Intel Assembly Language - Chapters 2-3 MCQ Questions

Computer-Based Test (CBT) Preparation

Total Questions: 200
Chapters Covered: Chapter 2 (x86 Processor Architecture) & Chapter 3 (Assembly Language Fundamentals)

CHAPTER 2: x86 PROCESSOR ARCHITECTURE

Section 2.1: General Concepts

Question 1: What are the three basic operations in the instruction execution cycle?

- a) Load, Store, Execute
- b) Fetch, Decode, Execute
- c) Read, Write, Process
- d) Input, Process, Output

Answer: b) Fetch, Decode, Execute

Question 2: Which component of the CPU performs arithmetic operations such as addition and subtraction?

- a) Control Unit (CU)
- b) Instruction Pointer (IP)
- c) Arithmetic Logic Unit (ALU)
- d) Memory Management Unit (MMU)

Answer: c) Arithmetic Logic Unit (ALU)

Question 3: What is the purpose of the clock in a CPU?

- a) To display the current time
- b) To synchronize internal operations of the CPU
- c) To count the number of instructions executed
- d) To manage power consumption

Answer: b) To synchronize internal operations of the CPU

Question 4: How many clock cycles does reading a single value from memory typically require?

- a) 1 clock cycle
- b) 2 clock cycles
- c) 4 clock cycles
- d) 8 clock cycles

Answer: c) 4 clock cycles

Question 5: What is a bus in computer architecture?

- a) A vehicle for transporting data
- b) A group of parallel wires that transfer data
- c) A type of memory storage
- d) A processing unit

Answer: b) A group of parallel wires that transfer data

Question 6: Which bus holds the addresses of instructions and data?

- a) Data bus
- b) Control bus
- c) Address bus
- d) I/O bus

Answer: c) Address bus

Question 7: What is a machine cycle also known as?

- a) Instruction cycle
- b) Clock cycle
- c) Memory cycle
- d) Bus cycle

Answer: b) Clock cycle

Question 8: If a clock oscillates at 1 GHz, what is the duration of one clock cycle?

- a) 1 second
- b) 1 millisecond

- c) 1 microsecond
- d) 1 nanosecond

Answer: d) 1 nanosecond

Question 9: What are wait states in the instruction execution cycle?

- a) Pauses for user input
- b) Empty clock cycles due to speed differences between CPU and memory
- c) Periods of system hibernation
- d) Interrupts from the operating system

Answer: b) Empty clock cycles due to speed differences between CPU and memory

---

Question 10: Which step in loading and executing a program involves the operating system searching for the program's filename?

- a) Step 1
- b) Step 2
- c) Step 3
- d) Step 4

Answer: a) Step 1

Section 2.2: 32-Bit x86 Processors

Question 11: What are the three primary modes of operation for x86 processors?

- a) User, Kernel, Supervisor
- b) Real, Virtual, Protected
- c) Protected, Real-address, System management
- d) Execution, Protected, Virtual

Answer: c) Protected, Real-address, System management

Question 12: In which mode does the processor run MS-DOS programs?

- a) Protected mode
- b) Real-address mode
- c) System management mode

- d) Virtual-8086 mode

Answer: b) Real-address mode

Question 13: What is the maximum amount of memory accessible in Real-address mode?

- a) 64 KB
- b) 640 KB
- c) 1 MB
- d) 4 GB

Answer: c) 1 MB

Question 14: Which mode allows the processor to directly execute real-address mode software such as MS-DOS programs while in Protected mode?

- a) System management mode
- b) Virtual-8086 mode
- c) Real-address mode
- d) Compatibility mode

Answer: b) Virtual-8086 mode

Question 15: What is the primary purpose of System Management Mode (SMM)?

- a) To run legacy DOS applications
- b) To implement operating system functions
- c) To implement power management and system security
- d) To provide virtual memory support

Answer: c) To implement power management and system security

Question 16: How much linear address space can a program access in Protected mode?

- a) 1 MB
- b) 16 MB
- c) 4 GB
- d) 1 TB

Answer: c) 4 GB

Question 17: Which Intel processor was the first to support Protected mode?

- a) Intel 8086
- b) Intel 80286
- c) Intel 80386
- d) Intel 80486

Answer: b) Intel 80286

Question 18: What does the term "flat segmentation model" mean in Protected mode?

- a) All segments are the same size
- b) All segments are mapped to the entire 32-bit physical address space
- c) Memory is divided into equal segments
- d) Segments cannot overlap

Answer: b) All segments are mapped to the entire 32-bit physical address space

---

Question 19: In Protected mode, which mechanism prevents application programs from accessing system data?

- a) Encryption
- b) Segment protection
- c) Memory locks
- d) Virtual addressing

Answer: b) Segment protection

Question 20: What is paging in the context of Protected mode?

- a) A technique to print documents
- b) A mechanism that allows a segment to be divided into 4096-byte blocks of memory
- c) A method to organize disk storage
- d) A way to number memory locations

Answer: b) A mechanism that allows a segment to be divided into 4096-byte blocks of memory

---

Section 2.3: 64-Bit x86-64 Processors

Question 21: How much linear address space can a 64-bit processor theoretically access?

- a) 4 GB
- b) 16 GB
- c) 16 exabytes
- d) Unlimited

Answer: c) 16 exabytes

Question 22: What is the native mode called in 64-bit processors?

- a) 64-bit mode
- b) Long mode
- c) Extended mode
- d) Hyper mode

Answer: b) Long mode

Question 23: In 64-bit mode, what is the default operand size?

