ZENITH STUDY HUB / ENGINEERING NOTES

8085 Microprocessor

Unit I • Architecture, Buses, Interfacing and Assembly Programming

THE LEARNING PROMISE

Understand the hardware, trace each register and memory operation, and write complete answers for the SPPU End-Semester Examination. Uses the official 2024 Pattern syllabus and the college Unit I question bank.

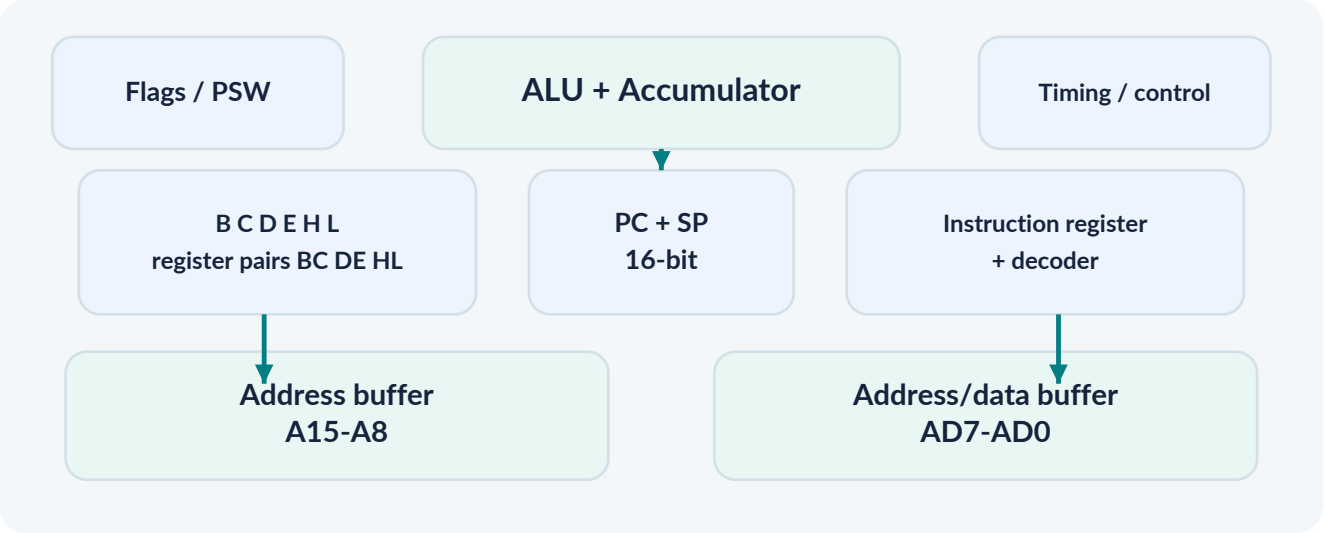


Figure 1. *Original functional architecture overview. Figure is conceptual, not a pin-level Intel drawing.*

Course: PCC302-ROA | Unit I | 08 teaching hours | ESE: 70 marks.
Unit-wise 14-mark distribution: study-planning assumption, not syllabus-guaranteed.

Edition: Premium Exam-Ready v3 review, 22 November 2026.
Academic review candidate. Not an official university publication.

How to study and write better exam answers

This guide is organized in the **same order as the official syllabus**, with extra practice drawn from the department question bank. Do not skip a concept because it has not appeared in a previous paper. Learn the idea first; then practice the structured answer.

Syllabus requirement	Where to study
8085 basic architecture	Sections 1–3
Address, data and control buses	Section 4
Memory and I/O interfacing basics	Sections 5–6
Instruction-set overview	Sections 7–8
8085 assembly programs	Sections 9–11
Exam answers and revision	Sections 12–14

EXAM ANSWER FORMULA
Definition → neat labelled diagram → working/steps → key points and examples → short conclusion. For assembly answers: **question / algorithm → code → dry run → result**. Adapt the detail to marks and instructions; exact question lengths are not guaranteed.

Important: The official syllabus specifies a 70-mark end-semester theory paper. An equal 14-mark allocation per unit is only a planning assumption unless a separate authorized examination scheme confirms it.

1 Fundamentals of the Intel 8085

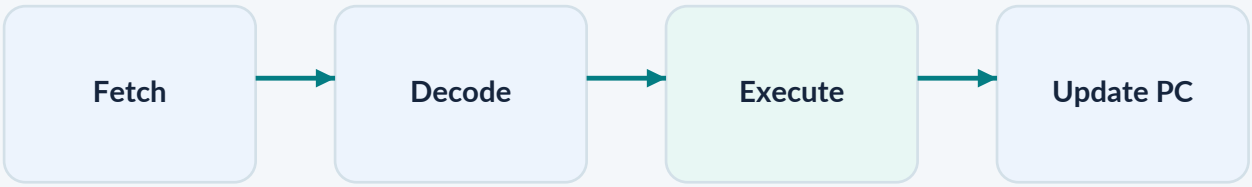
1.1 What is a microprocessor?

A **microprocessor** is a programmable integrated circuit containing a central processing unit (CPU). It fetches instructions from memory, decodes them, performs arithmetic/logic operations and controls external devices through electrical buses. The Intel 8085 is an **8-bit microprocessor** designed for general stored-program computation.

IN SIMPLE WORDS
Think of the CPU as the decision maker, memory as its notebook, and input/output interfaces as the parts that connect it to the outside world. The CPU reads its instructions from memory, works on the data, and sends the result back.

