Name: Meetraj Desai

Roll Number: CE070

ID: 20CEUOG107

LAB 04

Practice Assignment

1. Study and Analyze given instructions

a. DAA

The DAA (**Decimal Adjust after Addition**) instruction allows addition of numbers represented in 8-bit packed BCD code. It is used immediately after normal addition instruction operating on BCD codes. This instruction assumes the AL register as the source and the destination, and hence it requires no operand .

b. Branching Instruction JMP (Unconditional and Conditional)

Jump unconditionally

JMP	16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.

Jump conditionally

Operand: 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below.
Example: JZ 2034H or JZ XYZ.

2. HEX to BCD Conversion

a. (93F)16 =(0010001101100111)BCD

b. (1D)16 = (00101001)BCD

3. BCD to HEX Conversion

a. (1011001) BCD =(3B)16

b. (101100111) BCD = (A7)16

4. BCD to Binary Conversion

a. (1011001) BCD = (111011)2

b. (101100111) BCD =(10100111)2

5. Binary to BCD Conversion

a. (10110) 2 = (00100010)BCD

b. (10011) 2 = (00011001)BCD

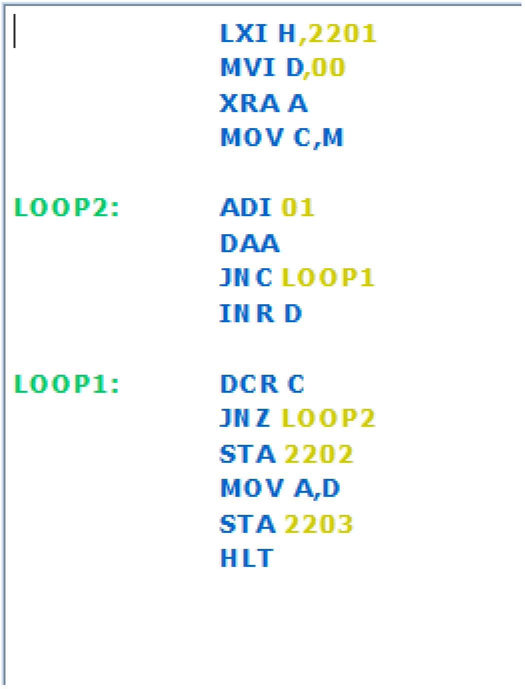
Assignment:

1. WAP for HEX to BCD conversion of a byte.

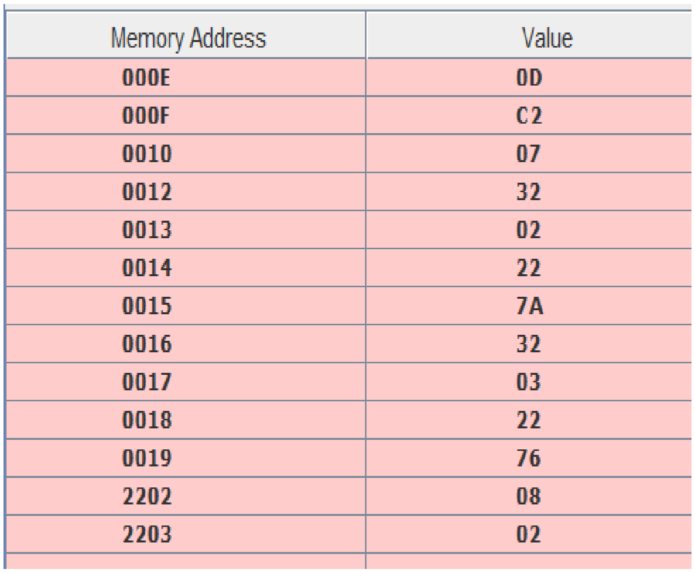
Input: (2201H) = 1CH

Output: (2202H) = 02H

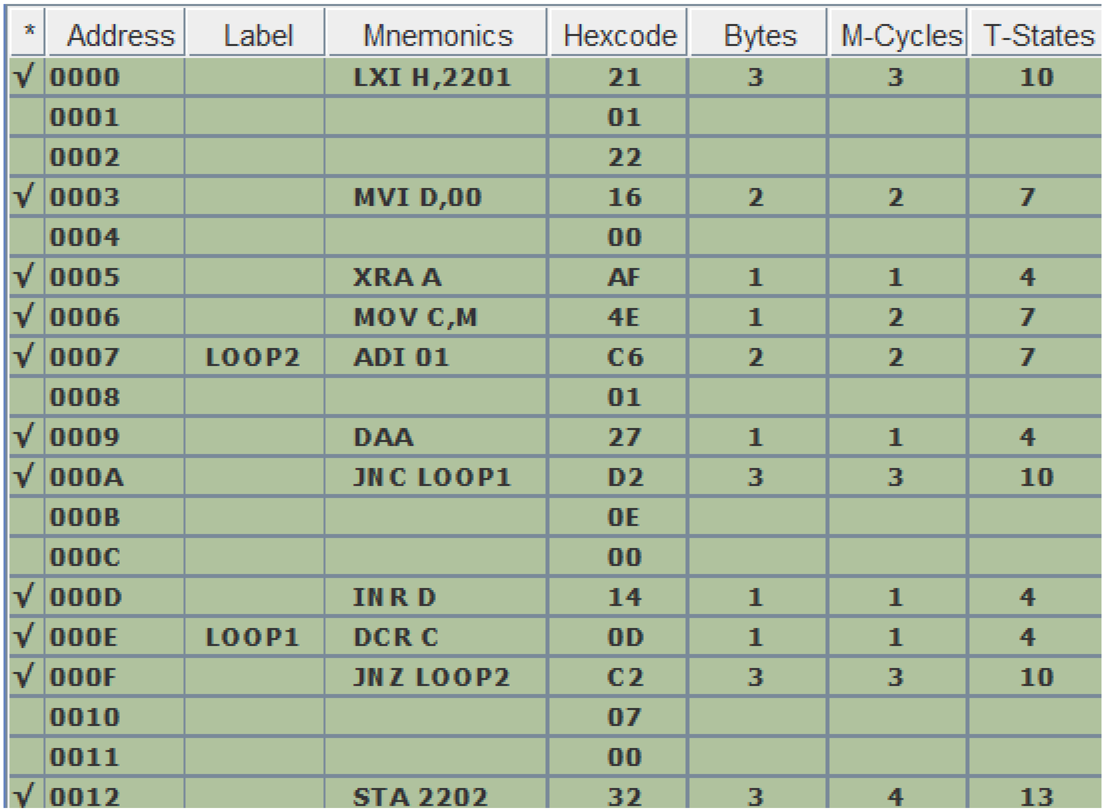
(2203H) = 08H



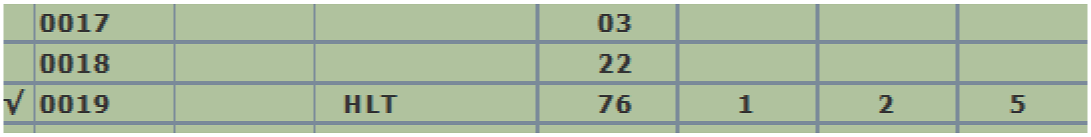
```
                LXI H,2201
                MVI D,00
                XRA A
                MOV C,M

LOOP2:          ADI 01
                DAA
                JNC LOOP1
                INR D

LOOP1:          DCR C
                JNZ LOOP2
                STA 2202
                MOV A,D
                STA 2203
                HLT
```



