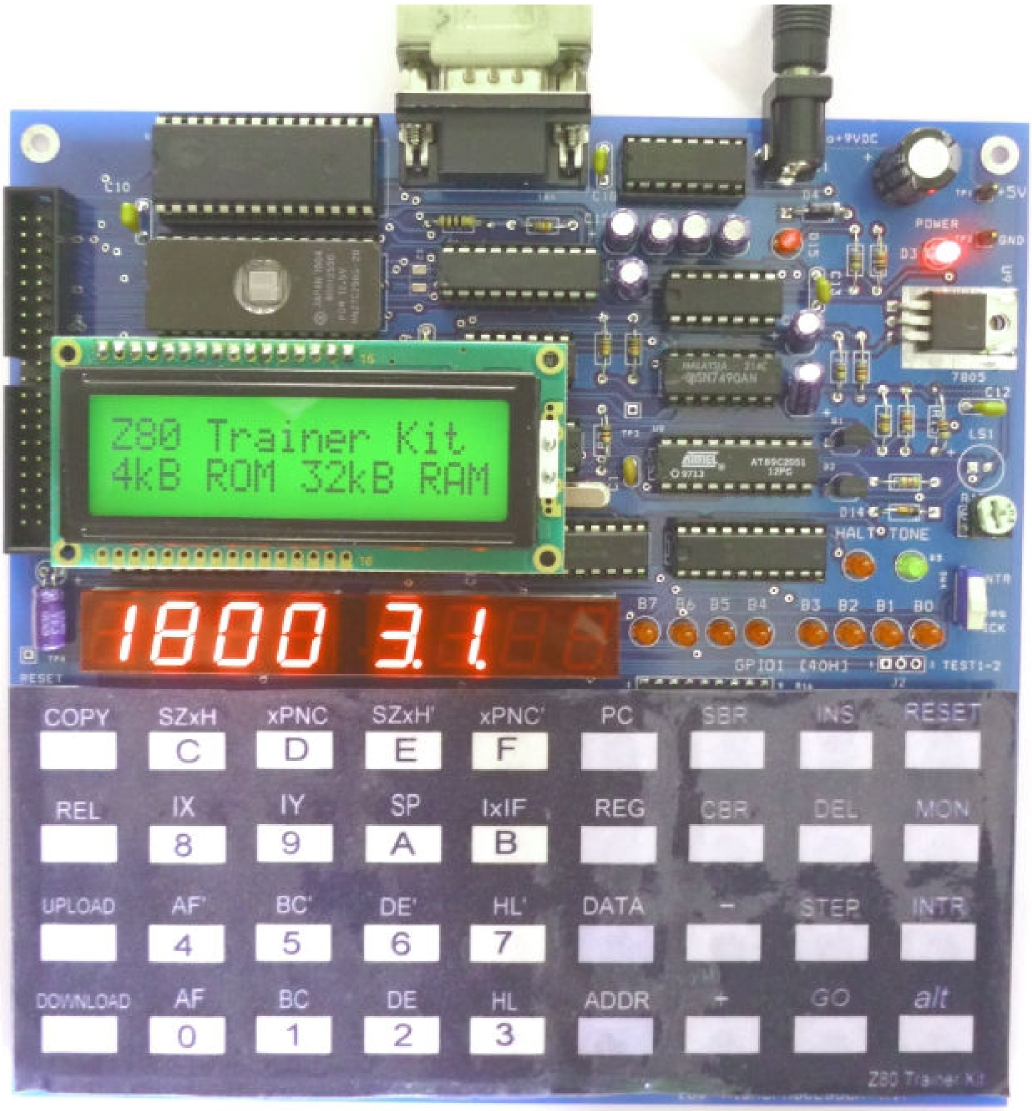
Z80 Microprocessor Kit Programming Lab Book



Student:_____________________________ID:_______________________

Study program:____

PREFACE

This lab book is designed for self-learning how to program the Z80 microprocessor in machine language with the Z80 Microprocessor Kit. The demonstration programs were written in assembly program using Z80 instructions. The program listings are provided with instruction hex code. Students can enter the program by using hex code to the memory and test it directly. Illustrations of program flow were also provided for easy understanding. After program testing the exercise will ask questions and program modification only at the hex code, no need the assembler program.