- **8-bit data path:** The accumulator and ALU normally operate on 8-bit values; one byte is 8 bits.
- **16-bit address space:** With 16 address bits, there are 2^{16} = 65,536 byte addresses (0000H–FFFFH), i.e. 64 KiB.
- **External memory and I/O:** The 8085 is not a complete microcontroller with all peripherals on one chip.
- **Power and timing:** The 8085 requires a properly designed clock, supply, reset and external circuitry; a real system must respect device datasheet specifications.

1.2 The stored-program cycle



The instruction may contain several memory or I/O machine cycles

Figure 2. *Conceptual instruction execution sequence. A single instruction can require several machine cycles.*

The **Program Counter (PC)** identifies the next opcode byte. The processor fetches that opcode, decodes it, accesses any operand bytes, performs the action, updates flags if the instruction specifies, and moves on. Normally the PC increments past instruction bytes; jumps, calls and returns may change it.

Example: If memory 2000H contains 3EH and 2001H contains 25H, those bytes encode **MVI A,25H**. The accumulator becomes 25H, and the next sequential instruction begins at address 2002H.

2 Internal architecture and registers

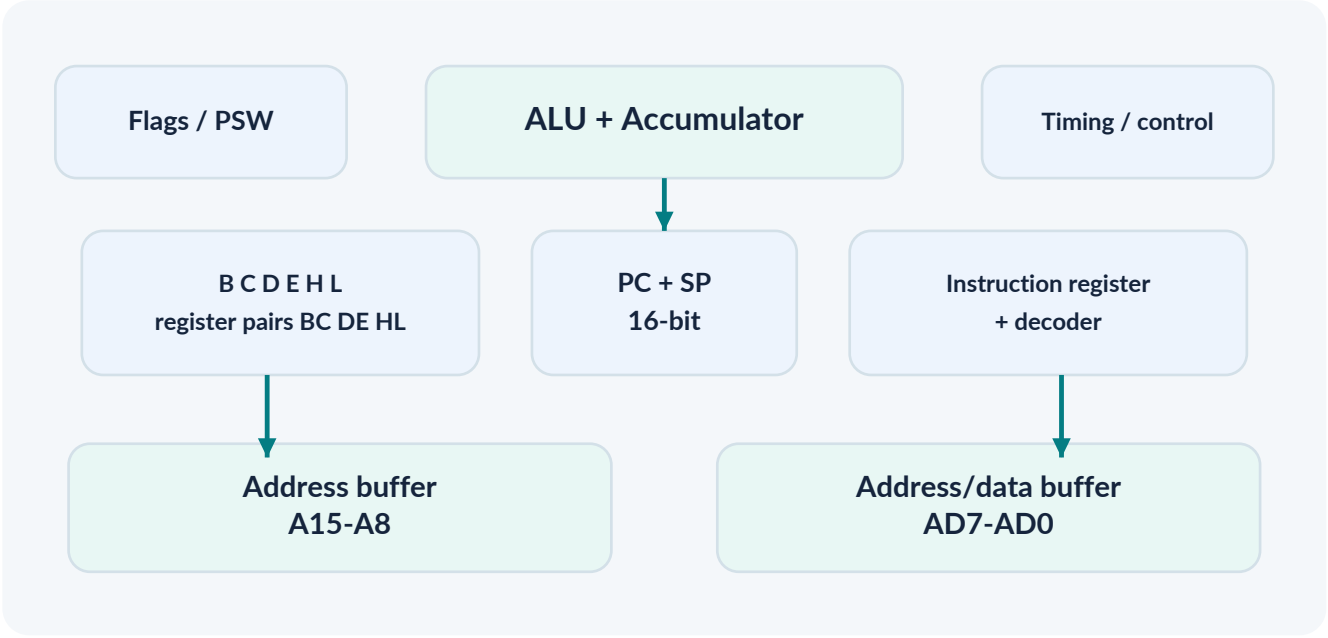


Figure 3. *Simplified 8085 CPU block diagram. Label ALU, accumulator, flags, register pairs, PC, SP, instruction decoder, timing/control and bus buffers.*

Block / register	Function
ALU + Accumulator A	8-bit arithmetic, logic and temporary result storage.
General registers B,C,D,E,H,L	Six 8-bit registers; BC, DE and HL can be used as 16-bit pairs. HL often points to memory.
Program Counter (PC)	16-bit address of the next instruction byte.
Stack Pointer (SP)	16-bit pointer to top of descending memory stack.

Block / register	Function
Instruction register / decoder	Decodes the opcode and controls internal operation.
Timing / control unit	Coordinates machine cycles, ALE, read/write and status signals.
Address / data buffers	Connect internal operations to the external address/data buses.

2.1 Flag register and condition codes



Figure 4. *Five 8085 condition flags. Other bits in the eight-bit flag register are not general-purpose arithmetic flags.*

Flag	What setting the flag means
S — Sign	Bit 7 of an 8-bit result is 1.
Z — Zero	The relevant result equals 00H.
AC — Auxiliary carry	Carry from bit 3 to bit 4 in certain arithmetic; important for DAA.
P — Parity	The result contains an even number of set bits.
CY — Carry	Unsigned carry out during addition or borrow during subtraction, where specified.

Quick example: Adding F8H and 15H produces 10DH. The accumulator retains the low 8 bits **0DH**; CY becomes 1. The result is nonzero, so Z is 0. Some instructions do not modify all flags: for example, **DCR does not change CY**.

EXAM MEMORY CUE
S Z AC P CY — Sign, Zero, Auxiliary Carry, Parity, Carry. For an answer, define each, mention when it changes, and give one short numerical or binary example.

