

# MIC - Practical - QB - Solutions - By - Th3\_

## 1. Write ALP to perform addition of two 8 bit numbers

The screenshot shows the emu8086 interface with the following assembly code:

```

01 DATA SEGMENT
02     NUM1 DB 42H
03     NUM2 DB 44H
04     SUM DB ?
05     CARRY DB 00H
DATA ENDS
06
07 CODE SEGMENT
08     ASSUME CS:CODE, DS:DATA
09 START:
10     MOV AX, DATA
11     MOV DS, AX
12
13     MOV AL, NUM1
14     ADD AL, NUM2
15     JNC NEXT
16     INC CARRY
17
18 NEXT:    MOV SUM, AL
19
20     MOV AH, 4CH
21     INT 21H
CODE ENDS
END START

```

Registers and memory dump windows are visible, showing the values of variables and the state of the CPU during execution.

## 2. Write ALP to perform addition of two 16 bit numbers

The screenshot shows the emu8086 interface with the following assembly code:

```

01 DATA SEGMENT
02     NUM1 DW 2014H
03     NUM2 DW 1625H
04     SUM DW ?
05     CARRY DB 00H
DATA ENDS
06
07 CODE SEGMENT
08     ASSUME CS:CODE, DS:DATA
09 START:
10     MOV AX, DATA
11     MOV DS, AX
12
13     MOV AX, NUM1
14     ADD AX, NUM2
15     JNC NEXT
16     INC CARRY
17
18 NEXT:    MOV SUM, AX
19
20     MOV AH, 4CH
21     INT 21H
CODE ENDS
END START

```

Registers and memory dump windows are visible, showing the values of variables and the state of the CPU during execution.

## MIC - Practical - QB - Solutions - By - Th3\_

### 3. Write ALP to perform multiplication of two unsigned 8 bit numbers

The screenshot shows the QBasic IDE interface with three main windows:

- Code Editor:** Displays the assembly code for multiplying two unsigned 8-bit numbers. The code uses the INT 21h interrupt to perform the multiplication.
- Variables Window:** Shows the values of variables: NUM1 = 05h, NUM2 = 06h, and RESULT = 00TEh.
- Registers Window:** Displays the CPU registers. The CX register contains the value 00 23, which is highlighted in blue. The instruction at address F400:0204 is INT 21h.

```

01 DATA SEGMENT
02 NUM1 DB 5
03 NUM2 DB 6
04 RESULT DW ?
05 DATA ENDS
06
07 CODE SEGMENT
08 ASSUME CS:CODE, DS:DATA
09
10 START:
11     MOV AX, DATA
12     MOV DS, AX
13
14     MOV AL, NUM1
15     MUL NUM2
16     MOV RESULT, AX
17
18     MOV AH, 4CH
19     INT 21H
20
21 CODE ENDS
22 END START

```

### 4. Write ALP to perform multiplication of two signed 8 bit numbers

The screenshot shows the QBasic IDE interface with three main windows:

- Code Editor:** Displays the assembly code for multiplying two signed 8-bit numbers using the IMUL instruction.
- Variables Window:** Shows the values of variables: NUM1 = 0FBh, NUM2 = 06h, and RESULT = 0FFE2h.
- Registers Window:** Displays the CPU registers. The CX register contains the value 00 23, which is highlighted in blue. The instruction at address F400:0204 is IMUL NUM2.

```

01 DATA SEGMENT
02 NUM1 DB -5
03 NUM2 DB 6
04 RESULT DW ?
05 DATA ENDS
06
07 CODE SEGMENT
08 ASSUME CS:CODE, DS:DATA
09
10 START:
11     MOV AX, DATA
12     MOV DS, AX
13     MOV AL, NUM1
14     IMUL NUM2
15     MOV RESULT, AX
16
17     MOV AH, 4CH
18     INT 21H
19
20 CODE ENDS
21 END START