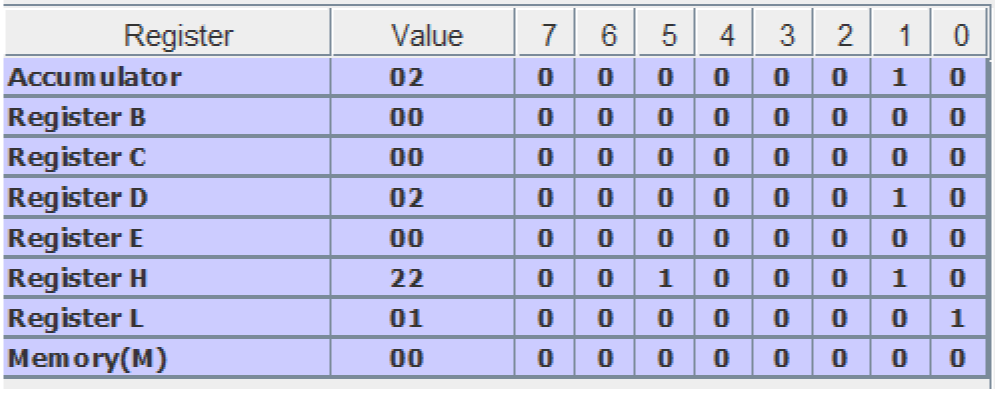
Memory Address	Value
000E	0D
000F	C2
0010	07
0012	32
0013	02
0014	22
0015	7A
0016	32
0017	03
0018	22
0019	76
2202	08
2203	02



*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	0000		LXI H,2201	21	3	3	10
	0001			01			
	0002			22			
√	0003		MVI D,00	16	2	2	7
	0004			00			
√	0005		XRA A	AF	1	1	4
√	0006		MOV C,M	4E	1	2	7
√	0007	LOOP2	ADI 01	C6	2	2	7
	0008			01			
√	0009		DAA	27	1	1	4
√	000A		JNC LOOP1	D2	3	3	10
	000B			0E			
	000C			00			
√	000D		INR D	14	1	1	4
√	000E	LOOP1	DCR C	0D	1	1	4
√	000F		JNZ LOOP2	C2	3	3	10
	0010			07			
	0011			00			
√	0012		STA 2202	32	3	4	13



	0017			03			
	0018			22			
√	0019		HLT	76	1	2	5



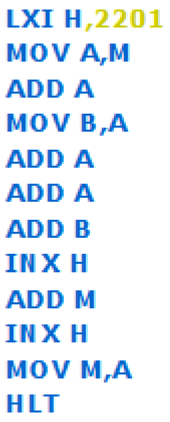
Register	Value	7	6	5	4	3	2	1	0
Accumulator	02	0	0	0	0	0	0	1	0
Register B	00	0	0	0	0	0	0	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	02	0	0	0	0	0	0	1	0
Register E	00	0	0	0	0	0	0	0	0
Register H	22	0	0	1	0	0	0	1	0
Register L	01	0	0	0	0	0	0	0	1
Memory(M)	00	0	0	0	0	0	0	0	0