3 Instruction cycles and timing

3.1 T-state, machine cycle and instruction cycle

Term	Definition	Example
T-state	One processor clock interval.	Single timing step
Machine cycle	One distinct external bus operation.	Opcode fetch or memory read
Instruction cycle	All machine cycles required to execute one instruction.	Fetch opcode + read operands

For **LXI H,2050H**, the opcode is followed by two immediate bytes. On the 8085, the low byte precedes the high byte: the sequence stores 50H then 20H. After execution HL = 2050H and PC advances by three bytes (unless interrupted or otherwise redirected).

3.2 The ALE signal

The eight shared **AD7–AD0** pins carry low address bits during the first stage of a bus cycle (T1) and data later. Address Latch Enable (**ALE**) signals an external latch to preserve A7–A0 so memory selection remains valid while AD pins switch to data transfer.

DO NOT CONFUSE
A **T-state** is not a complete instruction. Nor do all instructions have the same number of T-states. A faithful timing-diagram answer depends on the specified instruction and machine cycle.

4 Address, data and control buses



Figure 5. *Original conceptual bus diagram: the 8085 multiplexes AD0–AD7 and uses ALE plus an external latch. Not a pin-by-pin schematic.*

Bus	Width and direction	Role
Address	16 bits, outward from CPU	Select memory/I/O location. High byte A15–A8 dedicated; low A7–A0 multiplexed on AD7–AD0.
Data	8 bits, bidirectional	Carry instruction/data bytes to or from memory and devices.
Control and status	Multiple electrical signals	$\overline{RD}$ and $\overline{WR}$ specify reads/writes; IO/M identifies I/O vs memory; ALE and status lines coordinate access.

4.1 Step-by-step read transaction

- The CPU presents an address: A15–A8 directly and A7–A0 on AD7–AD0.
- ALE enables an external latch to retain A7–A0 when the shared pins change role.
- Higher address and status signals drive decoder/chip-select logic for a single target.
- The CPU asserts the appropriate active-low read signal ($\overline{RD}$); the selected device drives the data bus.
- The CPU samples the byte and completes the bus cycle according to timing requirements.

Why multiplex? Sharing eight package pins between address and data reduces the number of dedicated pins. However, an external latch is needed to hold the address while data is transferred.

5 Basic memory interfacing

Memory interfacing is the process of connecting external ROM/RAM to the 8085 using **address lines**, **data lines**, read/write controls and **address decoding**. The decoder ensures that only the intended device responds to a particular memory address.

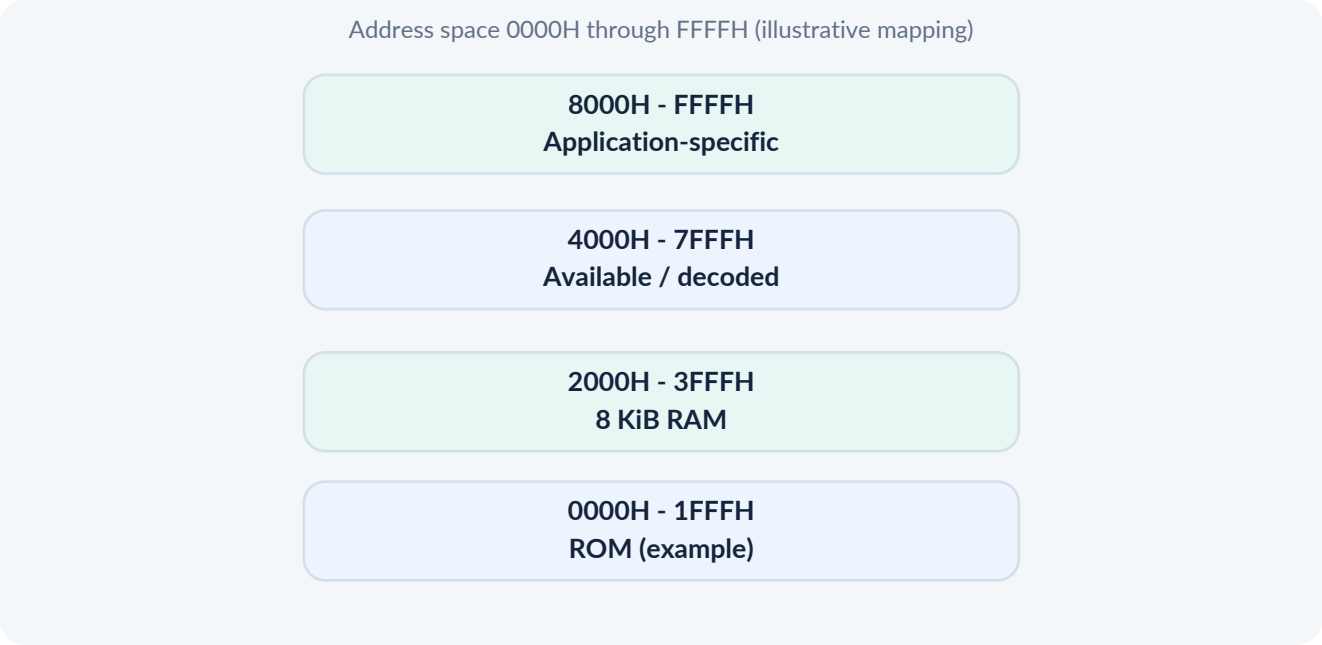


Figure 6. *Illustrative 64 KiB external address map. The actual RAM and ROM arrangement is chosen by the hardware designer.*

5.1 Worked address-decoding example

Connect an **8 KiB RAM** to addresses 2000H through 3FFFH. Since 8 KiB = 8 × 1024 = 8192 = 2^{13} bytes, the chip needs **13 address inputs A0–A12**. The remaining high address bits must be decoded appropriately so this chip is selected only within its 8 KiB region.