```

## MIC - Practical - QB - Solutions - By - Th3\_

### 5. Write an ALP to add series of 10 Numbers

The screenshot shows the QBasic IDE interface with the following assembly code:

```

01 CODE SEGMENT
02 ASSUME CS:CODE
03 START:
04 MOV AX, 2000H
05 MOV DS, AX
06 MOV SI, 4000H
07 MOV CX, 0000H
08 MOV AX, 0000H
09
10 BACK:
11 ADD AX, [SI]
12 JNC SKIP_CARRY
13 INC AH
14 SKIP_CARRY:
15 INC SI
16 LOOP BACK
17
18 MOV [SI], AX
19 CODE ENDS
20 END START

```

Registers and memory dump windows are visible in the background.

### 6. Write an ALP to Find Largest number

The screenshot shows the QBasic IDE interface with the following assembly code:

```

01 CODE SEGMENT
02 ASSUME CS:CODE, DS:DATA
03 START:
04 MOV AX, 4000H
05 MOV DS, AX
06 MOV SI, 2000H
07 MOV CX, 0000H
08 MOV AL, 00H
09
10 BACK:
11 CMP AL, [SI]
12 JNC NEXT
13 MOV AL, [SI]
14 INC SI
15 NEXT: LOOP BACK
16
17 MOV [SI], AL
18 MOV AH, 4CH
19 INT 21H
20
21 CODE ENDS
22 END START
23
24

```

Registers and memory dump windows are visible in the background.

## MIC - Practical - QB - Solutions - By - Th3\_

### 7. Write an ALP to Find Smallest number

The screenshot shows the QBasic IDE with the following assembly code:

```

01 CODE SEGMENT
02 ASSUME CS:CODE, DS:DATA
03
04 START: MOV AX, 4000H
05 MOV DS, AX
06 MOV SI, 2000H
07 MOV CX, 000AH
08 MOV AL, 0FFH
09
10 BACK: CMP AL, [SI]
11 JC NEXT
12 MOV AL, [SI]
13 INC SI
14 LOOP BACK
15 MOV [SI], AL
16 MOV AH, 4CH
17 INT 21H
18
19 CODE ENDS
20 END START

```

The debugger window shows the memory dump at address 4000:2000, the original source code, and the assembly code with the instruction at address F400:0204 highlighted.

### 8. Write an ALP to Transfer 10 Bytes of Data From Memory Location 25000H to 49000H

The screenshot shows the QBasic IDE with the following assembly code:

```

01 CODE SEGMENT
02 ASSUME CS:CODE
03
04 START: MOV AX, 2500H
05 MOV DS, AX
06
07 MOV AX, 4900H
08 MOV ES, AX
09
10 MOV SI, 0000H
11 MOV DI, 0000H
12
13 MOV CX, 000AH
14 CLD
15 REP MOVSB
16
17 MOV AH, 4CH
18 INT 21H
19
20 CODE ENDS
21
22 END START

```

The debugger window shows the memory dump at address 4900:0000, the original source code, and the assembly code with the instruction at address F400:0204 highlighted.

## MIC - Practical - QB - Solutions - By - Th3\_

### 9. Write an ALP to Sort a Series in Ascending Order

The screenshot shows the QBasic IDE interface with the following assembly code:

```

01 CODE SEGMENT
02 ASSUME CS:CODE
03 START:
04 MOV AX, 2000H
05 MOV DS, AX
06 MOV CH, 00H
07
08 BACK2:
09 MOV CL, 04H
10 MOV SI, 5000H
11
12 BACK1:
13 MOV AL, [SI]
14 MOV AH, [SI+1]
15 CMP AL, AH
16 JC NEXT
17 JZ NEXT
18 MOV [SI+1], AL
19 MOV [SI], AH
20
21 NEXT:
22 INC SI
23 DEC CL
24 JNZ BACK1
25
26 DEC CH
27 JNZ BACK2
28
29 MOV AH, 4CH
30 INT 21H
31 CODE ENDS
32 END START

```

The assembly code window shows the memory dump from 2000:5000 to 2000:500B. The registers window shows the state at address 0710:001C. The emulator window shows the assembly code and the current instruction at 0710:001C.