LAB1 to LAB 6 are focusing on software programming. LAB7 to LAB12 are for hardware interfacing using the on-board I/O devices.

Z80 MICROPROCESSOR KIT
PROGRAMMING LAB BOOK

CONTENTS

LAB 1 RUNNING DOT LED..4

LAB 2 FILL MEMORY WITH CONSTANT...6

LAB 3 ADDING 16-BIT NUMBER...8

LAB 4 ADDING BCD NUMBER..11

LAB 5 COMPARISON...14

LAB 6 STOP-WATCH...17

LAB 7 INTERRUPT..21

LAB 8 7-SEGMENT DISPLAY..26

LAB 9 KEYBOARD..30

LAB 10 DIGITAL TIMER..34

LAB 11 LCD MODULE INTERFACING...37

LAB 12 SERIAL COMMUNICATION..43

APPENDIX A

MONITOR SUBROUTINES..50

APPENDIX B

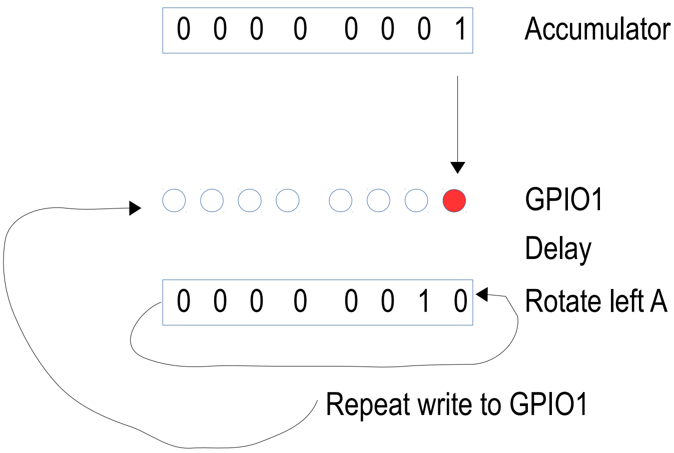
ASCII CODE CHART...51
DECIMAL, BINARY and HEXADECIMAL

LAB 1 RUNNING DOT LED

Your first program will get you familiar with how to enter the hex code and test run quickly.

The main code is repeat writing accumulator register to GPIO1 with delay.

```
line      addr hex code    label          instruction     comment
----      ----   --------    -----          ------------    --------------
0001   1800                                .ORG 1800H
0002   1800
0003   1800              GPIO1            .EQU 40H
0004   1800
0005   1800 3E 01        MAIN             LD A,1
0006   1802
0007   1802 D3 40        LOOP             OUT (GPIO1),A
0008   1804 CD 0C 18                      CALL DELAY
0009   1807 CB 07                         RLC A
0010   1809 C3 02 18                      JP LOOP
0011   180C
0012   180C
0013   180C 11 FF FF     DELAY            LD DE,-1
0014   180F 21 00 10                      LD HL,1000H
0015   1812 19           LOOP2            ADD HL,DE
0016   1813 38 FD                         JR C,LOOP2
0017   1815 C9                            RET
0018   1816
0019   1816                               .END
0020   1816
tasm: Number of errors = 0
```



Procedure

1. Enter the hex code from address 1800 to 1815.

2. Run the program with key PC, and key GO.

3. Stop program running with key RESET.

Exercise

1. Suppose we want new initial value to be loaded into register A, where is the byte to be modified? Try change it and run the program again.