VERIFY THE RANGE
3FFFH − 2000H + 1 = 2000H bytes = 8192 decimal bytes = 8 KiB. The “+1” includes both the first and last address.

A memory read requires the correct chip select and read signals; a write requires chip select and write control. ROM is commonly used for persistent program storage; RAM holds changing data, stack and temporary values.

5.2 Typical exam points

- Define the memory map and total 64 KiB byte address space.
- Calculate address lines as log2(number of locations) when chip capacity is a power of two.
- Demultiplex A7–A0, connect data pins, and decode high address lines for chip select.
- Check that ROM/RAM ranges do not overlap unless intentional and justified.
- Respect active-low control, bus direction, and electrical compatibility.

6 I/O interfacing: isolated versus memory-mapped

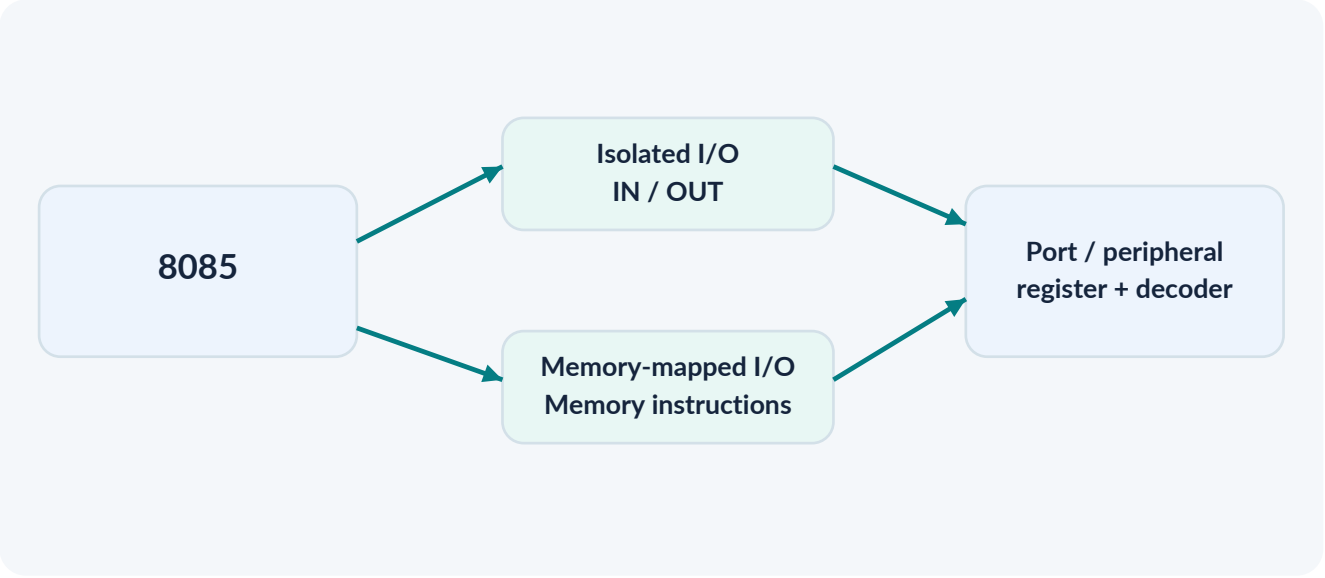


Figure 7. *Two conceptual interfacing choices. External latches, buffers, address decoders and drivers depend on the actual circuit.*

Property	Isolated I/O	Memory-mapped I/O
Address space	IN/OUT specify 8-bit port numbers, up to 256 ports.	Device occupies part of 64 KiB memory map.
Instructions	IN port, OUT port.	Memory-access instructions as supported by mapping.
Decoder	Port selection plus I/O status.	Memory address decoding plus control.
Advantage	Keeps port space separate from memory addresses.	Can use memory-type instructions for device registers.

Example: `MVI A,55H`, followed by `OUT 02H`, places 55H on the data bus during a write to port 02H, assuming the external circuitry decodes this port. A corresponding input uses `IN 01H` to load the byte supplied by port 01H into accumulator A.

ROBOTICS SAFETY
8085 bus pins cannot directly power motors or industrial actuators. Use suitable driver electronics, protection and interface circuits. The sample port addresses describe logic, not a complete hardware design.

7 Instruction-set overview

The 8085 instruction set tells the CPU how to transfer, transform and control data. Learn the function and a valid example for each group rather than memorizing an unexplained mnemonic list.

Category	Instructions	Purpose / example
Data transfer	MOV, MVI, LXI, LDA, STA, LHLD, SHLD	MOV A,B copies register B into A.
Arithmetic	ADD, ADC, SUB, SBB, INR, DCR, DAD, DAA	ADD B computes A = A+B.
Logic / comparison	ANA, ORA, XRA, CMP, CMA	CMP B changes flags as if A−B, without changing A.
Branching	JMP, JZ, JNZ, JC, JNC, CALL, RET	JNZ LOOP repeats until Z is set.
Stack and machine	PUSH, POP, EI, DI, HLT, NOP	Control execution or stack operations.

7.1 Length of an instruction

Length	Example	Stored bytes
1 byte	MOV A,B	Only opcode.
2 bytes	MVI A,25H	Opcode, immediate 25H.
3 bytes	LXI H,2050H	Opcode, 50H, then 20H (low byte first).