### 10. Write an ALP to Sort a Series in Descending Order

The screenshot shows the QBasic IDE interface with the following assembly code:

```

01 CODE SEGMENT
02 ASSUME CS:CODE
03 START:
04 MOV AX, 2000H
05 MOV DS, AX
06 MOV CH, 00H
07
08 BACK2:
09 MOV CL, 04H
10 MOV SI, 5000H
11
12 BACK1:
13 MOV AL, [SI]
14 MOV AH, [SI-1]
15 CMP AL, AH
16 JNC NEXT
17 JZ NEXT
18 MOV [SI+1], AL
19 MOV [SI], AH
20
21 NEXT:
22 INC SI
23 DEC CL
24 JNZ BACK1
25
26 DEC CH
27 JNZ BACK2
28
29 MOV AH, 4CH
30 INT 21H
31 CODE ENDS
32 END START

```

The assembly code window shows the memory dump from 2000:5000 to 2000:500B. The registers window shows the state at address 0710:0023. The emulator window shows the assembly code and the current instruction at 0710:0023.

## MIC - Practical - QB - Solutions - By - Th3\_

### 11. Write an ALP to Compare Two Strings

The screenshot shows the QBasic IDE interface with the following code:

```

01 DATA SEGMENT
02     ARRAY1 DB 'GOOD'
03     ARRAY2 DB 'GOOD'
04     CNT DB 04H
05     STR1 DB 'STRINGS ARE EQUAL $'
06     STR2 DB 'STRINGS ARE UNEQUAL $'
07 DATA ENDS
08
09 CODE SEGMENT
10     ASSUME CS:CODE, DS:DATA, ES:DATA
11
12 START:
13     MOV AX, DATA
14     MOV DS, AX
15     MOV ES, AX
16     MOV CL, CNT
17     LEA SI, ARRAY1
18     LEA DI, ARRAY2
19     REP CMPSB
20     JNZ L1
21     LEA DX, STR1
22     JMP L2
23
24 L1:
25     LEA DX, STR2
26
27 L2:
28     MOV AH, 09H
29     INT 21H
30     MOV AH, 4CH
31     INT 21H
32
33 CODE ENDS
34 END START
  
```

The assembly output in the emulator window shows the string comparison logic:

```

19 REP CMPSB
20 JNZ L1
21 LEA DX, STR1
22 JMP L2
23
24 L1:
25 LEA DX, STR2
26
27 L2:
28 MOV AH, 09H
29 INT 21H
30 MOV AH, 4CH
31 INT 21H
  
```

The registers and memory dump windows show the state of the program during execution.

### 12. Write an ALP to Check whether String is Palindrome or not

The screenshot shows the QBasic IDE interface with the following code:

```

01 DATA SEGMENT
02     STRING1 DB 07H, 16H, 25H, 22H, 25H, 16H, 07H
03     PAL DB 00H
04 DATA ENDS
05
06 EXTRA SEGMENT
07     STRING2 DB 07H DUP(?)
08 EXTRA ENDS
09
10 CODE SEGMENT
11     ASSUME CS:CODE, DS:DATA, ES:EXTRA
12
13 START:
14     MOV AX, DATA
15     MOV DS, AX
16     MOV AX, EXTRA
17     MOV ES, AX
18
19     LEA SI, STRING1
20     LEA DI, STRING2 + 06H
21     MOV CX, 0007H
22
23 BACK:
24     CLD
25     LODSB
26     STD
27     STOSB
28     LOOP BACK
29
30     LEA SI, STRING1
31     LEA DI, STRING2
32     MOV CX, 0007H
33
34     REPE CMPSB
35     JNZ NEXT
36     INC PAL
37
38 NEXT:
39     MOV AH, 4CH
40     INT 21H
41
42     MOV AH, 4CH
43     INT 21H
44
45 CODE ENDS
46 END START
  