- a) 16 bits
- b) 32 bits
- c) 64 bits
- d) 128 bits

Answer: b) 32 bits

Question 24: How many general-purpose registers are available in 64-bit mode?

- a) 8
- b) 12
- c) 16
- d) 32

Answer: c) 16

Question 25: What is compatibility mode in x86-64 processors?

- a) A mode that runs 16-bit programs

- b) A sub-mode of Long mode that permits legacy 32-bit programs to run
- c) A mode for backward compatibility with 8086
- d) A mode that emulates other processor types

Answer: b) A sub-mode of Long mode that permits legacy 32-bit programs to run

---

Section 2.4: Components of an x86 Microcomputer

Question 26: What is the motherboard?

- a) The main processing chip
- b) The main circuit board of the computer
- c) The power supply unit
- d) The graphics card

Answer: b) The main circuit board of the computer

Question 27: What does the chipset do in a microcomputer?

- a) Provides power to the CPU
- b) Contains the BIOS
- c) Controls the flow of data between CPU, memory, and peripherals
- d) Stores the operating system

Answer: c) Controls the flow of data between CPU, memory, and peripherals

---

Question 28: What is PCI Express primarily used for?

- a) Connecting keyboards and mice
- b) Connecting high-speed video cards and hard drives
- c) Connecting printers
- d) Connecting monitors

Answer: b) Connecting high-speed video cards and hard drives

Question 29: What type of memory is the CPU cache?

- a) ROM
- b) SRAM (Static RAM)
- c) DRAM (Dynamic RAM)

- d) Flash memory

Answer: b) SRAM (Static RAM)

Question 30: What is the difference between L1 and L2 cache?

- a) L1 is faster but smaller than L2
- b) L2 is faster but smaller than L1
- c) L1 is for data, L2 is for instructions
- d) There is no difference

Answer: a) L1 is faster but smaller than L2

Question 31: What does USB stand for?

- a) Universal System Bus
- b) Universal Serial Bus
- c) Unified Storage Bus
- d) Universal Synchronous Bus

Answer: b) Universal Serial Bus

Question 32: Which Intel chipset component handles I/O functions?

- a) Northbridge
- b) Memory Controller Hub
- c) I/O Controller Hub (ICH)
- d) Graphics Controller

Answer: c) I/O Controller Hub (ICH)

Question 33: What is the purpose of the system clock generator?

- a) To keep track of time
- b) To provide timing signals to synchronize system components
- c) To generate random numbers
- d) To control the power supply

Answer: b) To provide timing signals to synchronize system components

---

Question 34: What does CMOS stand for in the context of BIOS?

- a) Central Memory Operating System
- b) Complementary Metal-Oxide Semiconductor
- c) Computer Memory Output System
- d) Central Motherboard Operating Sequence

Answer: b) Complementary Metal-Oxide Semiconductor

Question 35: What information is typically stored in CMOS memory?

- a) The operating system
- b) System configuration information and hardware settings
- c) User documents
- d) Application programs

Answer: b) System configuration information and hardware settings

---

Section 2.5: Input-Output System

Question 36: What are the three levels of I/O access in PC systems?

- a) Hardware, Firmware, Software
- b) High-level, Mid-level, Low-level
- c) Application, Operating System, BIOS
- d) User, Kernel, Driver

Answer: c) Application, Operating System, BIOS

Question 37: Which level of I/O provides the fastest access but requires detailed hardware knowledge?

- a) High-level language functions
- b) Operating system functions
- c) BIOS functions
- d) Hardware level

Answer: d) Hardware level

Question 38: What does BIOS stand for?

- a) Basic Integrated Operating System
- b) Binary Input/Output System
- c) Basic Input/Output System
- d) Built-In Operating System

Answer: c) Basic Input/Output System

Question 39: How are BIOS functions typically called in assembly language?

- a) Using CALL instructions
- b) Using software interrupts (INT instruction)
- c) Using jump instructions
- d) Using procedure calls

Answer: b) Using software interrupts (INT instruction)

Question 40: Which BIOS interrupt is commonly used for video services?

- a) INT 10h
- b) INT 16h
- c) INT 21h
- d) INT 13h

Answer: a) INT 10h

Question 41: Which BIOS interrupt handles keyboard services?

- a) INT 10h
- b) INT 16h
- c) INT 21h
- d) INT 13h

Answer: b) INT 16h

Question 42: In MS-DOS, which interrupt number is used for most DOS services?

- a) INT 10h
- b) INT 16h
- c) INT 21h

- d) INT 13h

Answer: c) INT 21h

Question 43: What is the primary advantage of using high-level language I/O functions?

- a) Faster execution
- b) Direct hardware control
- c) Portability across platforms
- d) Lower memory usage

Answer: c) Portability across platforms

Question 44: What is the main disadvantage of using BIOS-level I/O?

- a) It's too fast
- b) It's slow and not suitable for graphical applications
- c) It requires special permissions
- d) It only works on specific processors

Answer: b) It's slow and not suitable for graphical applications

---

Question 45: In protected mode operating systems, which method is typically used for I/O operations?

- a) Direct hardware access
- b) BIOS interrupts
- c) Operating system API calls
- d) Assembly language only

Answer: c) Operating system API calls

Section 2.6: x86 Memory Management

Question 46: In Real-address mode, how is a physical address calculated from a segment:offset pair?

- a) Segment + offset
- b) (Segment × 16) + offset
- c) Segment × offset

- d) Segment OR offset

Answer: b) (Segment × 16) + offset

Question 47: If a segment register contains 0300h and the offset is 0040h, what is the linear address?

