Solution for HW#3: Instruction Set Summary (80 points)
ECE473/ECE573, Microprocessor System Design

1. Determine the Opcode for each of the following instructions (10 points)
   (1) ADD A, R3
       (see page p249: MOV A, Rn)
       Opcode = 00101rrr = 00101011B = 2BH (1-byte instruction)

   (2) MOV P2, A
       (see page 270: MOV direct, A)
       Opcode = 11110101 aaaaaaaa = F5H, A0H (2-byte instruction)

   (3) MOV PSW, #0F3H
       (see page 271: MOV direct, #data)
       Opcode = 01110101 aaaaaaaa dddddddd = 75H, D0H, F3H (3-byte instruction)

   (4) ANL A, 56H
       (see page 252: ANL A, direct)
       Opcode = 01010101 aaaaaaaa = 55H, 56H (2-byte instruction)

   (5) MOV A, @R0
       (see page 269: MOV A, @Ri) ➔ Note that only R0 and R1 are used!!!
       Opcode = 1110011i = 11100110 = E6H (1-byte instruction)

   (6) MOV DPTR, #8040H
       (see page 272: MOV DPTR, #data16)
       Opcode = 10010000 dddddddd dddddddd = 90H, 80H, 40H (3-byte instruction)

   (7) SJMP A032H (in memory locations of A005 and A006)
       (see page 283: SJMP rel)
       Opcode = 10000000 eeeeee = 80H, 2BH (here A032-A007=2BH) (2-byte instruction)

   (8) AJMP 1F35H (in memory locations of 1F21 and 1F22)
       (see page 250: AJMP addr11)
       Since 1F35H = 0001 1(111: moved to front) 0011 0101B
       Opcode = aaa00011aaaaaa = (111)00001 0011 0101 = E1H, 35H (2-byte instruction)

   (9) LJMP 7FE3H
       (see page 268: JMP addr16)
       Opcode = 00000010 aaaaaaaa aaaaaaaa = 02H, 7FH, E3H (3-byte instruction)

   (10) MOVC A, @A+DPTR
       (see page 273: MOVC A, @A+DPTR)
       Opcode = 10010011 = 93H (1-byte instruction)
2. Write a program to copy the value 68H to RAM memory locations 40H to 89H using indirect addressing mode with a loop. (5 points)

```
    MOV A, #68H    ; A=68H
    MOV R0, #40H   ; R0 is a pointer starting from 40H
    MOV R2, #73    ; 89H-40H=49H = 73 decimal (loop iterations)
    Again: MOV @R0, A ; loop
            INC R0
            DJNZ R2, Again
```

3. Write a program to clear 20 RAM locations starting at RAM address 80H. (5 points)

```
    CLR A          ; the content in ACC is used to clear RAM locations
    MOV R1, #80H   ; R1 is a pointer starting from 80H
    MOV R7, #20    ; 20 iterations
    Again: MOV @R1, A ; loop
            INC R1
            DJNZ R7, Again
```

4. Write a program to copy a block of 10 bytes of data from RAM locations starting at 35H to RAM locations starting at 60H. (5 points)

```
    MOV R0, #35H   ; R0 is a pointer starting from 35H
    MOV R1, #60H   ; R1 is the other pointer starting from 60H
    MOV R3, #10    ; 10 iterations
    Back: MOV A, @R0 ; loop
             MOV @R1, A
             INC R0
             INC R1
             DJNZ R3, Back
```

5. Assuming that ROM space starting at 248H contains “Shiue”, write a program to transfer the bytes into RAM locations starting at 37H using indexed address mode (6 points)

```
    ORG 0          ; ROM location starts from 0000H
    MOV DPTR, #MYDATA ; Copy the content in ROM to RAM
    MOV R0, #37H    ; R0 is a pointer starting from 37H
    MOV R2, #5      ; “Shiue” has 5 characters
    Back: CLR A      ; A is reused, so it needs to be cleared
              MOVC A, @A+DPTR ; DPTR is a pointer
              MOV @R0, A
              INC DPTR
              INC R0
```
6. Write a program to get the x value from P1 and send $x^2$ to P2 continuously using indexed addressing mode. (6 points)

```assembly
ORG 0 ; ROM locations starts from 0000H
MOV DPTR, #300H ; DPTR is a pointer starting from 300H
MOV A, #0FFH ; Make P1 as an INPUT
MOV P1, A ;
Back: MOV A, P1 ; get x from P1
MOVC A, @A+DPTR ; get $x^2$ from table
MOV P2, A
SJMP Back

ORG 300H
Table: DB 0, 1, 4, 9, 16, 25, 36, 49, 64, 81
END
```

