrammed control-এর পার্থক্য কী?
উত্তর: Hardwired control দ্রুত কিন্তু পরিবর্তন কঠিন; Microprogrammed control তুলনামূলক ধীর কিন্তু সহজে পরিবর্তনযোগ্য।

197. মেমরি segmentation কেন উপকারী?
উত্তর: বড় প্রোগ্রামকে আলাদা আলাদা লজিক্যাল অংশে ভাগ করে ব্যবস্থাপনা সহজ করে।

198. Paging overhead কী?
উত্তর: Address translation ও পেজ টেবিল ব্যবহারে বাড়তি সময় ও রিসোর্স খরচ।

199. Wait state কেন যোগ করা হয়?
উত্তর: ধীর peripheral বা মেমরিকে CPU-এর স্পিডের সাথে মিলিয়ে কাজ করানোর জন্য।

200. Bus contention কী?
উত্তর: যখন একাধিক ডিভাইস একই বাস লাইন একসাথে drive করতে চায় এবং সংঘর্ষ হয়।

201. PUSH করার সময় Stack pointer auto-decrement কেন হয়?
উত্তর: Stack সাধারণত উপরের অ্যাড্রেস থেকে নিচের দিকে বাড়ে, তাই নতুন ডাটা রাখার আগে SP কমানো হয়।

202. Recursive call মাইক্রোপ্রসেসর কীভাবে হ্যান্ডেল করে?
উত্তর: প্রতিটি CALL-এর রিটার্ন অ্যাড্রেস ও লোকাল ডাটা stack-এ রেখে।

203. CPU pipeline-এর front-end আর back-end কী?
উত্তর: Front-end ইনস্ট্রাকশন ফেচ ও ডিকোড করে; Back-end execute ও write-back করে।

204. CPU কেন instruction reordering করে?
উত্তর: Stall কমিয়ে execution efficiency বাড়ানোর জন্য।

205. 8086-এ কোন ইনস্ট্রাকশনগুলো privileged?
উত্তর: যেমন CLI, STI ইত্যাদি – সিস্টেম রিসোর্স কন্ট্রোল করে, সাধারণত OS লেভেলে ব্যবহার হয়।

206. Instruction cache main memory-এর চেয়ে ছোট কেন?
উত্তর: খুব দ্রুত ও দামী হওয়ায় ছোট আকারে রাখা হয়, তবে access latency কম।

207. Context switch কী?
উত্তর: এক টাস্কের register, PC, state সংরক্ষণ করে অন্য টাস্কের state লোড করা।

208. Memory interleaving স্পিড কীভাবে বাড়ায়?
উত্তর: একাধিক মেমরি ব্যাংকে parallelভাবে অ্যাক্সেস করে।

209. কোন instruction set-এ SIMD অপারেশন থাকে?
উত্তর: আধুনিক Intel প্রসেসরে MMX, SSE, AVX ইত্যাদি।

210. Pipeline-এ hazard কেন তৈরি হয়?
উত্তর: ইনস্ট্রাকশনগুলোর মধ্যকার data, control বা resource dependency থাকার কারণে।

211. Branch prediction কীভাবে implement করা হয়?
উত্তর: Static (fixed rule) বা Dynamic (branch history ও pattern-এর উপর ভিত্তি করে) অ্যালগরিদম দিয়ে।

212. CPU-তে reorder buffer কী?
উত্তর: Out-of-order execute হওয়া ইনস্ট্রাকশনগুলোকে proper order-এ commit করার আগে ধরে রাখার buffer।

213. Speculative load কেন ব্যবহার হয়?
উত্তর: ডাটা আগে থেকেই ফেচ করে রাখলে পরের স্টেজে delay কমে।

214. 8086-এ signed division কোন ইনস্ট্রাকশন করে?
উত্তর: IDIV।

215. CPU-তে multiple clock domain কেন থাকে?
উত্তর: আলাদা অংশকে তাদের উপযুক্ত ফ্রিকোয়েন্সিতে চালাতে, power ও performance ব্যালান্স করার জন্য।

216. Memory-mapped I/O-এর সুবিধা কী?
উত্তর: একই instruction set দিয়ে মেমরি আর I/O ডিভাইস উভয়ই অ্যাক্সেস করা যায়।

217. Branch target buffer (BTB) কীভাবে কাজ করে?
উত্তর: সম্প্রতি নেওয়া branch-এর target address cache করে রাখে, যাতে branch ফেচ দ্রুত হয়।

218. Instruction decode স্টেজ এত critical কেন?
উত্তর: ভুল ডিকোড হলে পুরো execution ভুল হয়; আর এখান থেকেই control signal নির্ধারণ হয়।

219. Dual-ported memory কী?
উত্তর: দুইটি আলাদা পোর্ট দিয়ে একই সময়ে parallel read/write করা যায় এমন মেমরি।

220. Instruction prefetch performance কীভাবে বাড়ায়?
উত্তর: প্রয়োজনের আগেই instruction cache/buffer-এ এনে fetch delay কমায়।

221. Write buffer কী?
উত্তর: CPU থেকে আসা write অপারেশন সাময়িকভাবে ধরে রাখে, পরে main memory-তে লিখে।

222. Instruction scheduling কেন গুরুত্বপূর্ণ?
উত্তর: Stall কমিয়ে pipeline সর্বোচ্চ ব্যস্ত রাখতে।

223. Bit manipulation সাপোর্ট করে কোন instruction set?
উত্তর: 8085/8086-এ ROL, ROR, SHL, SHR, BT, BTS, BTR ইত্যাদি।

224. Hardware loop counter কীভাবে কাজ করে?
উত্তর: একটি রেজিস্টার স্বয়ংক্রিয়ভাবে ডিক্রিমেন্ট হয়ে zero হলে লুপ শেষ হয়, এর মাধ্যমে লুপ কন্ট্রোল হয়।