- a) 0340h
- b) 3040h
- c) 3400h
- d) 3000h

Answer: b) 3040h

Question 48: What is the maximum value an offset can have in Real-address mode?

- a) FFFFh (65,535)
- b) FFh (255)
- c) FFFh (4,095)
- d) FFFFFFFFh (4,294,967,295)

Answer: a) FFFFh (65,535)

Question 49: How large is a segment in Real-address mode?

- a) 64 KB
- b) 128 KB
- c) 256 KB
- d) 1 MB

Answer: a) 64 KB

Question 50: What is the maximum physical address accessible in Real-address mode?

- a) FFFFh
- b) FFFFFh (1 MB)
- c) FFFFFFh (16 MB)
- d) FFFFFFFFh (4 GB)

Answer: b) FFFFFh (1 MB)

Question 51: In Protected mode, what does a segment selector point to?

- a) Physical memory address
- b) A descriptor table entry
- c) An offset value
- d) A page table

Answer: b) A descriptor table entry

Question 52: What information does a segment descriptor contain?

- a) Only the segment size
- b) Base address, segment limit, and access rights
- c) Just the starting address
- d) Only permission bits

Answer: b) Base address, segment limit, and access rights

Question 53: What is a linear address in Protected mode?

- a) The same as physical address
- b) A 32-bit address before paging translation
- c) The offset within a segment
- d) A 16-bit address

Answer: b) A 32-bit address before paging translation

Question 54: When paging is enabled in Protected mode, what translates a linear address to a physical address?

- a) Segment selector
- b) Descriptor table
- c) Page directory and page tables
- d) CPU directly

Answer: c) Page directory and page tables

Question 55: What is the size of a page in x86 paging?

- a) 1 KB

- b) 2 KB
- c) 4 KB
- d) 8 KB

Answer: c) 4 KB

Section 2.7: x86 Registers

Question 56: How many general-purpose registers are there in 32-bit x86 processors?

- a) 4
- b) 6
- c) 8
- d) 16

Answer: c) 8

Question 57: What are the 32-bit general-purpose registers?

- a) AX, BX, CX, DX, SI, DI, BP, SP
- b) EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
- c) RAX, RBX, RCX, RDX, RSI, RDI, RBP, RSP
- d) AL, BL, CL, DL, AH, BH, CH, DH

Answer: b) EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP

Question 58: What is the traditional use of the EAX register?

- a) Stack pointer
- b) Accumulator for arithmetic operations
- c) Base pointer
- d) Counter for loops

Answer: b) Accumulator for arithmetic operations

Question 59: Which register is commonly used as a loop counter?

- a) EAX
- b) EBX
- c) ECX

- d) EDX

Answer: c) ECX

Question 60: What does the ESP register point to?

- a) The beginning of the stack
- b) The top of the stack
- c) The base of the current stack frame
- d) The end of memory

Answer: b) The top of the stack

Question 61: Which register is used as the base pointer for stack frames?

- a) ESP
- b) EBP
- c) ESI
- d) EDI

Answer: b) EBP

Question 62: What are ESI and EDI commonly used for?

- a) Arithmetic operations
- b) String and array operations (source and destination)
- c) Loop counters
- d) Address calculations

Answer: b) String and array operations (source and destination)

Question 63: How can you access the lower 16 bits of EAX?

- a) EAX
- b) AX
- c) AL
- d) AH

Answer: b) AX

Question 64: What does the AL register represent?

- a) The entire EAX register
- b) The lower 16 bits of EAX
- c) The lower 8 bits of EAX
- d) The upper 8 bits of AX

Answer: c) The lower 8 bits of EAX

Question 65: Which segment register traditionally holds the address of the current code segment?

- a) DS
- b) SS
- c) CS
- d) ES

Answer: c) CS

Question 66: What does the SS segment register point to?

- a) Code segment
- b) Data segment
- c) Stack segment
- d) Extra segment

Answer: c) Stack segment

Question 67: How many segment registers are there in x86 processors?

- a) 4
- b) 6
- c) 8
- d) 16

Answer: b) 6 (CS, DS, SS, ES, FS, GS)

Question 68: What does the EIP register contain?

- a) The current instruction being executed

- b) The address of the next instruction to execute
- c) The address of the last instruction executed
- d) The number of instructions executed

Answer: b) The address of the next instruction to execute

Question 69: What is the EFLAGS register used for?

- a) Storing arithmetic results
- b) Containing status and control flags
- c) Addressing memory
- d) Counting iterations

Answer: b) Containing status and control flags

Question 70: Which flag is set when an arithmetic operation produces a carry out of the most significant bit?

- a) Zero flag
- b) Sign flag
- c) Carry flag
- d) Overflow flag

Answer: c) Carry flag

Question 71: When is the Zero flag (ZF) set?

- a) When the result is negative
- b) When the result is zero
- c) When there's a carry
- d) When there's an overflow

Answer: b) When the result is zero

Question 72: What does the Sign flag (SF) indicate?

- a) The carry state
- b) Whether the result is zero
- c) The sign of the result (1 = negative, 0 = positive)

- d) Whether overflow occurred

Answer: c) The sign of the result (1 = negative, 0 = positive)

Question 73: The Overflow flag (OF) is set when:

- a) The result is zero
- b) There's a carry
- c) A signed arithmetic operation generates a result too large to fit in the destination
- d) The result is negative

Answer: c) A signed arithmetic operation generates a result too large to fit in the destination

---

Question 74: What does the Parity flag (PF) reflect?