2. WAP for BCD to HEX conversion of a byte.

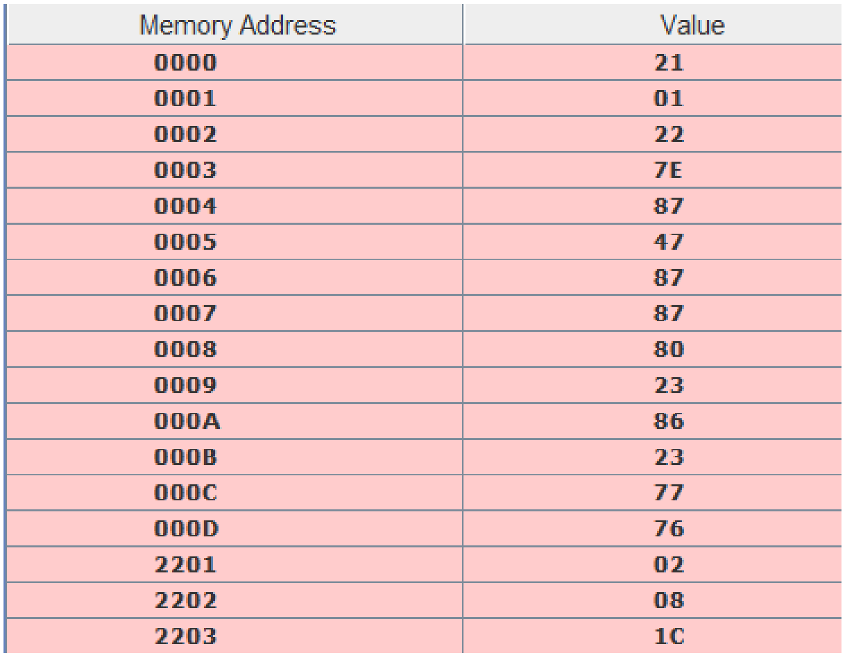
Input: (2201H) = 02H

(2202H) = 08H

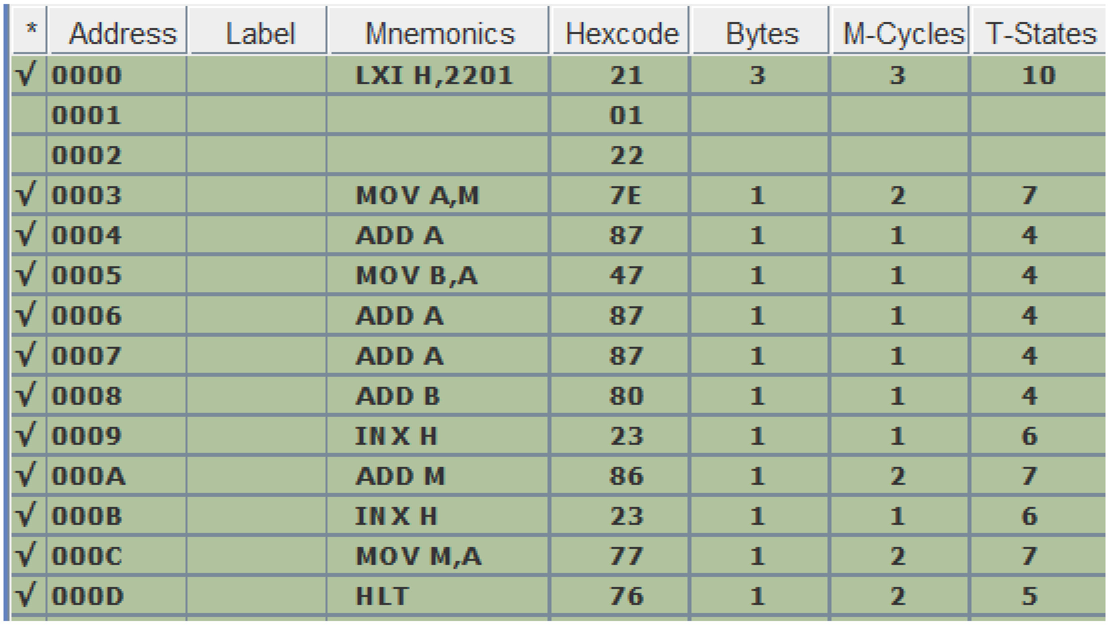
Output: (2203H) = 1CH



```
LXI H,2201
MOV A,M
ADD A
MOV B,A
ADD A
ADD A
ADD B
INX H
ADD M
INX H
MOV M,A
HLT
```



Memory Address	Value
0000	21
0001	01
0002	22
0003	7E
0004	87
0005	47
0006	87
0007	87
0008	80
0009	23
000A	86
000B	23
000C	77
000D	76
2201	02
2202	08
2203	1C