7.2 Addressing modes

Mode	Meaning	Example
Register	Operand in a CPU register.	MOV A,C
Immediate	Operand value follows opcode.	MVI B,05H
Direct	16-bit effective memory address encoded.	LDA 2050H
Register indirect	Address held in a register pair.	MOV A,M (HL points to byte)
Implicit	Operand implied by opcode.	CMA

WHY CMP IS USEFUL
CMP M compares A with the byte at address HL by setting flags as for a subtraction, but it does not replace the value of A. In unsigned comparisons, **JNC** indicates A ≥ compared byte, and **JC** indicates A is smaller.

8 Hexadecimal and trace fundamentals

Hex digits range 0–9 and A–F. A byte can be shown as two hexadecimal digits. To trace any instruction, write its starting registers and memory bytes, execute it according to the ISA, update only the flags specified for that instruction, and record the resulting registers/memory.

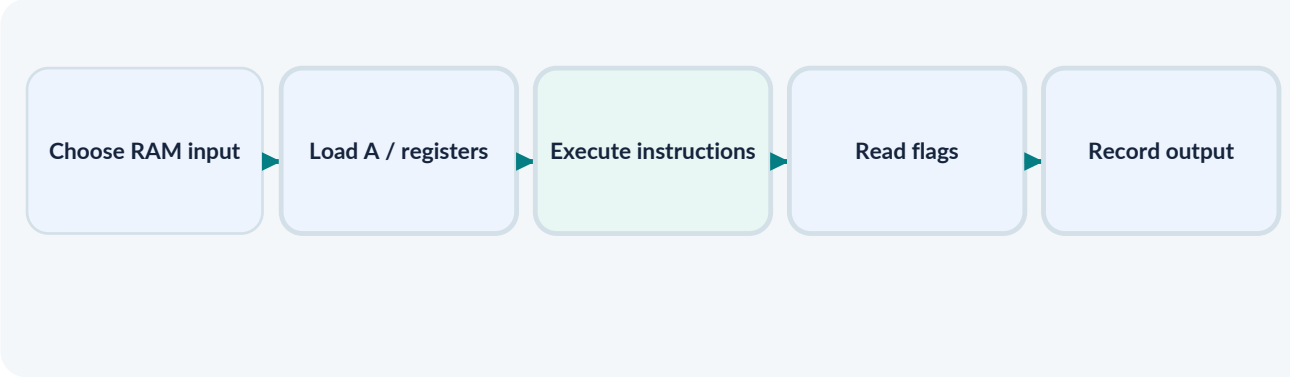


Figure 8. *Original exam dry-run workflow. Trace the actual instruction sequence, not only the final arithmetic value.*

Operation	Working	Expected result
F8H + 15H	0xF8 + 0x15 = 0x10D	A = 0DH, CY = 1
05H parity	Binary 00000101 has two 1 bits.	P = 1 (even parity)
35H CMP 42H	A=35H < 42H as unsigned bytes.	CY=1; JC will branch
DCR C from 03H	03H→02H→01H→00H	Z=1 at zero; CY unchanged

9 Core assembly programs

The following are original programs designed for learning. Unless a problem states otherwise, assume the code executes from separate program memory and the listed data addresses contain valid RAM. Addresses and sample values are illustrative.

Program 1 — Add two 8-bit numbers and save carry

Question-bank objective: Add memory bytes at 2050H and 2051H. Store the result at 2052H and the carry at 2053H.

```
LDA 2050H      ; Load first byte into A
MOV B,A        ; Save first byte in B
LDA 2051H      ; Load second byte
ADD B          ; A = low sum; CY = carry
STA 2052H      ; Save low 8 bits
MVI A,00H      ; MVI does not alter CY
ADC A          ; Save old CY as 00H/01H
STA 2053H      ; Store carry byte
HLT
```

Dry run: F8H + 15H = 10DH. Therefore [2052H] = **0DH** and [2053H] = **01H**. If the inputs are 25H and 13H, then [2052H] = 38H and carry byte = 00H. **Technical note:** MVI and STA do not modify carry; ADC A starts with A=0 and converts the carry flag to an ordinary byte.

Program 2 — Subtract two 8-bit numbers (new question-bank coverage)

Objective: Compute [2050H] − [2051H], store the 8-bit difference at 2052H and, for unsigned subtraction, preserve borrow status at 2053H.

```
LDA 2050H      ; First unsigned byte
MOV B,A        ; Keep first operand
LDA 2051H      ; Second unsigned byte
MOV C,A        ; Keep second operand
MOV A,B        ; A = first operand
SUB C          ; A = first - second
STA 2052H      ; Store modulo-256 result
MVI A,00H      ; Preserve CY from SUB
ADC A          ; A = 01H if borrow, else 00H
STA 2053H      ; Store borrow indicator
HLT
```

Trace A: 35H − 12H = 23H; CY = 0, so memory contains [2052H]=23H, [2053H]=00H. **Trace B:** 12H − 35H = −23H (−35 decimal), represented in eight bits as DDH; CY = 1, so [2052H]=DDH, [2053H]=01H. The borrow indicator is separate from the wrapped eight-bit result.

Program 3 — Largest unsigned byte in two locations (new question-bank coverage)

```
LDA 2050H      ; First unsigned number
MOV B,A        ; Save first number
LDA 2051H      ; A = second number
CMP B          ; Compare second with first
JNC SECOND     ; If second >= first, keep A
MOV A,B        ; Otherwise first is larger
SECOND: STA 2052H
HLT
```

Why it works: CMP B sets carry when A is below B as an unsigned value. JNC therefore preserves A when second ≥ first; otherwise MOV A,B makes A the first. For [2050H]=9AH and [2051H]=37H, [2052H] = **9AH**. For 12H and F0H, result = F0H.