- a) Whether the number is even or odd
- b) Whether the lowest byte has an even or odd number of 1 bits
- c) The sign of the result
- d) Whether there's a carry

Answer: b) Whether the lowest byte has an even or odd number of 1 bits

---

Question 75: Which flag is used to control the direction of string operations?

- a) Carry flag
- b) Direction flag
- c) Trap flag
- d) Interrupt flag

Answer: b) Direction flag

CHAPTER 3: ASSEMBLY LANGUAGE FUNDAMENTALS

Section 3.1: Basic Elements of Assembly Language

Question 76: What are the three basic types of statements in assembly language?

- a) Instructions, variables, constants
- b) Instructions, directives, macros
- c) Commands, functions, procedures

- d) Operations, declarations, definitions

Answer: b) Instructions, directives, macros

Question 77: What is an instruction in assembly language?

- a) A command to the assembler
- b) A statement that is executed by the processor at runtime
- c) A memory allocation command
- d) A comment for documentation

Answer: b) A statement that is executed by the processor at runtime

---

Question 78: What is a directive (or pseudo-operation)?

- a) An instruction executed by the CPU
- b) A command that is recognized by the assembler
- c) A loop control structure
- d) A conditional branch

Answer: b) A command that is recognized by the assembler

Question 79: Which of the following is a valid identifier in assembly language?

- a) 123abc
- b) my-variable
- c) myVariable
- d) for

Answer: c) myVariable

Question 80: What character is typically used to start a comment in MASM?

- a) //
- b) /*
- c) ;
- d) #

Answer: c) ;

Question 81: Which directive is used to define a byte variable in MASM?

- a) BYTE
- b) DB
- c) DWORD
- d) DEFINE_BYTE

Answer: b) DB (Define Byte)

Question 82: What does the DWORD directive define?

- a) A double word (8 bytes)
- b) A doubleword (4 bytes)
- c) A word (2 bytes)
- d) A byte (1 byte)

Answer: b) A doubleword (4 bytes)

Question 83: In the instruction `mov eax, ebx`, what is 'mov'?

- a) The destination operand
- b) The source operand
- c) The mnemonic (operation)
- d) A register name

Answer: c) The mnemonic (operation)

Question 84: How many operands does the MOV instruction typically have?

- a) 0
- b) 1
- c) 2
- d) 3

Answer: c) 2

Question 85: In the instruction `add eax, 5`, what is '5'?

- a) A register
- b) An immediate operand (constant)

- c) A memory address
- d) A label

Answer: b) An immediate operand (constant)

Question 86: Which of the following is NOT a valid assembly language statement?

- a) mov eax, ebx
- b) count DB 100
- c) ; This is a comment
- d) 5add eax, ebx

Answer: d) 5add eax, ebx

Question 87: What is the purpose of a label in assembly language?

- a) To comment code
- b) To mark a position in code or data
- c) To define a variable
- d) To import libraries

Answer: b) To mark a position in code or data

Question 88: Which directive marks the beginning of the code segment?

- a) .CODE
- b) .DATA
- c) .STACK
- d) BEGIN

Answer: a) .CODE

Question 89: What does the .DATA directive indicate?

- a) The start of executable code
- b) The start of the data segment for variables
- c) The start of the stack
- d) The end of the program

Answer: b) The start of the data segment for variables

Question 90: Which directive ends a procedure definition?

- a) END
- b) ENDP
- c) RET
- d) EXIT

Answer: b) ENDP

Section 3.2: Example: Adding Three Integers

Question 91: Which instruction is used to transfer data between registers or from memory to register?

- a) ADD
- b) SUB
- c) MOV
- d) XOR

Answer: c) MOV

Question 92: In the instruction `mov eax, 10`, what happens?

- a) 10 is moved to memory at address EAX
- b) The value 10 is loaded into register EAX
- c) EAX is moved to location 10
- d) EAX and 10 are added

Answer: b) The value 10 is loaded into register EAX

Question 93: What does the ADD instruction do?

- a) Multiplies two operands
- b) Adds the source operand to the destination operand
- c) Subtracts the source from destination
- d) Compares two operands

Answer: b) Adds the source operand to the destination operand

---

Question 94: After executing `mov eax, 5` followed by `add eax, 3`, what value is in EAX?

- a) 5
- b) 3
- c) 8
- d) 15

Answer: c) 8

Question 95: Can you move data directly from one memory location to another using a single MOV instruction?

- a) Yes, always
- b) No, you must use a register as an intermediary
- c) Yes, but only in protected mode
- d) Yes, but only for byte-sized data

Answer: b) No, you must use a register as an intermediary

Question 96: Which of the following is a valid MOV instruction?

- a) mov 5, eax
- b) mov eax, ebx
- c) mov cs, ds
- d) mov mem1, mem2

Answer: b) mov eax, ebx

Question 97: What happens when you execute `add eax, eax`?

- a) EAX is cleared to zero
- b) EAX is doubled
- c) EAX remains unchanged
- d) An error occurs

Answer: b) EAX is doubled

Question 98: Which instruction subtracts the source operand from the destination?

- a) ADD
- b) SUB

- c) DEC
- d) NEG

Answer: b) SUB

Question 99: After `mov eax, 10` and `sub eax, 3`, what is in EAX?

- a) 3
- b) 7
- c) 10
- d) 13

Answer: b) 7

Question 100: What does the INC instruction do?

- a) Decreases the operand by 1
- b) Increases the operand by 1
- c) Inverts all bits
- d) Initializes the operand

Answer: b) Increases the operand by 1

Section 3.3: Assembling, Linking, and Running Programs

Question 101: What is the file extension for an assembly source file in MASM?