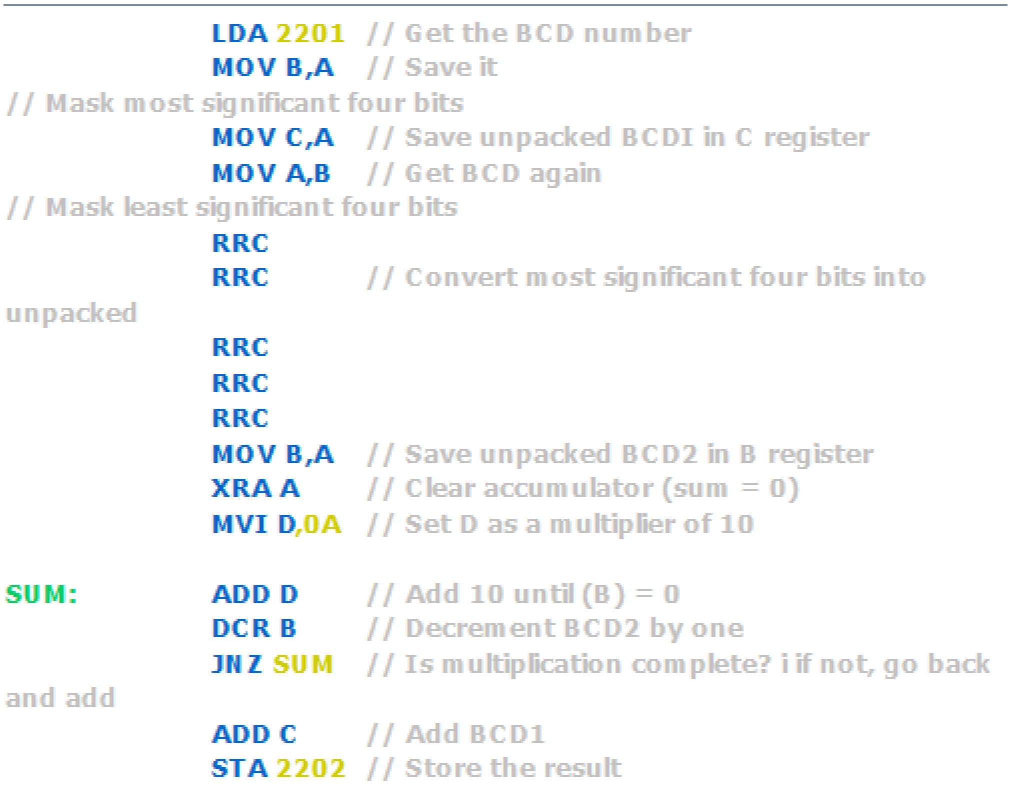


*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	0000		LXI H,2201	21	3	3	10
	0001			01			
	0002			22			
√	0003		MOV A,M	7E	1	2	7
√	0004		ADD A	87	1	1	4
√	0005		MOV B,A	47	1	1	4
√	0006		ADD A	87	1	1	4
√	0007		ADD A	87	1	1	4
√	0008		ADD B	80	1	1	4
√	0009		INX H	23	1	1	6
√	000A		ADD M	86	1	2	7
√	000B		INX H	23	1	1	6
√	000C		MOV M,A	77	1	2	7
√	000D		HLT	76	1	2	5

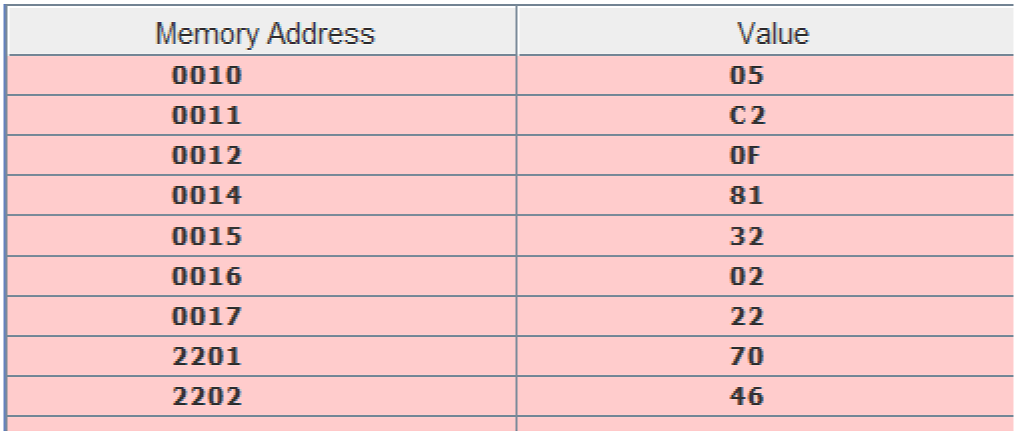
3. WAP for BCD to binary conversion of a byte.