Program 4 — Transfer a block of five bytes

```
LXI H,2050H    ; Source pointer
LXI D,3050H    ; Destination pointer
MVI C,05H      ; Count five bytes
COPY: MOV A,M  ; Read source byte
STAX D         ; Store byte via DE
INX H          ; Move source pointer
INX D          ; Move destination pointer
DCR C          ; Decrement count
JNZ COPY       ; Repeat until zero
HLT
```

Dry run: 12H, 34H, 56H, 78H, 9AH at locations 2050H–2054H appear in the same order at 3050H–3054H. These blocks **do not overlap**; an overlapping block copy needs a different direction-aware design.

Program 5 — Largest unsigned number among five bytes

```
LXI H,2050H    ; First array element
MOV A,M        ; Current largest value
MVI C,04H      ; Four elements remain
NEXT: INX H    ; Next element
CMP M          ; Compare A with [HL]
JNC KEEP       ; Keep A if A >= [HL]
MOV A,M        ; New maximum
KEEP: DCR C
JNZ NEXT
STA 3050H      ; Output max
HLT
```

Dry run: [12H, 99H, 3AH, 85H, 22H] results in **99H** stored at 3050H. Do not confuse unsigned comparison with signed byte ordering.

10 Additional high-yield programming

Program 6 — Count even and odd array bytes (new question-bank coverage)

Objective: Inspect five bytes beginning at 2050H. Save the number of even values at 3050H and odd values at 3051H. Any byte is even if its least significant bit is zero; use ANI 01H to test that bit.

```
LXI H,2050H    ; Input array
MVI B,05H      ; Five elements
MVI D,00H      ; Even count
MVI E,00H      ; Odd count
LOOP: MOV A,M  ; Load next byte
ANI 01H        ; Keep least-significant bit
JNZ ODD        ; If result 01H -> odd
INR D          ; Increment even count
JMP NEXT
ODD:  INR E     ; Increment odd count
NEXT: INX H    ; Next array address
DCR B          ; One fewer input byte
JNZ LOOP
MOV A,D        ; Move even count to A
STA 3050H
MOV A,E        ; Move odd count to A
STA 3051H
HLT
```

Dry run: [02H, 07H, 10H, 15H, FFH] has two even bytes (02H,10H) and three odd bytes (07H,15H,FFH). Therefore [3050H]=**02H**, [3051H]=**03H**. This program assumes a nonzero fixed count of five. For a zero-length array, modify the termination logic before using it.

Program 7 — Sum five unsigned bytes, low-byte result

```
LXI H,2050H     ; First array byte
MVI C,05H       ; Five bytes to sum
MVI A,00H       ; Clear sum
ADDLOOP: ADD M  ; Add next element to A
INX H           ; Advance pointer
DCR C           ; Decrement count
JNZ ADDLOOP
STA 3050H       ; Store low 8-bit sum
HLT
```

Dry run: [01H,02H,03H,04H,05H] gives 0FH. This routine stores only the low 8 bits; if total exceeds FFH, a separate 16-bit accumulation method is required. The final CY flag alone cannot reliably encode all earlier carries in a multi-byte loop.

Program 8 — Add two 16-bit numbers

```
LHLD 2050H      ; HL = 16-bit first number
XCHG             ; DE = first number
LHLD 2052H      ; HL = second number
DAD D            ; HL = HL + DE, sets CY
SHLD 3050H      ; Low byte first in RAM
HLT
```

Trace: If the first word is 1234H and second is 0112H, final HL = 1346H; [3050H] = 46H and [3051H] = 13H. **DAD** sets carry on 16-bit overflow but the code does not store the seventeenth carry bit.

Program 9 — Packed BCD addition

```
MVI A,45H       ; Packed BCD decimal 45
ADI 38H         ; Add packed BCD 38
DAA             ; Decimal adjust A
STA 3050H       ; Store packed BCD 83
HLT
```

Intermediate binary addition yields 7DH. **DAA** adjusts this to 83H, representing decimal 83. It is intended for suitable packed-BCD arithmetic and depends on previous arithmetic state; it is not a universal binary-to-BCD converter.