---

2. Let us make running speed slower. Where is the byte to be modified? Test it.

---

3. Can you change the direction of LED running? How to do that?

Another example is to show the useful of delay subroutine for fun.

```
0001   1800                              .ORG 1800H
0002   1800
0003   1800                GPIO1         .EQU 40H
0004   1800                TONE2K        .EQU 05E2H
0005   1800
0006   1800 21 64 00       MAIN          LD HL,100
0007   1803 CD E2 05                     CALL TONE2K
0008   1806 CD 0C 18                     CALL DELAY
0009   1809 C3 00 18                     JP MAIN
0010   180C
0011   180C 11 FF FF       DELAY         LD DE,-1
0012   180F 21 00 10                     LD HL,1000H
0013   1812 19             LOOP2         ADD HL,DE
0014   1813 38 FD                        JR C,LOOP2
0015   1815 C9                           RET
0016   1816
0017   1816                              .END
0018   1816
tasm: Number of errors = 0
```

The new subroutine called TONE2K is built-in code that produces 2kHz tone with period set by HL. The delay subroutine is the same as LED running. Students may try enter the hex code and test run again. You may learn how the code running and modify it.

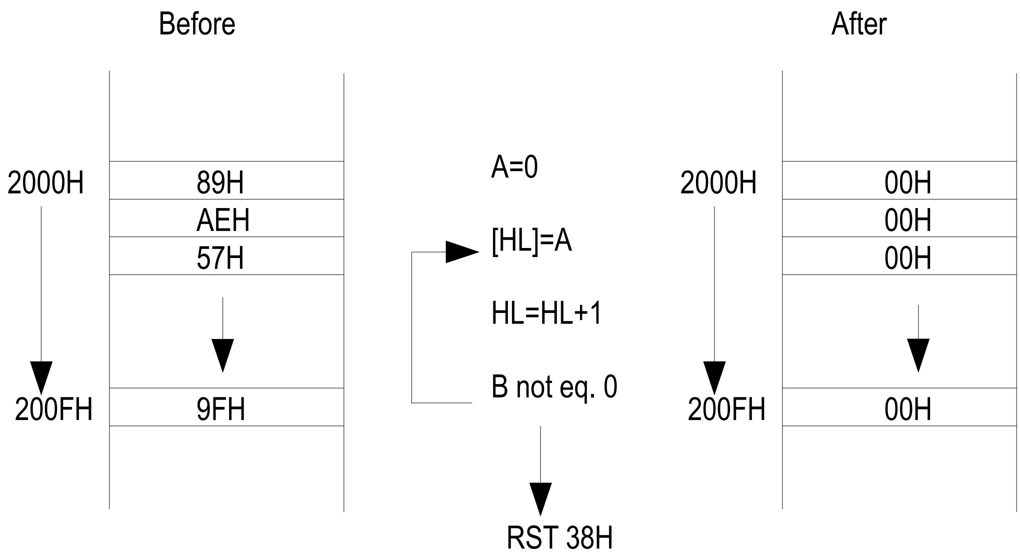
Summary

The delay subroutine is a useful code for demonstration computer program running and even for complicated applications as well. Since the microprocessor cycle time is quite short compare to our response. If students change the load value of HL register to a small number, and try the LED running. We will see all LED will be lit, why?

LAB 2 FILL MEMORY WITH CONSTANT

This lab demonstrates how to fill block of memory using indirect addressing mode.

```
line    addr hex code        label        instruction       comment
----    ----  --------       -----        -------------     ---------------
0001   1800                               .ORG 1800H
0002   1800
0003   1800
0004   1800 06 10            MAIN         LD B,16      ; loop counter
0005   1802 21 00 20                      LD HL,2000H ; HL = 2000H
0006   1805 3E 00                         LD A,0     ; A=0
0007   1807 77               LOOP         LD (HL),A ; [HL]=A
0008   1808 23                            INC HL     ; HL=HL+1
0009   1809 10 FC                         DJNZ LOOP ; jump if B!=0
0010   180B FF                            RST 38H    ; jump back monitor
0011   180C
0012   180C                               .END
0013   180C
tasm: Number of errors = 0
```



Procedure

1. Enter the hex code from address 1800 to 180B.

2. Write down the content of memory locations 2000 to 200F before press key GO.

3. Press key PC to set address to 1800 then key GO to run the program.

4. Write down the content of memory locations 2000 to 200F after press key GO.

Results

Address	Data (before)	Data (after)
2000H		
2001H		
2002H		
2003H		
2004H		
2005H		
2006H		
2007H		
2008H		
2009H		
200AH		
200BH		
200CH		
200DH		
200EH		
200FH		

Exercise

1. Now we want to clear 256 bytes from 2000 to 20FF. How to do that? Try and show the result to TA.

2. Instead of clearing the memory, let us fill the block of data with FF, 256 bytes, from 2000 to 20FF. How to do that? Try and show the result to TA.

