MPMC LAB ASSIGNMENT-12

NAME: Pratyush Nandi

ROLL NUMBER: IMT2017518

AIM: Write a code which will load an ihex file into one kit and then transfer it to another kit, decode and load the program, run and test it. When the entire file is successfully decoded and loaded into memory, read keys to override this starting address in hex until EXEC key is pressed. Jump to this address to run it.Test this program like in Exercise 9 and verify that it executes successfully. This program should exit back to the Monitor.

EQUIPMENT AND TOOLS:

- Two ESA 31 kits with power adapters.
- 3 jumper cables.
- Connect it GND-GND, Tx-Rx and Rx-Tx.
- 8051 Emulator (mcu8051ide or equivalent).
- Program code and listing for Exercise 9 (ihex encoder).
- The baud rate is fixed at 9600 bps.

ESA SUBROUTINES:

- **02A2H** is the subroutine called for taking input.
- **18ADH** is the subroutine called for writing in program code.
- **04FDH** is the subroutine called for returning control back to monitor.

PREPARATION:

INTEL IHEX FILE:

- This file should be stored in a location such that it shouldn't overlap with any piece of code that is occupied by subroutine as mentioned in the file.
- So preferably store it in data memory in location **A000H.**

IHEX ENCODER(CAN BE USED AS ONE INPUT CASE):

- **TO BE STORED IN HOST KIT:**
  - Start storing the code in data memory at **A000H** which begins from **INITIALIZE** header.
- **CODE:**
  - **APPENDIX:**

Assignment1
PAGE 1

```
                    1     INITIALIZE:     ORG     0A000H
A000 7FBF           2             MOV     R7, #0BFH
A002 7EFA           3             MOV     R6, #0FAH
A004 7DCF           4             MOV     R5, #0CFH
A006 7CFA           5             MOV     R4, #0FAH
A008 8D83           6             MOV     DPH, R5
A00A 8C82           7             MOV     DPL, R4
                    8
                    9     HEADER:
A00C 743A           10            MOV     A, #3AH
A00E F0             11            MOVX    @DPTR, A
A00F A3             12            INC     DPTR
A010 7410           13            MOV     A, #10H
A012 F0             14            MOVX    @DPTR, A
A013 2B             15            ADD     A, R3
A014 FB             16            MOV     R3, A
A015 A3             17            INC     DPTR
A016 EF             18            MOV     A, R7
A017 F0             19            MOVX    @DPTR, A
A018 2B             20            ADD     A, R3
A019 FB             21            MOV     R3, A
A01A A3             22            INC     DPTR
A01B EE             23            MOV     A, R6
A01C F0             24            MOVX    @DPTR, A
A01D 2B             25            ADD     A, R3
A01E FB             26            MOV     R3, A
A01F A3             27            INC     DPTR
A020 7400           28            MOV     A, #00H
A022 F0             29            MOVX    @DPTR, A
A023 2B             30            ADD     A, R3
A024 FB             31            MOV     R3, A
A025 A3             32            INC     DPTR
A026 7A10           33            MOV     R2, #10H
A028 AD83           34            MOV     R5, DPH
A02A AC82           35            MOV     R4, DPL
                    36
                    37    LOOP1:
A02C 8F83           38            MOV     DPH, R7
A02E 8E82           39            MOV     DPL, R6
A030 E0             40            MOVX    A, @DPTR
A031 A3             41            INC     DPTR
```

```
A032 AF83          42          MOV    R7, DPH
A034 AE82          43          MOV    R6, DPL
A036 8D83          44          MOV    DPH, R5
A038 8C82          45          MOV    DPL, R4
A03A F0            46          MOVX   @DPTR, A
A03B A3            47          INC    DPTR
A03C AD83          48          MOV    R5, DPH
A03E AC82          49          MOV    R4, DPL
A040 2B            50          ADD    A, R3
A041 FB            51          MOV    R3, A
A042 1A            52          DEC    R2
A043 BA00E6        53          CJNE   R2, #00H, LOOP1
                   54
                   55   CHECKSUM:
A046 EB            56          MOV    A, R3
A047 F4            57          CPL    A
A048 04            58          INC    A
A049 8D83          59          MOV    DPH, R5
A04B 8C82          60          MOV    DPL, R4
A04D F0            61          MOVX   @DPTR, A
```