```

The assembly output in the emulator window shows the string comparison logic:

```

19 LEA SI, STRING1
20 LEA DI, STRING2
21 MOV CX, 0007H
22 REPE CMPSB
23 JNZ NEXT
24 INC PAL
25
26 NEXT:
27 MOV AH, 4CH
28 INT 21H
29
30 MOV AH, 4CH
31 INT 21H
  
```

The registers and memory dump windows show the state of the program during execution.

## MIC - Practical - QB - Solutions - By - Th3\_

### 13. Write an ALP to count ODD and EVEN numbers

The screenshot shows the QBasic IDE interface with the following code:

```

01 CODE SEGMENT
02 ASSUME CS:CODE
03
04 START:
05     MOV AX, 2000H
06     MOV DS, AX
07     MOV SI, 5000H
08     MOV CX, 05H
09     MOV BL, 00H
0A     MOV BH, 00H
0B
0C BACK:
0D     MOV AL, [SI]
0E     ROR AL, 01H
0F     JC ODD
0G     INC BL
0H     JMP NEXT
0I
0J ODD:
0K     INC BH
0L
0M NEXT:
0N     INC SI
0O     LOOP BACK
0P
0Q     MOV [SI], BX
0R     MOV AH, 4CH
0S
0T     INT 21H
0U
0V CODE ENDS
0W END START
0X

```

The assembly output in the emulator window shows the following assembly code:

```

JMP NEXT
ODD:
INC BH
NEXT:
INC SI
LOOP BACK
MOV [SI], BX
MOV AH, 4CH
INT 21H

```

The registers window shows:

	H	L
AX	4C	83
BX	03	02
CX	09	00
DX	00	00
CS	F400	
IP	0204	
SS	0710	
SP	FFFA	
BP	0000	
SI	5005	
DI	0000	
DS	2000	
ES	0700	

The stack dump window shows:

	F400:0204	F400:0204
F4200:	FF 255 RES	BIOS DI
F4201:	FF 255 RES	INT 021h
F4202:	CD 205 =	INT
F4203:	21 033 !	HLT
F4204:	CB 207 =	ADD BX + SI, AL
F4205:	00 000 NULL	ADD BX + SI, AL
F4207:	00 000 NULL	ADD BX + SI, AL
F4208:	00 000 NULL	ADD BX + SI, AL
F4209:	00 000 NULL	ADD BX + SI, AL
F420A:	00 000 NULL	ADD BX + SI, AL
F420B:	00 000 NULL	ADD BX + SI, AL
F420C:	00 000 NULL	ADD BX + SI, AL
F420D:	00 000 NULL	ADD BX + SI, AL
F420E:	00 000 NULL	ADD BX + SI, AL
F420F:	00 000 NULL	ADD BX + SI, AL
F4210:	00 000 NULL	ADD BX + SI, AL
F4211:	00 000 NULL	ADD BX + SI, AL
F4212:	00 000 NULL	ADD BX + SI, AL
F4213:	00 000 NULL	ADD BX + SI, AL
F4214:	00 000 NULL	ADD BX + SI, AL
F4215:	00 000 NULL	ADD BX + SI, AL

### 14. Write an ALP to Check whether given number is Zero, Positive or Negative

The screenshot shows the QBasic IDE interface with the following code:

```

01 CODE SEGMENT
02 ASSUME CS:CODE
03
04 START:
05     MOV AX, 2000H
06     MOV DS, AX
07     MOV SI, 2000H
08     MOV CX, 05H
09     MOV BL, 00H
0A     MOV BH, 00H
0B     MOV DL, 00H
0C
0D BACK:
0E     MOV AL, [SI]
0F     ADD AL, 00H
0G     JZ ZERO
0H     ROL AL, 01H
0I     JC NEGATIVE
0J     INC BL
0K     JMP NEXT
0L
0M NEGATIVE:
0N     INC BH
0O     JMP NEXT
0P
0Q ZERO:
0R     INC DL
0S
0T NEXT:
0U     INC SI
0V     LOOP BACK
0W
0X     MOV [SI], BL
0Y     MOV [SI+1], BH
0Z     MOV [SI+2], DL
0A
0B CODE ENDS
0C END START
0D