Summary

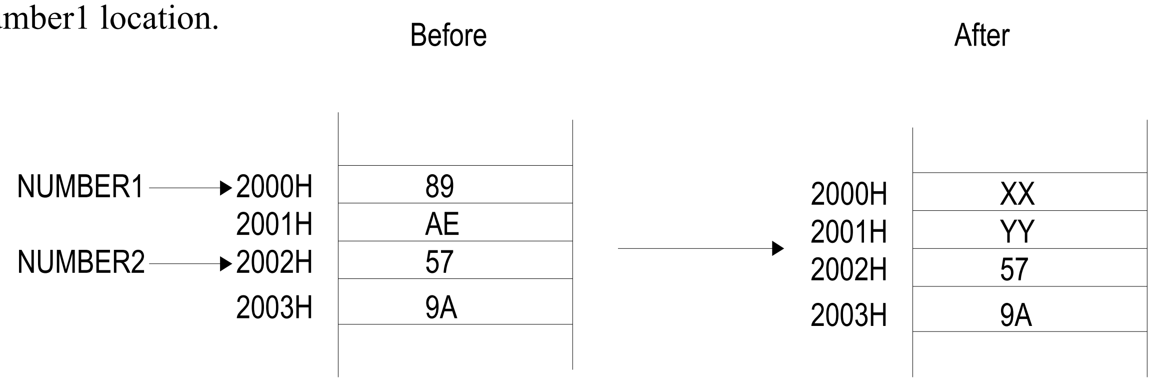
We can use register B together with DJNZ instruction to make loop running. The load value to register B will be loop counter. The body of loop can be any code or subroutine to be repeated. HL register pair forms a 16-bit pointer. We use it to point the memory address with indirect addressing mode.

LAB 3 ADDING 16-BIT BINARY NUMBER

We will learn how to add two 16-bit binary numbers using indirect addressing mode.

```
line      addr hex code        label          instruction       comment
----   ---- --------    -----          ------------   --------------
0001   1800                                .ORG 1800H
0002   1800
0003   1800                NUMBER1     .EQU 2000H
0004   1800                NUMBER2     .EQU 2002H
0005   1800
0006   1800                MAIN
0007   1800 21 00 20                   LD HL,NUMBER1
0008   1803 11 02 20                   LD DE,NUMBER2
0009   1806
0010   1806 06 02                      LD B,2    ; loop counter
0011   1808 AF                         XOR A ; clear carry flag
0012   1809
0013   1809 1A             LOOP        LD A,(DE)
0014   180A 8E                         ADC A,(HL)
0015   180B 77                         LD (HL),A
0016   180C
0017   180C 13                         INC DE
0018   180D 23                         INC HL
0019   180E
0020   180E 10 F9                      DJNZ LOOP
0021   1810
0022   1810 FF                         RST 38H
0023   1811
0024   1811                            .END
0025   1811
tasm: Number of errors = 0
```

Number1 and number2 are memory address that stores two 16-bit data. The program uses HL and DE registers as a pointer. Register B is loop counter. The result will be saved to number1 location.




```
      AE 89+
CY    9A 57
      YY XX
```

Procedure

1. Enter the hex code from address 1800 to 1810.

2. Edit the data to be added for number1 and number2 as shown above.

3. Compute by hand with binary addition, keep the result.

4. Run the program, press key PC and key GO.

5. Write the result that saved in number1 location and carry flag after adding. Compare the result with hand calculation. Carry flag can be viewed with key REG, D. The display will show the low nibble of flag register. Carry flag is the right most bit.

Results

Address	Data (before)	Hand compute	Data (after)
2000H			
2001H			
2002H			
2003H			
Carry Flag=	--		

Exercise

1. Let us try with new value of number1 and number2 by editing them in location 2000H to 2003H. Then ask your friend add it by hand calculation. Check the your friend result with Z80 running, correct or not?