TRANSMITTER:

- **SENDER:**
  - Start storing the code in **8000H** which begins from **INITIALIZE** header.
  - Store the **printsub** subroutine in **0FFF0H** in program memory.
  - Store **addrsub** subroutine in **0E000H** in program memory.
  - Store **shap** subroutine in **0F050H** in program memory.
  - Store **subroutine** subroutine in **0FFF9H** in program memory.
  - After filling this press “GO” in ESA kit and enter **8000H** and press “EXEC”

- **DESIGN:**
  - **INITIALIZE:** In this section of code all registers are reset and all interrupts required for transmitting are set with necessary values. Serial interrupts, timer interrupts and timer interrupts are the ones which are set.
  - **LOOP1:** In this piece of code a call is made to a subroutine which displays start address from where the ihex code is being sent for a brief amount of time, and then the difference between higher order bytes of stop address and start address is calculated. Based on carry bit the jump is made.
  - **JUMP:** In this piece of code difference between lower order bytes of stop address and start address is calculated, we reach this loop because

difference between higher order bytes of stop address and start address turns out to be zero.

- **JUMP2:** In this piece of code accumulator is checked if it turns out to be zero that means we have given the same address for start and stop and control is returned to monitor eventually. If accumulator is not zero and carry bit is not set (check for stop address < start address), we jump to another location which resets all the register and program starts transmitting.
- **STAGE1:** In piece of code all the headers are transmitted at once. That includes [:(3A), length of ihex, address, data, data type, checksum].
- **STAGE2:** In this chunk of code we send all the data one by one and loop it back to LOOP1 so that address can be displayed and again the difference is calculated with after incrementing the start address by length of data and again all the loops and jumps are visited.
- **EOF:** This piece of code is visited when all the data is transferred, and now only last piece of code is sent ie [:(3A), 00, 00 00, 00, 01, FF]. After this we call a subroutine for printing done.
- **printsub:** It fetches the led format from 1bb7 and places it in EC00 and EC001 respectively.
- **dOnE:** This subroutine helps us print dOnE from 90 - 93 leds. It transfers data to two specific registers, first data contains led number and second register has the data to be printed in hex format. It repeatedly calls the subroutine which only places led number in EC001 and data on EC00.
- **addrsub:** This subroutine is called in LOOP1 and has only one job which is to display address in leds 90-93 using another subroutine which gets the led format from 1bb7 and places it in EC00 and EC001 respectively.
- **subroutine:** This subroutine is present in 0FFF9 which just clears the TI bit and returns.