```

The assembly output in the emulator window shows the following assembly code:

```

JMP NEXT
ZERO:
INC DL
NEXT:
INC SI
LOOP BACK
MOV [SI], BL
MOV [SI+1], BH
MOV [SI+2], DL

```

The registers window shows:

	H	L
AX	20	00
BX	00	03
CX	00	00
DX	00	02
CS	0710	
IP	0044	
SS	0710	
SP	0000	
BP	0000	
SI	2005	
DI	0000	
DS	2000	
ES	0700	

The stack dump window shows:

	0710:0044	0710:0044
0711E:	90 144 E	NOP
0711F:	90 144 E	NOP
07120:	90 144 E	NOP
07121:	90 144 E	NOP
07122:	90 144 E	NOP
07123:	90 144 E	NOP
07124:	90 144 E	HLT
07125:	00 000 NULL	ADD BX + SI, AL
07126:	00 000 NULL	ADD BX + SI, AL
07127:	00 000 NULL	ADD BX + SI, AL
07128:	00 000 NULL	ADD BX + SI, AL
07129:	00 000 NULL	ADD BX + SI, AL
0712A:	00 000 NULL	ADD BX + SI, AL
0712B:	00 000 NULL	ADD BX + SI, AL
0712C:	00 000 NULL	ADD BX + SI, AL
0712D:	00 000 NULL	ADD BX + SI, AL
0712E:	00 000 NULL	ADD BX + SI, AL
0712F:	00 000 NULL	ADD BX + SI, AL
07130:	00 000 NULL	ADD BX + SI, AL
07131:	00 000 NULL	ADD BX + SI, AL
07132:	00 000 NULL	ADD BX + SI, AL
07133:	00 000 NULL	ADD BX + SI, AL
07134:	00 000 NULL	ADD BX + SI, AL
07135:	00 000 NULL	ADD BX + SI, AL

## MIC - Practical - QB - Solutions - By - Th3\_

15. Write an ALP using procedure to solve equation such as  $Z = (A + B) * (C + D)$

The screenshot shows the QBasic IDE interface with the following code:

```

01 DATA SEGMENT
02 A DB 10H
03 B DB 20H
04 C DB 30H
05 D DB 40H
06 Z DW ?
07 DATA ENDS
08
09 CODE SEGMENT
10 ASSUME CS:CODE, DS:DATA
11
12 START:
13     MOV AX, DATA
14     MOV DS, AX
15
16     MOV AL, A
17     MOV BL, B
18     CALL SUM
19
20     MOV CL, AL
21     MOV AL, C
22     MOV BL, D
23     CALL SUM
24
25     MUL CL
26     MOV Z, AX
27
28     MOV AH, 4CH
29     INT 21H
30
31 SUM PROC NEAR
32     ADD AL, BL
33     RET
34 SUM ENDP
35
36 CODE ENDS
37 END START
38

```

The assembly code window shows the following variables:

Variable	Value
A	10h
B	20h
C	30h
D	40h
Z	1500h

The debugger window shows the registers and memory dump. The CPU register pane shows:

Register	H	L	Value
AX	4C	00	FF 255
BX	00	40	CD 207
CX	00	30	INT 21h
DX	00	00	NULL
CS	F400		
IP	0204		
SS	0710		
SP	FFFA		
BP	0000		
SI	0000		
DI	0000		
DS	0710		
ES	0700		

The memory dump pane shows the stack area (F400:0204) containing the INT 21h instruction.