225. CPU speculative execution কেন ব্যবহার করে?
উত্তর: pipeline idle না রেখে ভবিষ্যতের ইনস্ট্রাকশন আগে থেকে চালিয়ে throughput বাড়াতে।

226. Memory alignment কী?
উত্তর: ডাটা তার সাইজ দ্বারা বিভাজ্য address-এ সংরক্ষণ করা, যেন অ্যাক্সেস কার্যকর হয়।

227. Cache line size গুরুত্বপূর্ণ কেন?
উত্তর: একবার fetch-এ কতটা ধারাবাহিক মেমরি আনা হবে তা নির্ধারণ করে, hit rate প্রভাবিত হয়।

228. কোন ইনস্ট্রাকশন flags stack-এ সেভ করে?
উত্তর: 8086-এ PUSHF।

229. Instruction register কেন গুরুত্বপূর্ণ?
উত্তর: বর্তমানে execute হওয়া instruction-টি ধরে রাখে।

230. Dynamic Voltage and Frequency Scaling (DVFS) কী?
উত্তর: লোড অনুযায়ী CPU-এর ভোল্টেজ ও ফ্রিকোয়েন্সি পরিবর্তন করে power efficiency বাড়ানো।

231. TLB virtual memory access কীভাবে দ্রুত করে?
উত্তর: সাম্প্রতিক virtual→physical address mapping cache করে রাখে।

232. Branch misprediction এত costly কেন?
উত্তর: ভুল পথের সব ইনস্ট্রাকশন flush করে pipeline আবার ভরতে হয়, অনেক সাইকেল নষ্ট হয়।

233. কোন ইনস্ট্রাকশন string operation রিপিট করে?
উত্তর: REP, REPE/REPZ, REPNE/REPNZ।

234. Priority encoder কী?
উত্তর: একাধিক ইনপুটের মধ্যে highest-priority active ইনপুট সনাক্ত করে।

235. Stack বেশিরভাগ প্রসেসরে নিচের দিকে গ্রো করে কেন?
উত্তর: কোড ও ডাটার বিপরীত দিকে stack বাড়িয়ে মেমরি সংঘর্ষ কমানোর জন্য।

236. Nested interrupt CPU কীভাবে হ্যান্ডেল করে?
উত্তর: একের পর এক interrupt-এর রিটার্ন অ্যাড্রেস ও রেজিস্টার state stack-এ সেভ করে, প্রায়োরিটি অনুযায়ী সার্ভিস করে।

237. কোন ইনস্ট্রাকশন word-এর ভিতরে byte swap করতে পারে?
উত্তর: XCHG রেজিস্টারগুলোর মধ্যে swap করে; কিছু আর্কিটেকচারে বিশেষ byte swap ইনস্ট্রাকশনও থাকে (৩৮০x86-এ BSWAP ইত্যাদি)।

238. Control unit হার্ডওয়্যারড বা microprogrammed – দুই ধরনের কেন?
উত্তর: Hardwired দ্রুত কিন্তু কম flexible; microprogrammed একটু ধীর কিন্তু সহজে পরিবর্তনযোগ্য ও জটিল instruction সাপোর্ট করে।

239. Synchronous আর Asynchronous interrupt-এর পার্থক্য কী?
উত্তর: Synchronous interrupt কোনো instruction-এর execution-এর সাথে জড়িত (যেমন divide by zero); Asynchronous বাহ্যিক ইভেন্ট থেকে আসে (যেমন I/O ডিভাইস)।

240. Instruction fusion কীভাবে performance বাড়ায়?
উত্তর: একাধিক সাধারণ instruction মিলিয়ে একটি micro-op হিসেবে execute করে, pipeline overhead কমায়।

241. CPU multi-level cache কেন ব্যবহার করে?
উত্তর: L1 খুব দ্রুত ও ছোট, L2/L3 একটু ধীর কিন্তু বড় – এইভাবে latency আর capacity-এর ব্যালান্স করা হয়।

242. Branch folding কী?
উত্তর: যখন branch-এর ফলাফল আগে থেকেই জানা, তখন branch instruction আলাদা করে execute না করে সরাসরি target পথে এগিয়ে যাওয়া।

243. Atomic operation সাপোর্ট করে কোন ইনস্ট্রাকশনগুলো?
উত্তর: XCHG, CMPXCHG এবং LOCK prefix-যুক্ত ইনস্ট্রাকশন।

244. Stack frame pointer debugging-এ কেন উপকারী?
উত্তর: প্রত্যেক ফাংশনের লোকাল ভ্যারিয়েবল ও কল ইতিহাস সহজে ট্রেস করা যায়।

245. Superscalar pipeline কীভাবে একাধিক instruction হ্যান্ডেল করে?
উত্তর: স্বাধীন ইনস্ট্রাকশনগুলোকে একাধিক execution unit-এ parallelভাবে পাঠিয়ে একসাথে execute করে।

যে টপিকগুলো নিয়ে তোমার এক্সাম বেশি ফোকাস করে (যেমন 8085/8086, interrupt, addressing mode, pipelining, cache, ইত্যাদি), চাইলে সেগুলোর ছোট ছোট বাংলা নোট আলাদা করে গুছিয়ে করে দিতে পারি 🫡