7. Illustrate how to add two 6-digit binary-coded decimal (BCD). The first is in internal memory locations 40H, 41H, and 42H, and the second is in locations 43H, 44H, and 45H. The most significant digits are in locations 40H and 43H. Place the BCD result in locations 70H, 71H and 72H. (4 points)

```assembly
MOV A, 42H
ADD A, 45H ; Add 2 low-byte
DA A ; Decimal adjust for BCD
MOV 70H, A

MOV A, 41H
ADDC A, 44H ; using ADDC due to carry
DA A
MOV 71H, A

MOV A, 40H
ADDC A, 43H ; using ADDC due to carry
DA A
MOV 72H, A
```
8. Write a program to add two 16-bit numbers. The numbers are 3CE7H and 3B8DH. Place the sum in R7 (high byte) and R6 (low byte). (4 points)

```
CLR C
MOV A, #0E7H
ADD A, #80H
MOV R6, A  ; low byte

MOV A, #3CH
ADDC A, #3BH
MOV R7, A  ; high byte
```

9. Assume that 5 BCD data item are stored in RAM starting at 40H. Write a program to sum all numbers. The result should be BCD. 40=(71H), 41=(11H), 42=(65H), 43=(59H), and 44=(37H) (6 points)

```
MOV R0, #40H ; R0 is a pointer
MOV R2, #5  ; 5 iterations
CLR A
MOV R7, A
Again:  ADD A, @R0  ; loop
        DA A      ; adjust for BCD
        JNC Next ; Jump if C=0
        INC R7   ; C=1
Next:   INC R0   ; C=0
        DJNZ R2, Again
```

10. Write a program to get hex data in the range of 00~FFH from P1 and convert it to decimal. Save the digits in R7 (LSB), R6, and R5. Analyze the program, assuming that P1 has a value of FDH for data. (hex converts to decimal using DIV AB) (6 points)

```
MOV A, #0FFH
MOV P1, A  ; P1 as an INPUT
MOV A, P1  ; Get data from P1

MOV B, #10 ; Hex ➔ Decimal
DIV AB
MOV R7, B

MOV B, #10
DIV AB
MOV R6, B

MOV R5, A ; if A <10
11. Assume P1 is an input port connected to a temperature sensor. Write a program to read the temperature and test it for the value 75. (6 points)

   if T = 75 \Rightarrow A=75
   T < 75 \Rightarrow R1=T
   T > 75 \Rightarrow R2=T

   MOV P1, #0FFH ; P1 is an INPUT
   MOV A, P1    ; get data from P1
   CJNE A, #75, Over
   SJMP Exit    ; if A=75, exit
   Over: JNC Next
   MOV R1, A    ; A<75
   SJMP Exit
   Next: MOV R2, A  ; A>75
   Exit: :

12. Assume internal RAM memory locations 40H~44H contain the daily temperature for 5 days. Such as 40H=(76), 41H=(79), 42H=(69), 43H=(65), 44H=(62). Search to see if only if the value equals to 65. If value 65 does exist in the table, give its location to R4; otherwise, make R4=0. (6 points)

   MOV R4, #0
   MOV R0, #40H
   MOV R2, #05
   Back: MOV A, #65
          CJNE A, @R0, Next
          MOV R4, R0
          SJMP Exit
   Next: INC R0
          DJNZ R2, Back
   Exit: :

13. Write a program that finds the number of 1’s in a given byte. (6 points)

   MOV R1, #0
   MOV R7, #8
   MOV A, #97H
   Again: RLC A
          JNC Next
   INC R1
   Next: DJNZ R7, Again
14. Assume the register A has packed BCD (29H), write a program to convert packed BCD to two ASCII numbers and place them in R2 and R6. (5 points)

```
MOV A, #29H
MOV R2, A
ANL A, #0FH
ORL A, #30H
MOV R6, A

MOV A, R2
ANL A, #0F0H
SWAP A
ORL A, #30H
MOV R2, A
```