Address	Data (before)	Hand compute	Data (after)
2000H			
2001H			
2002H			
2003H			
Carry Flag=	--		

2. If we want to add two 32-bit number, how to do that?

Summary

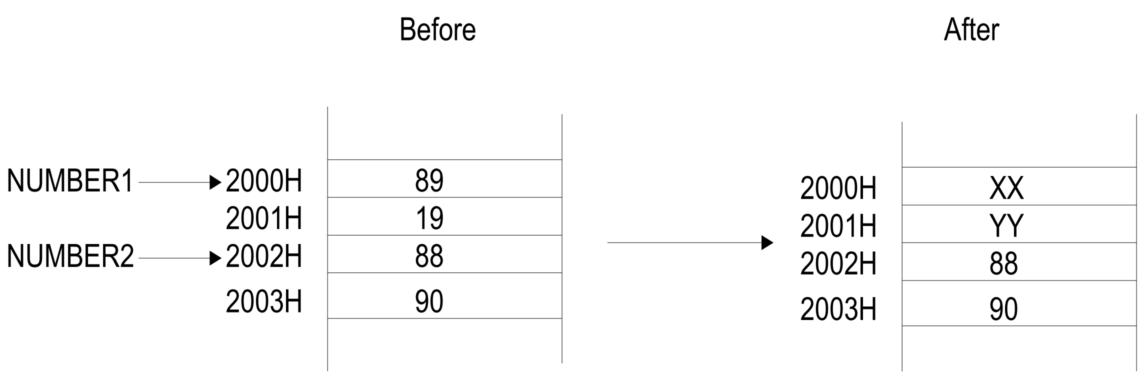
Adding binary number can be done with 8-bit addition instruction. For multi-byte adding, we can use ADC instruction with loop running. The number of loop will be the number of byte to be added. If there is a carry bit from lower significant byte adding, the carry bit will be added to the next higher significant byte automatically.

Important note: adding instruction is for binary number. The hexadecimal representation are for shorter written and code entering.

LAB 4 ADDING BCD NUMBER

For some applications that use BCD number, Z80 also has instruction DAA used to adjust or correct the result of addition in accumulator.

```
line     addr hex code        label         Instruction       comment
----     ----   --------      -----         -------------     ---------------
0001   1800                                 .ORG 1800H
0002   1800
0003   1800                   NUMBER1       .EQU 2000H
0004   1800                   NUMBER2       .EQU 2002H
0005   1800
0006   1800                   MAIN
0007   1800 21 00 20                        LD HL,NUMBER1
0008   1803 11 02 20                        LD DE,NUMBER2
0009   1806
0010   1806 06 02                           LD B,2
0011   1808 AF                              XOR A  ; clear carry flag
0012   1809
0013   1809 1A                LOOP          LD A,(DE)
0014   180A 8E                              ADC A,(HL)  ; Binary add
0015   180B 27                              DAA        ; adjust result to BCD
0016   180C 77                              LD (HL),A
0017   180D
0018   180D 13                              INC DE
0019   180E 23                              INC HL
0020   180F
0021   180F 10 F8                           DJNZ LOOP
0022   1811
0023   1811 FF                              RST 38H
0024   1812
0025   1812                                 .END
0026   1812
tasm: Number of errors = 0
```




```
     19 89+
CY   90 88
     YY XX
```

We see that above program is the same as LAB 3. Only, DAA instruction is followed with ADC instruction. For now number1 and number2 must be BCD, all digits must be 0-9 only. As we know that BCD is a decimal representation by 4-bit binary number.

Procedure

1. Enter the hex code from address 1800 to 1811.

2. Edit the data to be added for number1 and number2 as shown above.

3. Compute by hand, keep the result.

4. Run the program, press key PC and key GO.

5. Write the result that saved in number1 location and carry flag after adding. Compare the result with hand calculation. Carry flag can be viewed with key REG, D. The display will show the low nibble of flag register. Carry flag is the right most bit.