- **CODE:**
  - **APPENDIX:**

```
Assignment2
PAGE 1
 04FD                1     HALT    EQU     04FDH
                     2
                     3
                     4
                     5
                     6     INITIALIZE:     ORG     8000H
8000 7531FD          7           MOV     31H, #0FDH
8003 753200          8           MOV     32H, #00H
```

```
8006 75333A          9          MOV    33H, #3AH      ; HEADER START
8009 753400          10         MOV    34H, #00H      ;LENGTH
800C 7535B0          11         MOV    35H, #0B0H     ;START
ADDRESS HIGH
800F 753600          12         MOV    36H, #00H      ;START
ADDRESS LOW
8012 753700          13         MOV    37H, #00H      ;TYPE
8015 7538B0          14         MOV    38H, #0B0H     ;STOP
ADDRESS HIGH
8018 75391F          15         MOV    39H, #01FH     ;STOP
ADDRESS LOW
801B 7800            16         MOV    R0, #00H
801D 758800          17         MOV    88H, #00H      ;SETTING TCON
8020 758920          18         MOV    89H, #20H      ;SETTING TMOD
8023 85318D          19         MOV    8DH, 31H       ; SETTING TH1
8026 758700          20         MOV    87H, #00H      ;SETTING SMOD
8029 759850          21         MOV    98H, #50H      ;SETTING SCON
802C 75A890          22         MOV    0A8H, #90H     ;SETTING
INTERRUPT ENABLE
                     23
                     24
802F 12E000          25    LOOP1:  LCALL addrsub
8032 E538            26         MOV    A, 38H
8034 C3              27         CLR    C
8035 9535            28         SUBB   A, 35H
8037 5002            29         JNC    JUMP
8039 805C            30         SJMP   EOF
803B B40014          31    JUMP:   CJNE   A, #00H, JUMP1
803E E539            32         MOV    A, 39H
8040 C3              33         CLR    C
8041 9536            34         SUBB   A, 36H
8043 5002            35         JNC    JUMP2
8045 8050            36         SJMP   EOF
                     37
                     38    JUMP2:
8047 B40003          39         CJNE   A, #00H, J2
804A 0204FD          40         LJMP   HALT
804D F534            41    J2:    MOV    34H, A
804F C3              42         CLR    C
```

```
8050 5003          43          JNC     JUMP3
                   44     JUMP1:
8052 7534FF         45          MOV     34H, #0FFH
                   46
                   47     JUMP3:
8055 7900          48          MOV     R1, #00H       ;STAGE REGISTER
8057 7A00          49          MOV     R2, #00H       ;CHECKSUM
REGISTER
8059 7833          50          MOV     R0, #33H       ;LENGTH
REGISTER
805B 758840         51          MOV     TCON, #40H
                   52     STAGE1:
805E D231          53          SETB    31H
8060 8699          54          MOV     SBUF, @R0
8062 2031FD         55          JB      31H, $
8065 E6            56          MOV     A, @R0
8066 2A            57          ADD     A, R2
8067 FA            58          MOV     R2, A
8068 B838F3         59          CJNE    R0, #38H, STAGE1
806B 853583         60          MOV     DPH, 35H
806E 853682         61          MOV     DPL, 36H
8071 AB37          62          MOV     R3, 37H
                   63     STAGE2:
8073 D231          64          SETB    31H
8075 E0            65          MOVX    A, @DPTR
8076 F599          66          MOV     SBUF, A
8078 2031FD         67          JB      31H, $
807B 2A            68          ADD     A, R2
807C FA            69          MOV     R2, A
807D A3            70          INC     DPTR
807E 1B            71          DEC     R3
807F BB00F1         72          CJNE    R3, #00H, STAGE2
8082 A3            73          INC     DPTR
8083 EA            74          MOV     A, R2
8084 F4            75          CPL     A
8085 2401          76          ADD     A, #01h
8087 D231          77          SETB    31h
8089 FE            78          mov     r6, a
808A F599          79          MOV     SBUF, A
```

```
808C 2031FD          80          JB      31H, $
808F 858335          81          MOV     35H, DPH
8092 858236          82          MOV     36H, DPL
8095 8098            83          SJMP    LOOP1
                     84
8097 D231            85    EOF:  SETB    31H
8099 75993A          86          MOV     SBUF,#3AH
809C 2031FD          87          JB      31H,$
809F D231            88          SETB    31H
80A1 759900          89          MOV     SBUF,#00H
80A4 2031FD          90          JB      31H,$
80A7 D231            91          SETB    31H
80A9 759900          92          MOV     SBUF,#00H
80AC 2031FD          93          JB      31H,$
80AF D231            94          SETB    31H
80B1 759900          95          MOV     SBUF,#00H
80B4 2031FD          96          JB      31H,$
80B7 D231            97          SETB    31H
80B9 759901          98          MOV     SBUF,#01H
80BC 2031FD          99          JB      31H,$
80BF D231           100          SETB    31H
80C1 7599FF         101          MOV     SBUF,#0FFH
80C4 2031FD         102          JB      31H,$
80C7 D231           103          SETB    31H
80C9 0280CC         104          LJMP    dOnE
                    105
80CC 757090         106    dOnE: MOV     70H, #90H
80CF 7571E5         107          MOV     71H, #0E5H
80D2 12F000         108          LCALL   printsub
80D5 0570           109          INC     70H
80D7 7571F3         110          MOV     71H, #0F3H
80DA 12F000         111          LCALL   printsub
80DD 0570           112          INC     70H
80DF 757145         113          MOV     71H, #45H
80E2 12F000         114          LCALL   printsub
80E5 0570           115          INC     70H
80E7 757197         116          MOV     71H, #97H
80EA 12F000         117          LCALL   printsub
80ED 80DD           118          SJMP    dOnE
```

```
                119
                120     printsub:       ORG     0F000H
F000 90EC01          121             MOV     DPTR, #0EC01H
F003 E570            122             MOV     A, 70H
F005 F0              123             MOVX    @DPTR, A
F006 90EC00          124             MOV     DPTR, #0EC00H
F009 E571            125             MOV     A, 71H
F00B F0              126             MOVX    @DPTR, A
F00C 22              127             RET
                128
                129     addrsub:        ORG     0E000H
E000 E535            130             MOV     A,35H
E002 54F0            131             ANL     A,#0F0H
E004 C4              132             SWAP    A
E005 757090          133             MOV     70H, #90H
E008 12F050          134             LCALL   shap
E00B E535            135             MOV     A,35H
E00D 540F            136             ANL     A,#0FH
E00F 0570            137             INC     70H
E011 12F050          138             LCALL   shap
E014 E536            139             MOV     A,36H
E016 54F0            140             ANL     A,#0F0H
E018 C4              141             SWAP    A
E019 0570            142             INC     70H
E01B 12F050          143             LCALL   shap
E01E E536            144             MOV     A,36H
E020 550F            145             ANL     A,0FH
E022 0570            146             INC     70H
E024 12F050          147             LCALL   shap
E027 22              148             RET
                149
                150     shap:   ORG     0F050H
F050 901BB7          151             MOV     DPTR,#1BB7H
F053 93              152             MOVC    A,@A+DPTR
F054 F571            153             MOV     71H,A
F056 12F000          154             LCALL   printsub
F059 22              155             RET
                156
                157     SUB:    org     0023H
```

```
0023 02FFF9          158          LJMP    subroutine
                     159
                     160
                     161    subroutine:    ORG    0FFF9H
FFF9 C231            162          CLR    31H
FFFB C299            163          CLR    TI
FFFD 08              164          INC    R0
FFFE 32              165          RETI

                     166
                     167
                     168
                     169
                     170
                     171
                     172
                     173
                     174
                     175
```