11 Six-step method to trace 8085 assembly

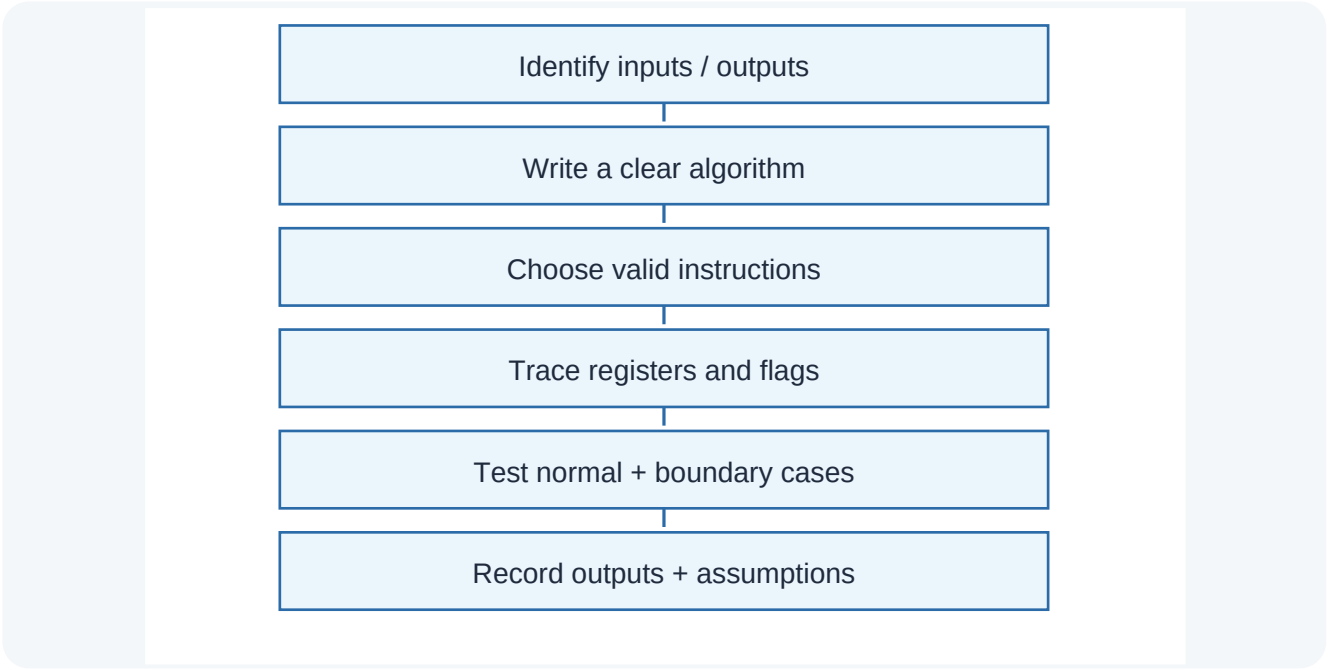


Figure 9. *Original sequence for solving exam assembly questions. Show input and output addresses explicitly.*

Step 1: Identify input data type (unsigned/signed/BCD), memory addresses and expected output. **Step 2:** Write an algorithm in words. **Step 3:** Choose instructions supported by the 8085. **Step 4:** Annotate registers and flag changes. **Step 5:** Dry-run on ordinary and boundary test values. **Step 6:** Report RAM contents and state assumptions.

Program	Test input	Expected output
Add	F8H, 15H	Low 0DH, carry 01H
Subtract	12H, 35H	Low DDH, borrow 01H
Max of two	9AH, 37H	9AH
Copy five	12H,34H,56H,78H,9AH	Same five bytes at destination
Even / odd	02H,07H,10H,15H,FFH	Even 02H, odd 03H
Max five	12H,99H,3AH,85H,22H	99H

12 Syllabus-aligned exam answers

Q1. Define a microprocessor and explain the architecture of the Intel 8085 with a neat diagram.

An 8085 is an **8-bit stored-program microprocessor** which fetches instructions from external memory, decodes them, performs arithmetic or logical operations and controls memory/I/O transfers. Draw **Figure 3** and label: (i) ALU and accumulator, (ii) flag register, (iii) B,C,D,E,H,L register bank, (iv) 16-bit PC and SP, (v) instruction register and decoder, (vi) timing/control logic, and (vii) address and address/data buffers.

Working: PC supplies the next instruction address; the CPU fetches and decodes the opcode; registers/ALU operate on data; flags describe the specified result; the bus interface accesses external memory or peripherals. It uses a 16-bit address space and an 8-bit external data bus. **Conclusion:** the register, ALU and control sections coordinate to execute a stored program.

Q2. Differentiate address, data and control buses of the 8085.

An **address bus** selects one of 65,536 locations using 16 bits; the upper eight bits A15–A8 are dedicated and the low bits are multiplexed on AD7–AD0. An **8-bit bidirectional data bus** transfers a byte between CPU and selected memory/device. **Control signals** coordinate the access: ALE captures the low address, $\overline{RD}$ initiates a read, $\overline{WR}$ initiates a write, and IO/M distinguishes memory from I/O cycles. Draw **Figure 5** to show the external address latch.

Q3. Explain memory interfacing with an 8 KiB RAM example.

Memory interfacing connects RAM/ROM using 16-bit CPU addresses, 8-bit data lines, active-low read/write control and a decoder or chip-select. For an 8 KiB chip mapped to 2000H–3FFFH: 2^{13} = 8192 positions, so A0–A12 drive chip address pins; high-order address lines select the block. 3FFFH − 2000H + 1 = 8192. Show a labelled memory map and the CPU-to-RAM control signals. Keep I/O and memory decode conventions distinct.

Q4. Compare memory-mapped and isolated I/O.

In **isolated I/O**, IN and OUT use an 8-bit port number in a separate I/O space. In **memory-mapped I/O**, device registers occupy memory addresses, sharing the 64 KiB space and typically using memory-access instructions. Both need suitable external hardware selection, data transfer and protection. Explain with a simple port or sensor interface example and clearly label the control signals.

Q5. Classify the instruction set and illustrate five addressing modes.

Five useful instruction categories are data transfer, arithmetic, logical/compare, branch, and stack/machine-control. Examples: MOV, ADD, CMP, JNZ and PUSH. Addressing modes: register (MOV A,C), immediate (MVI B,05H), direct (LDA 2050H), register indirect (MOV A,M through HL), implicit (CMA). State whether the operand is a literal value, CPU register or memory address.

Q6. Write programs for addition, subtraction, block transfer, largest of two numbers, and even/odd counts.

Use the exact annotated programs in **Sections 9 and 10**. Write the algorithm, register use, code, at least one dry-run table and final memory locations. In subtraction, the borrow must be explained separately from the modulo-256 result. In parity counting, check bit 0 via ANI 01H, count in independent registers, and define where the counters are stored.