Results

Address	Data (before)	Hand compute	Data (after)
2000H			
2001H			
2002H			
2003H			
Carry Flag=	--		

Exercise

1. Let us try with new value of number1 and number2 by editing them in location 2000H to 2003H. Then ask your friend add it by hand calculation. Check the your friend result with Z80 running, correct or not?

Address	Data (before)	Hand compute	Data (after)
2000H			
2001H			
2002H			
2003H			
Carry Flag=	--		

2. Suppose we want to add 10 digits BCD number. For example,

Number1 = 1234567890
Number2 = 8976543254

Find the unknown values, xx, yy, jj, kk and nn in the program below.

```
addr hex code    label      Instruction    comment
---- --------    -----      ------------   --------------
1800                        .ORG 1800H
1800
1800
1800             MAIN
1800 21 xx yy               LD HL,NUMBER1
1803 11 jj kk               LD DE,NUMBER2
1806
1806 06 nn                  LD B,nn
1808 3F                     CCF
1809
1809 1A          LOOP       LD A,(DE)
180A 8E                     ADC A,(HL)
180B 27                     DAA      ; Adjust to BCD
180C 77                     LD (HL),A
180D
180D 13                     INC DE
180E 23                     INC HL
180F
180F 10 F8                  DJNZ LOOP
1811
1811 FF                     RST 38H
1812
1812                   .END
```

3. Test your program running by checking the result of addition.

Summary

Z80 provides DAA instruction for correcting the result after using ADD or ADC instructions. DAA must be placed after ADD or ADC instruction. The values to be added also must be BCD number, 0-9. DAA is not the instruction that converts binary to decimal number. It is designed for correcting the result of BCD number adding.

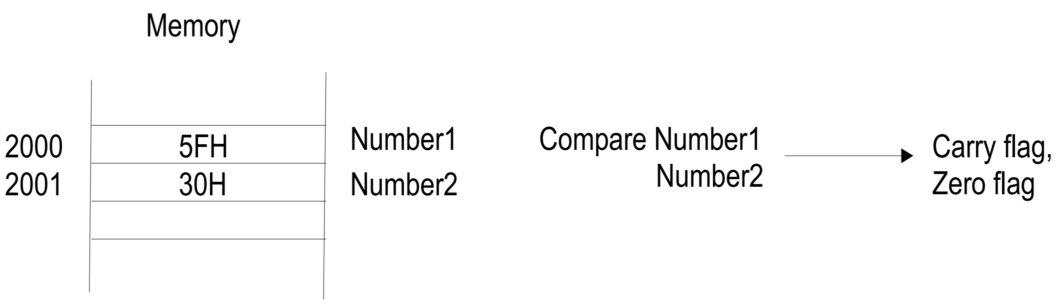
LAB 5 COMPARISON

Number comparison can be used to control flow or decision of program running. Z80 provides 8-bit compare instruction. The carry flag and zero flag will be affected according to the result of comparison.

Suppose we have two 8-bit number to be compared, A and B. The result would be,

1. A < B, A is less than B,
2. A = B, A is equal to B,
3. A > B, A is larger than B.

```
1800                                  .ORG 1800H
1800
1800                    NUMBER1       .EQU 2000H ; A
1800                    NUMBER2       .EQU 2001H ; B
1800
1800                    MAIN
1800 3A 00 20                         LD A,(NUMBER1)
1803 21 01 20                         LD HL,NUMBER2
1806 BE                               CP (HL)      ; A-B
1807 FF                               RST 38H
1808
1808                                  .END
1808
tasm: Number of errors = 0
```



Procedure

1. Enter the hex code from address 1800 to 1807.

2. Edit the data to be compared for number1 and number2 at location 2000 and 2001.

4. Run the program, press key PC and key GO. Write down the result of carry and zero flags with key REG, C, or REG D.

Results table

2000H (A)	2001H (B)	Zero flag	Carry flag	comment
2AH	2FH			CASE 1: A < B
2FH	2FH			CASE 2:A = B
35H	2FH			CASE 3: A > B

Check your results with below table.