RECEIVER:

- **LOADER:**
  - Start storing the code in **8000H** which begins from **INITIATE** header.
  - Store the **print** subroutine in **0FFF0H** in program memory.
  - Store **addrsub** subroutine in **0B000H** in program memory.
  - Store **shap** subroutine in **0F050H** in program memory.
  - Store **subroutine1** subroutine in **0C000H** in program memory.
  - After filling this press “GO” in ESA kit and enter **8000H** and press “EXEC”
- **DESIGN**
  - **INITIATE:** In this section of code all registers are reset and all interrupts required for transmitting are set with necessary values. Serial interrupts, timer interrupts and timer interrupts are the ones which are set.
  - **READY:** In this piece of code we print “rEAdY” in leds from 90-93 and wait until we receive any data from the SENDER kit.
  - **STAGE1:** In this piece of code we wait for the for the data and once we receive it the subroutine handles the data. Some specific bits are set once header bytes are being received, this loop goes on till we receive all the header bytes. Once we receive all the header bytes we enter another loop which goes until we the length of data specified in header is received.

This loop also has a checker for checksum, so if checksum matches it displays the address and “GO” in data and leds. I checksum doesn't match then it prints “Er” in data led.

- **printsub:** It fetches the led format from 1bb7 and places it in EC00 and EC001 respectively.
- **addrsub:** This subroutine is called when checksum is correct and it has only one job which is to display address in leds 90-93 and “GO” in data leds using another subroutine which gets the led format from 1bb7 and places it in EC00 and EC001 respectively.
- **input:** This subroutine is called after displaying the address, the user can override and read 4 keys which will be used to set new address where the pc will go and after pressing the “EXEC” will run the code which is sent by the SENDER.
- **subroutine1:** This subroutine is called in FFF9h address, it handles all the headers and data and also stores the address of first ihex code. It clears the RI bit and returns to STAGE1.