13 Fast revision and common errors

Remember	Key fact or correction
Address range	0000H–FFFFH; 65,536 byte locations (64 KiB).
Data path	8-bit accumulator/ALU/data bus, 16-bit PC/SP/addressing.
Multiplexing	AD7–AD0 carry address during T1 and data afterward; ALE controls latch.
Comparison	CMP sets flags without modifying A.
Carry	ADD sets CY on unsigned carry; SUB sets CY on unsigned borrow.
Loops	DCR changes Z but does not modify CY.
Indirect memory	M refers to memory at the address in HL.
16-bit immediate	LXI stores low immediate byte before high byte.
Count parity	Use ANI 01H; result 0 is even, 1 is odd.
Memory vs I/O	Separate isolated 8-bit port numbers from 16-bit memory map.

LAST-MINUTE LEARNING CHECK

Can you redraw the CPU and multiplexed buses without seeing the answer? Explain the difference between PC and SP? Trace at least one carry, one borrow, one CMP branch and one loop? Explain the difference between an actual byte and an address? If yes, you can build a structured exam response.

14 Source provenance and verification limits

Official scope: Savitribai Phule Pune University, TE Robotics and Automation, 2024 Pattern (effective AY 2026–27), course PCC302-ROA, Unit I, syllabus printed p. 16; 70-mark end-semester written examination based on five units.

Primary architecture and programming reference: Intel, *8080/8085 Assembly Language Programming Manual* (1981), retrieved from the user's authorized Drive folder MM/Books and inspected for instruction mnemonics and 8085 usage. Intel's original architecture and instruction descriptions take precedence over informal slides where they differ.

Question practice: College *unit 1 and 2 Question Bank.docx*, TE Robotics and Automation. This edition now includes subtract-two, largest-of-two and even/odd-count routines that were not provided as complete programs in the earlier 19-page Zenith notes. All test values are original teaching examples.

Content input: Zenith *Zenith_MM_Unit_01_8085_Complete_Notes.pdf*, 19 pages; the official *TE Syllabus.pdf*; college *MM Unit-1.pdf* lecture resource. The prior document remains preserved. No copyrighted textbook figures or scanned pages were copied.

Diagrams: Figures 1–9 are original vector educational illustrations (not Intel schematic pinouts). Real hardware use requires datasheet pin numbers, electrical checks, address decoding and driver circuits beyond this theory guide.

PUBLICATION REVIEW LIMIT

This is a new **review candidate**, not an instructor-approved or hardware-validated final edition. Arithmetic and sample traces have been checked, but the assembly listings should also be run in a native 8085 assembler/emulator before any final public replacement. This file has not replaced a Drive copy or gone live.

Exam answers and verified bus connections

Q. Why are address, data and control buses logically distinct?

Address lines identify a memory location, bidirectional data lines carry bytes, and control signals define read/write operations and timing. In the 8085, AD7-AD0 are multiplexed: they carry A7-A0 during T1, then carry data. ALE controls an external latch to retain A7-A0.

Q. Why is correct memory interfacing essential?

An address decoder selects the intended memory device; the latch keeps the low address stable after T1, and control logic enables valid read and write transactions. Incorrect selection can cause bus contention or accesses to the wrong device.

Conceptual 8085 to 8 KiB RAM connection (2000H–3FFFH)



SIGNAL	CONNECTION	FUNCTION
AD7-AD0 during T1	8085 -> latch	Low address A0-A7 retained after ALE
AD7-AD0 during data	8085 <-> RAM	Bidirectional D0-D7 data bus
A8-A12	8085 -> RAM	Upper five of the RAM address inputs
A15-A13	8085 -> decoder	Decode 001 to select 2000H-3FFFH
ALE, RD, WR	8085 -> interface	Latch timing and gated read/write controls

Exam answer: Memory-mapped versus isolated I/O

Isolated I/O uses IN and OUT with 8-bit port addresses in a separate I/O space. Memory-mapped I/O uses locations inside the 64 KiB memory space and appropriate memory instructions. Both require address selection, bus-compatible logic and output drivers.

Flag and notation reminder

2^16 = 65,536 bytes (64 KiB); 2^13 = 8,192 bytes (8 KiB). Flag bits: S=7, Z=6, AC=4, P=2, CY=0. The figure is a conceptual educational interconnect, not a pin-level schematic or a timing diagram.

Program QA and release criteria

Program semantics and validity

Nine original listings remain in source pages 13-17. The review checks ordinary and boundary arithmetic, unsigned CMP/JNC behavior, loop termination, carry/borrow storage, 16-bit low-byte-first results, packed-BCD adjustment, and preservation of unrelated memory. The independent Python checker is an educational instruction-level model, not a verified native assembler, silicon timing simulator or hardware test.

Mandatory release blockers still open

1. Assemble all nine listings with a native 8085-compatible assembler, resolving syntax as necessary.

2. Execute each assembled program using a native 8085 emulator and compare final registers, flags and RAM.

3. Independently inspect all nine figures and every page at 100% and mobile reading scale.

4. Inspect PDF accessibility (tagging, reading order, alternate descriptions and bookmarks).

5. Proofread technical claims and confirm examination scheme provenance before approval.

Preserved pages and revision policy

Pages 1-22 preserve the original v2 content and vector illustrations except the corrected examination statements on pages 1-2 and the replacement six-stage trace diagram on page 18. This appendix adds explanatory material without claiming final academic approval. The original v2 remains the rollback baseline.

Status

REVIEW CANDIDATE ONLY. NOT FOR PUBLICATION. This PDF does not claim native 8085 validation, full accessibility certification, or final page-by-page academic sign-off.