- a) .exe
- b) .obj
- c) .asm
- d) .lst

Answer: c) .asm

Question 102: What does an assembler produce from a source file?

- a) An executable file
- b) An object file
- c) A library file
- d) A header file

Answer: b) An object file

Question 103: What is the typical extension of an object file?

- a) .exe
- b) .obj or .o
- c) .asm
- d) .dll

Answer: b) .obj or .o

Question 104: What does a linker do?

- a) Assembles source code
- b) Combines object files and libraries into an executable
- c) Debugs the program
- d) Optimizes machine code

Answer: b) Combines object files and libraries into an executable

---

Question 105: What is the typical extension of an executable file in Windows?

- a) .obj
- b) .asm
- c) .exe
- d) .com

Answer: c) .exe

Question 106: What is a listing file?

- a) The executable program
- b) A file showing source code, machine code, and addresses
- c) A library of functions
- d) A configuration file

Answer: b) A file showing source code, machine code, and addresses

---

Question 107: What file extension is typically used for listing files?

- a) .lst
- b) .txt
- c) .log
- d) .out

Answer: a) .lst

Question 108: Which program would you use to step through assembly code line by line?

- a) Assembler
- b) Linker
- c) Debugger
- d) Editor

Answer: c) Debugger

Question 109: What is the purpose of a make file or build script?

- a) To write source code
- b) To automate the assembly and linking process
- c) To debug programs
- d) To create documentation

Answer: b) To automate the assembly and linking process

Question 110: In a two-pass assembler, what is done in the first pass?

- a) Generate machine code
- b) Build a symbol table and determine addresses
- c) Link object files
- d) Optimize code

Answer: b) Build a symbol table and determine addresses

Section 3.4: Defining Data

Question 111: Which directive defines a byte (8-bit) variable?

- a) DW
- b) DD

- c) DB
- d) DQ

Answer: c) DB

Question 112: What does DW stand for?

- a) Define Word (16 bits)
- b) Double Word
- c) Data Write
- d) Declare Word

Answer: a) Define Word (16 bits)

Question 113: How many bytes does DD (Define Doubleword) allocate?

- a) 1 byte
- b) 2 bytes
- c) 4 bytes
- d) 8 bytes

Answer: c) 4 bytes

Question 114: Which directive would you use to define a 64-bit integer?

- a) DB
- b) DW
- c) DD
- d) DQ

Answer: d) DQ (Define Quadword)

Question 115: What does the following define? `value DB 25`

- a) A word variable initialized to 25
- b) A byte variable initialized to 25
- c) A doubleword variable initialized to 25
- d) An array of 25 bytes

Answer: b) A byte variable initialized to 25

Question 116: How do you define an array of bytes in MASM?

- a) array DB 10 DUP(0)
- b) array[10] DB 0
- c) DB array[10]
- d) ARRAY 10, 0

Answer: a) array DB 10 DUP(0)

Question 117: What does `DUP` do in a data definition?

- a) Duplicates the variable name
- b) Creates multiple copies of the initial value
- c) Defines an unsigned pointer
- d) Declares a user procedure

Answer: b) Creates multiple copies of the initial value

Question 118: What does `buffer DB 100 DUP(?)` create?

- a) An uninitialized buffer of 100 bytes
- b) A buffer of 100 bytes all set to 0
- c) A buffer of 100 bytes all set to ?
- d) 100 different variables

Answer: a) An uninitialized buffer of 100 bytes

Question 119: How do you define a string in MASM?

- a) Using DB with quoted text
- b) Using STRING directive
- c) Using DW with quoted text
- d) Strings cannot be defined in assembly

Answer: a) Using DB with quoted text

Question 120: What does `message DB "Hello", 0` define?

- a) A byte variable

- b) A null-terminated string
- c) An array of integers
- d) A word variable

Answer: b) A null-terminated string

Question 121: What is the purpose of the EQU directive?

- a) To allocate memory
- b) To create a symbolic constant
- c) To define a procedure
- d) To end a program

Answer: b) To create a symbolic constant

Question 122: In `MAXVAL EQU 100`, what is MAXVAL?

- a) A variable
- b) A label
- c) A symbolic constant (equate)
- d) A procedure name

Answer: c) A symbolic constant (equate)

Question 123: What is the difference between EQU and = in MASM?

- a) There is no difference
- b) EQU cannot be redefined, = can be
- c) = cannot be redefined, EQU can be
- d) EQU is for numbers, = is for strings

Answer: b) EQU cannot be redefined, = can be

Question 124: Which directive creates a real number (floating-point) variable?

- a) DB
- b) DW
- c) REAL4, REAL8, or REAL10
- d) DF

Answer: c) REAL4, REAL8, or REAL10

Question 125: What does REAL8 define?

- a) An 8-bit real number
- b) An 8-byte (64-bit) double-precision real number
- c) An 8-element array
- d) An 8-bit integer

Answer: b) An 8-byte (64-bit) double-precision real number

Section 3.5: Symbolic Constants

Question 126: What is a symbolic constant?

- a) A variable that can change
- b) A named constant that represents a value
- c) A memory address
- d) A CPU register

Answer: b) A named constant that represents a value

Question 127: Why use symbolic constants instead of literal values?

- a) They execute faster
- b) They make code more readable and easier to maintain
- c) They use less memory
- d) They are required by the assembler

Answer: b) They make code more readable and easier to maintain

---

Question 128: Can you use a symbolic constant defined with EQU in an arithmetic expression?

