Solution Hw#3

1. 8 bit

2. 8 bit

3. 8 bit

4. DPTR or program counter

5. Necessary.

6. 28H and it is kept in accumulator.

7. (a),(d),(g) are illegal and for (f) only 0 is required before F5H

8. (a), (c),(d), (f), (g) are illegal.

9. 44H and kept in Accumulator (A).

10. 1EH and kept in Accumulator (A).

1. 255

2. The instruction following the jump instruction

3. Program Counter (PC)

4. The mnemonic SJMP stand for Short Jump. 2-byte instruction

5. The mnemonic LJMP stand for long jump, it’s a 3-byte instruction

6. SJMP is short jump. In this 2-byte instruction the first byte is opcode and second byte is relative address of target location. This can save some byte of memory in many applications where memory space is in short supply where as LJMP (Long Jump) is 3-byte instruction in which 1st byte is opcode and 2nd and 3rd byte represent the 16- bit address of target location.

7. True

8. False

9. LJMP

10. Short Jump. 2-byte instruction see #6 above)

11. True

12. MOV R1, #100

HERE1: MOV R2 ,#10

HERE: DJNZ R2, HERE

DJNZ R1, HERE1

13. MOV R1, #100

HERE2: MOV R2, #100

HERE1: MOV R3, #10

HERE: DJNZ R3, HERE

DJNZ R2, HERE1

DJNZ R1, HERE2

14. 200\*100=20,000 (times).

15. 128 Bytes

16. 127 Bytes

17. 3 byte

18. 2 byte

19. 2 K

20. 64Kb

21. 2 bytes

22. 2 bytes

23. They need to be equal (in order for the stack to remain unchanged) so that when the RET instruction is executed control goes back where it originated.

24. The action associated with the POP instructions is simply the reverse of the PUSH instruction. The contents of stack (pointed by stack pointer) are copied to the destination register and stack pointer is decremented by one.

25. SP Stack Content

09 00

08 0E

1.

(a) CY=0 No carry out from D7  
AC=1 carry from D3 to D4

(b) CY=0 No carry out from D7

AC=1 carry from D3 to D4

(c) CY=1  
AC=0

(d) CY=1  
AC=1

(e) CY=1  
AC=0

(f) CY=0  
AC=0

3.

**ORG 250H**

**MYDATA: DB 54, 86, 21, 45, 90, 75, 31, 80**

**ORG 0**

**MOV DPTR ,#MYDATA** ;Save the starting location of

;MYDATA (MOV;DPTR, #250 H is another option ;here)

**MOV R7, #8** ;Number of data to be added

**MOV R3, #0** ;clear R3

**MOV R2, #0** ;clear R2

**LOOP: CLR A ;** clear A

**MOVC A, @A+DPTR** ; Access MYDATA one at a time ;and move it to A

**ADD A, R3 ;** A = A+R3 add data one at a time

**MOV R3, A ;** Save the sum in R3

**JNC NOCARRY ;** Jump to NOCARRY if there

;is no carry

**INC R2 ;** Increment R2 if there is a carry

**NOCARRY: INC DPTR ;** Increment DPTR

**DJNZ R7, LOOP ;** Decrement R7 and execute

;addition again

4. Just use DA( Decimal Adjust instruction in question 3)

6.   
1. Take the 2’s complement of the subtrahend  
2. Add it to the minuend  
3. Invert the carry

(a) 23H-12H  
23H -> 00100011   
12H -> 00010010 – 2’s complement+11101110  
100010001  
Step 3: CY=1 -> CF=0 **Answer**=**11H(00010001)**  
(b) 43H-53H  
43H -> 01000011   
53H -> 01010011 – 2’s complement+10101101  
11110000  
Step 3: CY=0 -> CF=1 **Answer=** **F0H**  
(c) 99-99  
99 -> 10011001   
99 -> 10011001 – 2’s complement+01100111  
100000000  
Step 3: CY=1 -> CF=0 **Answer**=**0**