2000H (A)	2001H (B)	Zero flag	Carry flag	comment
2AH	2FH	0	1	CASE 1: A < B
2FH	2FH	1	0	CASE 2: A = B
35H	2FH	0	0	CASE 3: A > B

We may combine the result for A<=B, Zero and carry flags will be set.

For A > = B, Zero flag will be set and carry will be cleared.

We can use instructions conditional jump after carry or zero flags are affected. So the decision can be made. The example below shows how to use the conditional jump.

```
1800                                .ORG 1800H
1800
1800                    GPIO1       .EQU 40H
1800                    NUMBER1     .EQU 2000H
1800                    NUMBER2     .EQU 2001H
1800
1800                    MAIN
1800 3A 00 20                       LD A,(NUMBER1)
1803 21 01 20                       LD HL,NUMBER2
1806 BE                             CP (HL)
1807 DA 12 18                       JP C,LT
180A
180A CA 17 18                       JP Z,LTEQ
180D
180D 3E 03              GT          LD A,3
180F D3 40                          OUT (GPIO1),A ; CASE 3
1811 FF                             RST 38H
1812
1812 3E 01              LT          LD A,1
1814 D3 40                          OUT (GPIO1),A ; CASE 1
1816 FF                             RST 38H
1817
1817 3E 02              LTEQ        LD A,2
```

```
0024   1819 D3 40                     OUT (GPIO1),A ; CASE 2
0025   181B FF                        RST 38H
0026   181C
0027   181C                           .END
0028   181C
tasm: Number of errors = 0
```

Procedure

1. Enter the hex code from address 1800 to 181B.

2. Edit the data to be compared for number1 and number2 at location 2000 and 2001.

4. Run the program, press key PC and key GO. Write down the result of GPIO1 LED for three cases (1 for LED lit, 0 for LED off)

2000H (A)	2001H (B)	GPIO1 LED	comment
2AH	2FH		CASE 1: A < B
2FH	2FH		CASE 2: A = B
35H	2FH		CASE 3: A > B

Summary

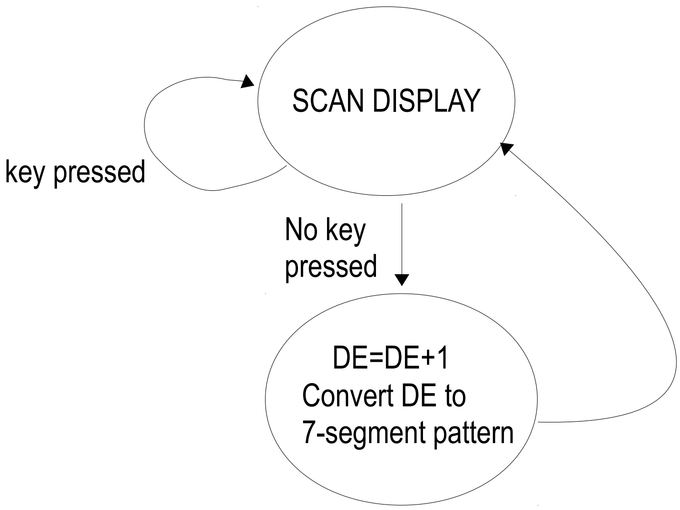
Comparison instruction can be used to make decision of program flow. After two numbers are compared, the zero and carry flags will be affected. Program flow can be controlled with conditional jump instructions.