- a) No, never
- b) Yes, the assembler will substitute its value
- c) Only in data definitions
- d) Only in conditional statements

Answer: b) Yes, the assembler will substitute its value

Question 129: What happens if you try to redefine a symbol defined with EQU?

- a) The assembler uses the new value
- b) The assembler generates an error
- c) The assembler ignores it
- d) The assembler generates a warning

Answer: b) The assembler generates an error

Question 130: Which is better for defining the size of an array: a literal number or a symbolic constant?

- a) Literal number, it's clearer
- b) Symbolic constant, it's easier to modify
- c) They are equivalent
- d) Neither, use a variable

Answer: b) Symbolic constant, it's easier to modify

Section 3.6: Real-Address Mode Programming

Question 131: What does INT stand for in assembly language?

- a) Integer
- b) Internal
- c) Interrupt
- d) Interface

Answer: c) Interrupt

Question 132: Which interrupt is used to terminate a DOS program?

- a) INT 10h
- b) INT 16h
- c) INT 21h, function 4Ch
- d) INT 20h

Answer: c) INT 21h, function 4Ch

Question 133: In real-address mode, how do you typically specify which DOS function to call?

- a) By setting a register value before the INT instruction
- b) By using different interrupt numbers
- c) By using different opcodes
- d) By passing parameters on the stack

Answer: a) By setting a register value before the INT instruction (usually AH)

Question 134: Which register typically holds the function number for INT 21h DOS services?

- a) AL
- b) AH
- c) BX
- d) CX

Answer: b) AH

Question 135: What does INT 21h, function 2 do?

- a) Read a character
- b) Write a character to standard output
- c) Terminate program
- d) Get system time

Answer: b) Write a character to standard output

Question 136: To display a string using INT 21h, function 9, where should the string address be placed?

- a) In AX
- b) In DS:DX
- c) In ES:DI
- d) On the stack

Answer: b) In DS:DX

Question 137: What character should terminate a string for INT 21h, function 9?

- a) Null (0)
- b) Dollar sign ($)

- c) Carriage return (13)
- d) Line feed (10)

Answer: b) Dollar sign ($)

Question 138: Which INT 10h function sets the video mode?

- a) Function 00h
- b) Function 01h
- c) Function 02h
- d) Function 06h

Answer: a) Function 00h

Question 139: What does INT 16h, function 0 do?

- a) Display a character
- b) Wait for and read a keyboard character
- c) Check keyboard status
- d) Clear the keyboard buffer

Answer: b) Wait for and read a keyboard character

Question 140: After INT 16h, function 0, where is the ASCII code of the key pressed?

- a) In AH
- b) In AL
- c) In BL
- d) In DL

Answer: b) In AL

Section 3.7: Protected Mode Programming (Introduction)

Question 141: In protected mode, what replaces direct BIOS and DOS interrupt calls?

- a) Hardware interrupts
- b) Operating system API calls
- c) Direct memory access
- d) Nothing, they still work the same

Answer: b) Operating system API calls

Question 142: Which library does the textbook use for console I/O in protected mode examples?

- a) Standard C library
- b) Windows API
- c) Irvine32 library
- d) BIOS library

Answer: c) Irvine32 library

Question 143: What does the WriteString procedure from Irvine32 do?

- a) Writes a string to a file
- b) Displays a null-terminated string on the console
- c) Reads a string from the keyboard
- d) Converts a string to uppercase

Answer: b) Displays a null-terminated string on the console

Question 144: Which Irvine32 procedure reads an integer from the keyboard?

- a) ReadInt
- b) GetInt
- c) InputInt
- d) ScanInt

Answer: a) ReadInt

Question 145: What does the WriteDec procedure do?

- a) Writes a decimal point
- b) Displays an unsigned integer in decimal
- c) Decrements and writes a value
- d) Writes a hexadecimal number

Answer: b) Displays an unsigned integer in decimal

Question 146: Which procedure displays a value in hexadecimal format?

- a) WriteHex
- b) WriteInt
- c) WriteBin
- d) WriteOct

Answer: a) WriteHex

Question 147: What does Crlf (from Irvine32) do?

- a) Clears the screen
- b) Writes a carriage return and line feed
- c) Clears a register
- d) Creates a new file

Answer: b) Writes a carriage return and line feed

Question 148: Which procedure would you use to generate a random number in Irvine32?

- a) Random
- b) Randomize followed by RandomRange
- c) GenRandom
- d) MakeRandom

Answer: b) Randomize followed by RandomRange

Question 149: What does the Clrscr procedure do?

- a) Clears a register
- b) Clears the screen
- c) Clears the stack
- d) Clears a string

Answer: b) Clears the screen

Question 150: Before using Irvine32 procedures, what must you include in your program?

- a) include irvine32.inc
- b) import irvine32.lib
- c) using irvine32

- d) link irvine32.dll

Answer: a) include irvine32.inc

Additional Questions on Data Transfer and Arithmetic

Question 151: Which instruction exchanges the values of two operands?

- a) SWAP
- b) XCHG
- c) EXCHANGE
- d) SWITCH

Answer: b) XCHG

Question 152: What does `xchg eax, ebx` do?

- a) Copies EAX to EBX
- b) Adds EAX and EBX
- c) Swaps the contents of EAX and EBX
- d) Compares EAX and EBX

Answer: c) Swaps the contents of EAX and EBX

Question 153: Which instruction decrements a register by 1?