- **CODE:**
  - **APPENDIX:**

```
 04FD              1    HALT    EQU     04FDH
                   2
                   3    INITIATE:       ORG     8000H
8000 7D00          4            MOV     R5, #00H        ;FOR FIRST ADDR
8002 7A05          5            MOV     R2, #05H        ;COUNTER
8004 7B00          6            MOV     R3, #00H        ;LENGTH
8006 7C00          7            MOV     R4, #00H        ;CHECKSUM
8008 7800          8            MOV     R0, #00H        ;DPL
800A 7900          9            MOV     R1, #000H       ;DPH
800C 758800        10           MOV     88H, #00H       ;SETTING TCON
800F 758920        11           MOV     89H, #20H       ;SETTING TMOD
8012 758DFD        12           MOV     8DH, #0FDH      ; SETTING TH1
8015 758700        13           MOV     87H, #00H       ;SETTING SMOD
8018 759850        14           MOV     98H, #50H       ;SETTING SCON
801B 75A890        15           MOV     0A8H, #90H      ;SETTING INTERRUPT
ENABLE
801E D28E          16           SETB    TR1             ;SET THE TIMER
8020 D232          17           SETB    32H
8022 C233          18           CLR     33H
```

```
8024 C230          19          CLR     30H
                   20
8026 757090        21    READY:  MOV     70H, #90H
8029 757105        22          MOV     71H, #05H
802C 12F000        23          LCALL   print
802F 0570          24          INC     70H
8031 757197        25          MOV     71H, #097H
8034 12F000        26          LCALL   print
8037 0570          27          INC     70H
8039 757177        28          MOV     71H, #77H
803C 12F000        29          LCALL   print
803F 0570          30          INC     70H
8041 7571E5        31          MOV     71H, #0E5H
8044 12F000        32          LCALL   print
8047 0570          33          INC     70H
8049 7571E6        34          MOV     71H, #0E6H
804C 12F000        35          LCALL   print
                   36
                   37
804F D231          38    STAGE1: SETB    31H
8051 2031FD        39          JB      31H, $
8054 BA0502        40          CJNE    R2, #05H, J          ;loop it to stage1
8057 8027          41          SJMP    J8                   ;HERE I MADE A
CHANGE
8059 2033F3        42    J:      JB      33H,STAGE1
805C BA00F0        43          CJNE    R2, #00H, STAGE1     ;CHECK R2
805F 303002        44          JNB     30H, J7
8062 801C          45          SJMP    J8           ;EOF
                   46
8064 E599          47    J7:     MOV     A, SBUF
8066 1218AD        48          LCALL   18ADH
8069 A3            49          INC     DPTR
806A 1B            50          DEC     R3
806B BB00E1        51          CJNE    R3, #00H, STAGE1
806E D232          52          SETB    32H
8070 EC            53          MOV     A, R4
8071 9599          54          SUBB    A, SBUF      ;CHECKSUM WE GOT
8073 F4            55          CPL     A
8074 2401          56          ADD     A, #01h
```

```
8076 FC          57          MOV     R4, A
8077 B59906      58          CJNE    A, SBUF, J8     ;HERE I MADE A
CHANGE
807A D232        59          SETB    32H
807C D233        60          SETB    33H           ;HERE WE GO  AGAIN
807E 80CF        61          SJMP    STAGE1
                 62
8080 02B000      63    J8:    LJMP    addrsub ;PRINT DONE
                 64
8083 757090      65    J9:    MOV     70H, #90H      ;PRINT ERROR
8086 757100      66          MOV     71H, #00H
8089 12F000      67          LCALL   print
808C 0570        68          INC     70H
808E 757197      69          MOV     71H, #097H
8091 12F000      70          LCALL   print
8094 0570        71          INC     70H
8096 757105      72          MOV     71H, #05H
8099 12F000      73          LCALL   print
809C 0570        74          INC     70H
809E 12F000      75          LCALL   print
                 76
                 77    shap:   ORG     0F050H
F050 901BB7      78          MOV     DPTR,#1BB7H
F053 93          79          MOVC    A,@A+DPTR
F054 F571        80          MOV     71H,A
F056 12F000      81          LCALL   print
F059 22          82          RET
                 83    print:  ORG     0F000H

F000 90EC01      84          MOV     DPTR, #0EC01H
F003 E570        85          MOV     A, 70H
F005 F0          86          MOVX    @DPTR, A
F006 90EC00      87          MOV     DPTR, #0EC00H
F009 E571        88          MOV     A, 71H
F00B F0          89          MOVX    @DPTR, A
F00C 22          90          RET
                 91    addrsub:       ORG     0B000H
B000 EF          92          MOV     A,R7
B001 54F0        93          ANL     A,#0F0H
```

```
B003 C4          94          SWAP    A
B004 757090        95          MOV     70H, #90H
B007 12F050        96          LCALL   shap
B00A EF          97          MOV     A,R7
B00B 540F          98          ANL     A,#0FH
B00D 0570          99          INC     70H
B00F 12F050        100         LCALL   shap
B012 EE          101         MOV     A,R6
B013 54F0          102         ANL     A,#0F0H
B015 C4          103         SWAP    A
