① Execution steps of 32-bit and 64-bit

⇒ 1) 32-bit uses registers like EAX
2) 64-bit uses registers like RAX
- 64-bit supports more memory and more registers
- Execution is similar, but 64-bit is more power.

② explanation of section.data, section.text, section.bss

⇒ section.data — initialized data
section.text — code / instructions
section.bss — uninitialized data

③ Why '2' in disp_msg 2?

⇒ It's parameter passed to the macro.

④ Why %1 and %2 in macro

⇒ They are placeholders for macro parameters.

⑤ Types of directives

⇒ Tell the assembler how to process code.
eg- global, section, mov

⑥ Diff. betn ax, eax, rax

⇒ AX — 16 bit
EAX — 32-bit
RAX — 64-bit

⑦ Full form of NASM — Netwide Assembler

⑧ string operation (explain)

⇒ Instructions for handling strings, like MOVS, LODS, STOS

(9) Meaning of resb, reslw
⇒ resb - reserve byte(s)
reslw - reserve word

(10) Why global-start?
⇒ It marks the entry point of the program

(11) Why use macro?
⇒ for code ~~reuse~~ reuse and simplification

(12) Macro vs procedure
⇒ macro - expanded at compile time
procedure - called at runtime

(13) divide instruction
⇒ Divides a number by a register or memory value

(14) DIV syntax
⇒ DIV reg/mem

(15) ADD instruction
⇒ Adds two values Ex- ADD AX, BX

(16) 80386 flag Register
⇒ stores status flags like zero, carry, sign, overflow

(17) SUB instruction with syntax
⇒ subtracts two values Ex- SUB AX, BX

(18) What is DEC instruction
⇒ Decreases a value by 1
Ex- DEC AX

(19) CMP instruction
⇒ compares two values sets flags but doesn't store result

(20) OR instruction
⇒ Bitwise OR. Ex - OR AX, BX

(21) XOR instruction
⇒ Bitwise exclusive OR
Ex - XOR AX, AX (sets AX to 0)

(22) Addressing modes
⇒ Ways to access data like
Immediate, Register, Direct, indirect

(23) syntax of Addressing modes
⇒ MOV AX, BX or MOV AX, [123GH]

(24) Ex of Addressing Modes
⇒ Immediate : MOV AX, 5
Register : MOV AX, BX
Direct : MOV AX, [1234H]

(25) Meaning of [] bratckets
⇒ It means memory access

(26) Conditional instructions
⇒ JE, JNE, JZ, JNZ, JG, JL

(27) Unconditional instructions
⇒ JMP (always jump)

(28) Ex of cond^n JUMP
⇒ JE label (jump if equal)

29) Ex of unconditional JUMP
⇒ JMP label

30) command line argument
⇒ Input passed to a program while running it

31) Rotate instruction
⇒ shifts bits circularly, Ex ROL, ROR

32) Ex of Rotate instruction
ROL AX,1 ROR AX,1 , RCL , RCR

33) GDTR LDTR TR MSW register
⇒ special register used in protected mode
GDTR - Global Discriptor Table Register
LDTR - local Disriptor Table Register

34) Features of 80386
⇒ 32-bit processor, paging, protected mode support

35) Flag registers of 8086 & 80386
8086 has 16 bit flags
80386 has more flags and is 32-bit

36) Memory size of 8086 & 80386
8086 - 1 MB
80386 - 4 GB

37) protected Mode
⇒ Advanced mode with memory protection

38) paging mechanism
⇒ Divides memory into pages for efficient use

(39) Control register
⇒ CR0 to CR4 : control CPU functioning

(40) Descriptors , selector
⇒ Descriptors - memory information
selector - index to descriptor

(41) Bus size of 8086
Address - 20 bit
Data - 16 bit

(42) Bus size of 80386
⇒ Address - 32 bit
Data - 32 bit

(43) Assembly directives
⇒ DB (byte), DW (word), DD (double word)
DQ (quad word)

(44) Convert protected to real mode
⇒ Clear PE (protection Enable) bit in CR0

(45) control register for paging
⇒ CR_3

(46) Size of GDTR / IDTR , LDT
⇒ GDTR / IDTR - 6 bytes
LDT - varies based on entries

(47) Hex 1 to 9 to ASCII
⇒ Add 30H
Ex - 1 = 31 H , 2 = 32 H - - 9 = 39 = H

(48) Hex A to F to ASCII
⇒ Add 37H.
A = 41H , B = 42H - - - F = 46H

(49) ⇒
(50) ⇒
(51) ⇒
(52) ⇒
(53) ⇒
(54) ⇒
(55) ⇒
(56) ⇒
(57) ⇒

(49) Why convert Hex to ASCII
⇒ To display hex values as readable text

(50) 8086 disadvantages, 80386 advantages
⇒ 8086 - Limited memory (1MB)
No protection
80386 - 4GB memory, support multitasking, protection

(51) Data Transfer instruction
⇒ Move data: MOV, PUSH, POP, IN, OUT

(52) Arithmetic instructions
⇒ Do Math - ADD, SUB, MUL, DIV, INC, DEC

(53) Bit manipulation instruction
⇒ change bits - AND, OR, XOR, NOT, SHL, SHR

(54) program execution Transfer instruction
⇒ control flow - JMP, CALL, RET, JE, JNE

(55) String instruction
⇒ Work on strings - MOVS, LODS, STOS, SCAS, CMPS

(56) processor control instruction
⇒ control CPU - HLT, WAIT, LOCK, STC, CLC

(57) Page, Page table, page directory sizes
⇒ Page - 4kB
Page table - 1024 entries
Page directory - 1024 entries

(12) Multitasking vs Multithreading

⇒ Multitasking - Many program
Multithreading - Many task in one program

(13) privilege level

⇒ 0 - most powerful (os)
3 - least (user)

(14) TLB Buffer

⇒ cache for page tables, faster memory access

Steps for execution

(1) nasm -f elf64 -o disp.o disp asm
converts ASM to object file

(2) ld -o disp disp.o
- Links object file, creates executable

(3) ./disp
Runs the program.