- a) SUB register, 1
- b) DEC register
- c) DECREMENT register
- d) Both a and b

Answer: d) Both a and b

Question 154: What is the advantage of using INC/DEC over ADD/SUB?

- a) They are faster and use less code space
- b) They can work with larger numbers
- c) They affect different flags
- d) There is no advantage

Answer: a) They are faster and use less code space

Question 155: Which instruction performs two's complement negation?

- a) NOT
- b) NEG
- c) NEGATE
- d) COMP

Answer: b) NEG

Question 156: After executing `mov al, 5` followed by `neg al`, what value is in AL?

- a) 5
- b) -5 (FBh in hex)
- c) 0
- d) 255

Answer: b) -5 (FBh in hex, which is 251 in unsigned)

Question 157: What does the MOVZX instruction do?

- a) Moves and zeros the destination
- b) Moves with zero extension (fills upper bits with 0)
- c) Moves only zero values
- d) Moves and XORs

Answer: b) Moves with zero extension (fills upper bits with 0)

Question 158: What is the purpose of MOVSX?

- a) Moves with sign extension
- b) Moves with XOR
- c) Moves sixteen bits
- d) Moves and swaps

Answer: a) Moves with sign extension

Question 159: If AL = 80h (10000000 binary) and you execute `movzx eax, al`, what is in EAX?

- a) 00000080h

- b) FFFFFF80h
- c) 80808080h
- d) 00000000h

Answer: a) 00000080h

Question 160: If AL = 80h (signed = -128) and you execute `movsx eax, al`, what is in EAX?

- a) 00000080h
- b) FFFFFF80h
- c) 80000000h
- d) 00000000h

Answer: b) FFFFFF80h

Questions on Addressing Modes

Question 161: In `mov eax, 25`, what addressing mode is used for the source operand?

- a) Register
- b) Immediate
- c) Direct
- d) Indirect

Answer: b) Immediate

Question 162: In `mov eax, ebx`, what addressing mode is used?

- a) Immediate
- b) Register
- c) Direct
- d) Indirect

Answer: b) Register

Question 163: In `mov eax, [var1]`, what addressing mode is used for the source?

- a) Immediate
- b) Register
- c) Direct (or displacement)

- d) Register indirect

Answer: c) Direct (or displacement)

Question 164: What does `mov eax, [ebx]` represent?

- a) Move EBX into EAX
- b) Move the value at the address contained in EBX into EAX
- c) Move EAX to the address in EBX
- d) Add EBX to EAX

Answer: b) Move the value at the address contained in EBX into EAX

Question 165: In register indirect addressing, what does the register contain?

- a) The actual data
- b) The address of the data
- c) An offset value
- d) A flag value

Answer: b) The address of the data

Question 166: What addressing mode does `mov eax, [ebx+4]` use?

- a) Register indirect
- b) Indexed
- c) Base-displacement (or base-plus-displacement)
- d) Direct

Answer: c) Base-displacement (or base-plus-displacement)

Question 167: In `mov eax, [ebx+esi]`, what is this addressing mode called?

- a) Direct
- b) Base-indexed
- c) Displacement
- d) Immediate

Answer: b) Base-indexed (or indexed)

Question 168: Which addressing mode allows you to access array elements efficiently?

- a) Immediate
- b) Register
- c) Indexed or base-displacement
- d) Direct only

Answer: c) Indexed or base-displacement

Question 169: Can you use ESP as an index register in indexed addressing?

- a) Yes, always
- b) No, ESP cannot be used as an index register
- c) Yes, but only in 64-bit mode
- d) Only with special prefixes

Answer: b) No, ESP cannot be used as an index register

Question 170: What does the LEA instruction do?

- a) Loads the effective address into a register
- b) Loads the actual value from memory
- c) Leaves the accumulator
- d) Links external addresses

Answer: a) Loads the effective address into a register

Questions on Procedures and the Stack

Question 171: What instruction is used to call a procedure?

- a) JMP
- b) CALL
- c) PROC
- d) INVOKE

Answer: b) CALL

Question 172: What does the CALL instruction do?

- a) Jumps to a procedure

- b) Pushes the return address and jumps to the procedure
- c) Returns from a procedure
- d) Calls an interrupt

Answer: b) Pushes the return address and jumps to the procedure

---

Question 173: Which instruction returns from a procedure?

- a) RETURN
- b) RET
- c) ENDP
- d) EXIT

Answer: b) RET

Question 174: What does the RET instruction do?

- a) Jumps to a label
- b) Pops the return address from the stack and jumps to it
- c) Returns a value
- d) Resets the program

Answer: b) Pops the return address from the stack and jumps to it

---

Question 175: What instruction pushes a value onto the stack?

- a) STORE
- b) PUT
- c) PUSH
- d) SAVE

Answer: c) PUSH

Question 176: What instruction removes (pops) a value from the stack?

- a) REMOVE
- b) GET
- c) POP
- d) RESTORE

Answer: c) POP

Question 177: What happens to ESP when you PUSH a value?

- a) ESP increases
- b) ESP decreases
- c) ESP remains unchanged
- d) ESP is cleared

Answer: b) ESP decreases (stack grows downward)

Question 178: What happens to ESP when you POP a value?

- a) ESP increases
- b) ESP decreases
- c) ESP remains unchanged
- d) ESP is set to 0

Answer: a) ESP increases

Question 179: In a procedure, what is a stack frame?

- a) The size of the stack
- b) The area on the stack reserved for local variables and parameters
- c) The first element on the stack
- d) The last element on the stack

Answer: b) The area on the stack reserved for local variables and parameters

---

Question 180: Which register is commonly used as a base pointer for accessing stack frames?