B016 0570          104         INC     70H
B018 12F050        105         LCALL   shap
B01B EE          106         MOV     A,R6
B01C 540F          107         ANL     A,#0FH
B01E 0570          108         INC     70H
B020 12F050        109         LCALL   shap
B023 7571D3        110         MOV     71H,#0D3H
B026 0570          111         INC     70H
B028 12F000        112         LCALL   print
B02B 7571F3        113         MOV     71H,#0F3H
B02E 0570          114         INC     70H
B030 12F000        115         LCALL   print
B033 A850          116         MOV     R0,50H
                   117
B035 1202A2        118     input:  LCALL   02A2H
B038 F6          119         MOV     @R0,A
B039 08          120         INC     R0
B03A B41FF8        121         CJNE    A,#1FH,input
B03D 18          122         DEC     R0
B03E 18          123         DEC     R0
B03F E6          124         MOV     A,@R0
B040 FB          125         MOV     R3,A
B041 18          126         DEC     R0
B042 E6          127         MOV     A,@R0
B043 C4          128         SWAP    A
B044 2B          129         ADD     A,R3
B045 F582          130         MOV     DPL,A
B047 18          131         DEC     R0
B048 E6          132         MOV     A,@R0
```

```
B049 FB          133        MOV    R3,A
B04A 18          134        DEC    R0
B04B E6          135        MOV    A,@R0
B04C C4          136        SWAP   A
B04D 2B          137        ADD    A,R3
B04E F583        138        MOV    DPH,A
B050 A882        139        MOV    R0,DPL
B052 A983        140        MOV    R1,DPH
B054 7402        141        MOV    A,#02H
B056 1218AD      142        LCALL  18ADH
B059 A3          143        INC    DPTR
B05A EF          144        MOV    A,R7
B05B 1218AD      145        LCALL  18ADH
B05E A3          146        INC    DPTR
B05F EE          147        MOV    A,R6
B060 1218AD      148        LCALL  18ADH
B063 90B075      149        MOV    DPTR,#0B075H
B066 7402        150        MOV    A,#02H
B068 1218AD      151        LCALL  18ADH
B06B A3          152        INC    DPTR
B06C E9          153        MOV    A,R1
B06D 1218AD      154        LCALL  18ADH
B070 A3          155        INC    DPTR
B071 E8          156        MOV    A,R0
B072 1218AD      157        LCALL  18ADH
                 158
                 159
                 160
                 161
                 162    SUB:   org    0023H
0023 02FFF9      163        LJMP   subroutine
                 164
                 165
                 166    subroutine:   ORG    0FFF9H
FFF9 02C000      167        LJMP   subroutine1
appropriate ORG directive to clarify correct code placement.
                 168
                 169    subroutine1:
                 170        ORG    0C000H
```

```
C000 C231            171          CLR     31H
C002 E599            172          MOV     A, SBUF
C004 30320B          173          JNB     32H, J1
C007 B43A08          174          CJNE    A, #3AH, J1
C00A 0D              175          INC     R5
C00B C232            176          CLR     32H
C00D 7A04            177          MOV     R2, #04H
C00F C298            178          CLR     RI
C011 32              179          RETI
                     180
C012 2C              181    J1:   ADD     A, R4
C013 FC              182          MOV     R4, A
C014 BA0406          183          CJNE    R2, #04H, J2
C017 1A              184          DEC     R2
C018 AB99            185          MOV     R3, SBUF
C01A C298            186          CLR     RI
C01C 32              187          RETI
                     188
C01D BA030C          189    J2:   CJNE    R2, #03H, J3
C020 1A              190          DEC     R2
C021 859983          191          MOV     DPH,SBUF
C024 BD0102          192          CJNE    R5, #01H,JJ1
C027 AF83            193          MOV     R7,DPH
                     194
C029 C298            195    JJ1:  CLR     RI
C02B 32              196          RETI
                     197
C02C BA020C          198    J3:   CJNE    R2, #02H, J4
C02F 1A              199          DEC     R2
C030 859982          200          MOV     DPL,SBUF
C033 BD0102          201          CJNE    R5, #01H,JJ2
C036 AE82            202          MOV     R6,DPL
                     203
C038 C298            204    JJ2:  CLR     RI
C03A 32              205          RETI
                     206
C03B BA0112          207    J4:   CJNE    R2, #01H, J6
C03E 1A              208          DEC     R2
C03F D233            209          SETB    33H
```

```
C041 E599          210          MOV    A, SBUF
C043 B40107        211          CJNE   A, #01H, J5
C046 C233          212          CLR    33H
C048 D230          213          SETB   30H
C04A C298          214          CLR    RI
C04C 32            215          RETI
                   216
C04D C298          217    J5:   CLR    RI
C04F 32            218          RETI
                   219
C050 C233          220    J6:   CLR    33H
C052 C298          221          CLR    RI
C054 32            222          RETI
                   223    END
```