Input: (2201H) = 70H

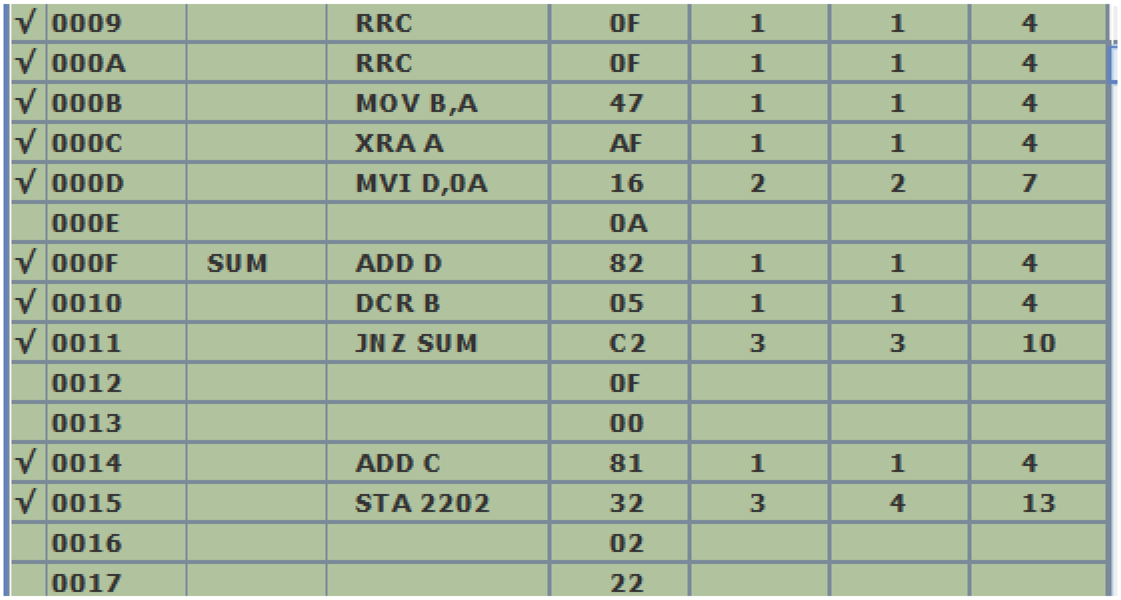
Output: (2202H) = 46H



```
                LDA 2201   // Get the BCD number
                MOV B,A    // Save it
// Mask most significant four bits
                MOV C,A    // Save unpacked BCDI in C register
                MOV A,B    // Get BCD again
// Mask least significant four bits
                RRC
                RRC        // Convert most significant four bits into
unpacked
                RRC
                RRC
                RRC
                MOV B,A    // Save unpacked BCD2 in B register
                XRA A      // Clear accumulator (sum = 0)
                MVI D,0A   // Set D as a multiplier of 10

SUM:            ADD D      // Add 10 until (B) = 0
                DCR B      // Decrement BCD2 by one
                JNZ SUM    // Is multiplication complete? i if not, go back
and add
                ADD C      // Add BCD1
                STA 2202   // Store the result
```



Memory Address	Value
0010	05
0011	C2
0012	0F
0014	81
0015	32
0016	02
0017	22
2201	70
2202	46



√	0009		RRC	0F	1	1	4
√	000A		RRC	0F	1	1	4
√	000B		MOV B,A	47	1	1	4
√	000C		XRA A	AF	1	1	4
√	000D		MVI D,0A	16	2	2	7
	000E			0A			
√	000F	SUM	ADD D	82	1	1	4
√	0010		DCR B	05	1	1	4
√	0011		JNZ SUM	C2	3	3	10
	0012			0F			
	0013			00			
√	0014		ADD C	81	1	1	4
√	0015		STA 2202	32	3	4	13
	0016			02			
	0017			22			