- a) ESP
- b) EBP
- c) ESI
- d) EDI

Answer: b) EBP

Miscellaneous Questions

Question 181: What does the NOP instruction do?

- a) Stops the program
- b) No operation (does nothing)
- c) Negates the operand
- d) Normalizes a value

Answer: b) No operation (does nothing)

Question 182: Why might you use NOP instructions?

- a) For timing purposes or code alignment
- b) To fill space
- c) For debugging
- d) All of the above

Answer: d) All of the above

Question 183: What is the opcode for NOP?

- a) 00h
- b) 90h
- c) FFh
- d) C3h

Answer: b) 90h

Question 184: Which instruction compares two operands by subtracting them without storing the result?

- a) SUB
- b) TEST
- c) CMP
- d) AND

Answer: c) CMP

Question 185: After a CMP instruction, which flags are typically affected?

- a) Only Zero flag
- b) Only Carry flag
- c) Zero, Carry, Sign, Overflow, Parity
- d) No flags are affected

Answer: c) Zero, Carry, Sign, Overflow, Parity

Question 186: What does the TEST instruction do?

- a) Tests if a register is zero
- b) Performs a logical AND and sets flags without storing the result
- c) Tests the condition codes
- d) Runs a test suite

Answer: b) Performs a logical AND and sets flags without storing the result

---

Question 187: Which instruction performs a logical AND operation?

- a) AND
- b) LAND
- c) ANDB
- d) LOGICAL_AND

Answer: a) AND

Question 188: What does `and eax, 0FFh` do?

- a) Sets EAX to 0FFh
- b) Clears all bits in EAX except the lowest 8 bits
- c) Inverts all bits in EAX
- d) Adds 0FFh to EAX

Answer: b) Clears all bits in EAX except the lowest 8 bits

Question 189: Which instruction performs a logical OR operation?

- a) OR
- b) LOR
- c) ORB

- d) LOGICAL_OR

Answer: a) OR

Question 190: What does `xor eax, eax` do?

- a) Doubles EAX
- b) Clears EAX to zero
- c) Inverts all bits in EAX
- d) Does nothing

Answer: b) Clears EAX to zero

Question 191: Why is `xor reg, reg` commonly used to zero a register?

- a) It's the only way
- b) It's faster and uses fewer bytes than `mov reg, 0`
- c) It sets special flags
- d) It works on all processors

Answer: b) It's faster and uses fewer bytes than `mov reg, 0`

Question 192: Which instruction performs a logical NOT (one's complement)?

- a) NEG
- b) NOT
- c) INVERT
- d) COMPLEMENT

Answer: b) NOT

Question 193: What is the difference between NOT and NEG?

- a) NOT is logical inversion, NEG is two's complement negation
- b) NOT is for unsigned, NEG is for signed
- c) There is no difference
- d) NOT works on bytes, NEG on words

Answer: a) NOT is logical inversion, NEG is two's complement negation

---

Question 194: Which instruction shifts bits to the left?

- a) SHL or SAL
- b) SHR
- c) ROL
- d) ROR

Answer: a) SHL or SAL

Question 195: What happens to the rightmost bit when you execute SHL?

- a) It's preserved
- b) It's filled with 0
- c) It's filled with 1
- d) It's filled with the sign bit

Answer: b) It's filled with 0

Question 196: What is the effect of `shl eax, 1` on the value in EAX?

- a) Divides by 2
- b) Multiplies by 2
- c) Adds 1
- d) No effect

Answer: b) Multiplies by 2

Question 197: Which instruction shifts bits to the right, filling with zeros?

- a) SHL
- b) SHR (logical shift right)
- c) SAR
- d) ROR

Answer: b) SHR (logical shift right)

Question 198: What is SAR (Shift Arithmetic Right)?

- a) A logical shift right
- b) A shift right that preserves the sign bit

- c) A rotation operation
- d) A swap operation

Answer: b) A shift right that preserves the sign bit

Question 199: Which instructions rotate bits without going through the carry flag?

- a) RCL and RCR
- b) SHL and SHR
- c) ROL and ROR
- d) SAL and SAR

Answer: c) ROL and ROR

Question 200: What is the difference between ROL and RCL?

- a) ROL rotates through carry, RCL doesn't
- b) RCL rotates through carry, ROL doesn't
- c) They are the same
- d) ROL is for bytes, RCL for words

Answer: b) RCL rotates through carry, ROL doesn't

END OF QUESTIONS

Summary

Total Questions: 200

Coverage:

- **Chapter 2 (Questions 1-75)**: x86 Processor Architecture
  - General Concepts (Q1-10)
  - 32-Bit x86 Processors (Q11-20)
  - 64-Bit x86-64 Processors (Q21-25)
  - Components of an x86 Microcomputer (Q26-35)
  - Input-Output System (Q36-45)
  - x86 Memory Management (Q46-55)
  - x86 Registers (Q56-75)

- **Chapter 3 (Questions 76-200)**: Assembly Language Fundamentals
    - Basic Elements (Q76-90)
    - Example: Adding Three Integers (Q91-100)
    - Assembling, Linking, Running (Q101-110)
    - Defining Data (Q111-125)
    - Symbolic Constants (Q126-130)
    - Real-Address Mode Programming (Q131-140)
    - Protected Mode Programming (Q141-150)
    - Data Transfer and Arithmetic (Q151-160)
    - Addressing Modes (Q161-170)
    - Procedures and Stack (Q171-180)
    - Miscellaneous (Q181-200)

Good luck with your CBT exam!