- **TESTCASE:(**Expected Output on given Input**)**
  - **TEST1**
    - **INPUT(**Data at A000H):

A000H	12H
A001H	02H
A002H	A2H
A003H	12H
A004H	04H
A005H	FDH

    - **OUTPUT(**Program at A000H**):**

A000H	12H
A001H	02H
A002H	A2H
A003H	12H

A004H 04H

A005H FDH

*This program takes an input and returns control to monitor.

- **TEST2**
  - **INPUT(**Data at A000H):

    A000H 75H

    A001H 65H

    A002H 23H

    A003H 85H

    A004H 65H

    A005H 60H

    A006H 12H

    A007H 01H

    A008H 9BH

  - **OUTPUT**(Program at A000H):

    A000H 75H

    A001H 65H

    A002H 23H

    A003H 85H

A004H 65H

A005H 60H

A006H 12H

A007H 01H

A008H 9BH

- **TEST3**
  - **INPUT(**Data at A000H):

    A000H 12H

    A001H 02H

    A002H A2H

    A003H 12H

    A004H 00H

    A005H 00H

  - **OUTPUT**(Program at A000H):

    A000H 12H

    A001H 02H

    A002H A2H

    A003H 12H

    A004H 00H

A005H 00H

*This program takes an input and jumps back to 0000H ie restart condition.

- **EXPERIMENT:(**Actual Output on given Input**)**
  - Put sender code in HOST kit and receiver code in TARGET kit.
  - Put the ihex program in data memory of host kit as mentioned above in preparation section.
  - First run the sender program in HOST kit, the run the loader program in TARGET kit.

  - **TEST1**
    - **INPUT(**Data at A000H):

      A000H 12H

      A001H 02H

      A002H A2H

      A003H 12H

      A004H 04H

      A005H FDH

    - **OUTPUT(**Program at A000H**):**

      A000H 12H

      A001H 02H

      A002H A2H

      A003H 12H

      A004H 04H

      A005H FDH

*This program takes an input and returns control to monitor.

- **TEST2**
  - **INPUT(**Data at A000H):

A000H	75H
A001H	65H
A002H	23H
A003H	85H
A004H	65H
A005H	60H
A006H	12H
A007H	01H
A008H	9BH

  - **OUTPUT**(Program at A000H):

A000H	75H
A001H	65H
A002H	23H
A003H	85H
A004H	65H
A005H	60H

A006H 12H

A007H 01H

A008H 9BH

- **TEST3**
  - **INPUT(**Data at A000H):

    A000H 12H

    A001H 02H

    A002H A2H

    A003H 12H

    A004H 00H

    A005H 00H

  - **OUTPUT**(Program at A000H):

    A000H 12H

    A001H 02H

    A002H A2H

    A003H 12H

    A004H 00H

    A005H 00H

    *This program takes an input and jumps back to 0000H ie restart condition.

- **ANALYSIS:**
    - The expected output and actual output match perfectly.
- **CONCLUSION:**
    - The code works as expected.