LAB 6 STOP-WATCH

This lab demonstrates using monitor subroutines to make a simple stop-watch. The timebase 1/100 second is done by display scanning loop period. For more accurate, we will use timer interrupt for the next lab.

```
0001   1800                                      .ORG 1800H
0002   1800
0003   1800                    BUFFER            .EQU 2000H
0004   1800                    SCAN1             .EQU 0624H
0005   1800                    HEX7SEG           .EQU 0678H
0006   1800
0007   1800 DD 21 00 20        MAIN              LD IX,BUFFER
0008   1804 11 00 00                             LD DE,0
0009   1807
0010   1807 CD 24 06           LOOP              CALL SCAN1
0011   180A 30 FB                                JR NC,LOOP
0012   180C
0013   180C 7B                                   LD A,E
0014   180D C6 01                                ADD A,1
0015   180F 27                                   DAA
0016   1810 5F                                   LD E,A
0017   1811 7A                                   LD A,D
0018   1812 CE 00                                ADC A,0
0019   1814 27                                   DAA
0020   1815 57                                   LD D,A
0021   1816 7B                                   LD A,E
0022   1817
0023   1817 21 00 20                             LD HL,BUFFER
0024   181A CD 78 06                             CALL HEX7SEG
0025   181D 36 02                                LD (HL),2
0026   181F 23                                   INC HL
0027   1820 7A                                   LD A,D
0028   1821 CD 78 06                             CALL HEX7SEG
0029   1824 36 00                                LD (HL),0
0030   1826
0031   1826 C3 07 18                             JP LOOP
0032   1829
0033   1829                                      .END
0034   1829
tasm: Number of errors = 0
```

Register pair DE holds SEC and SEC/100. We see that incrementing was done by two bytes adding with DAA for BCD adjustment. Result of BCD addition will be 00 to 99 for both register D and E. When user press any key, the carry flag will be cleared, so jump instruction at line 11 will repeat scanning without incrementing DE. This makes as a STOP key. When key was released, carry flag will be set, the incrementing will be continued.



Procedure

1. Enter the hex code from address 1800 to 1828.

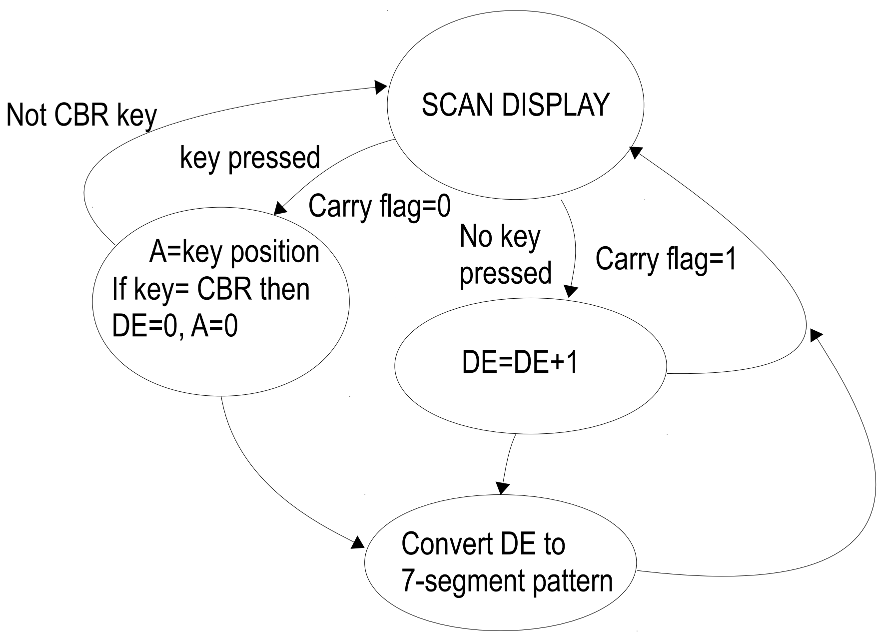
2. Run the program, press key PC and key GO.

3. Try press any key, it will stop counting.

Suppose we want to clear the count to zero with a given key. How to do that?

```
1800                                .ORG 1800H
1800
1800                 BUFFER         .EQU 2000H
1800                 SCAN1          .EQU 0624H
1800                 HEX7SEG        .EQU 0678H
1800
1800 DD 21 00 20     MAIN           LD IX,BUFFER
1804 11 00 00                       LD DE,0
1807
1807 CD 24 06        LOOP           CALL SCAN1
180A 38 0B                          JR C, SKIP
180C FE 18                          CP 18H            ; KEY CBR=18H
180E 20 F7                          JR NZ,LOOP
1810
1810 11 00 00                       LD DE,0
1813 3E 00                          LD A,0
1815 18 09                          JR SKIP2
1817
1817 7B              SKIP           LD A,E
1818 C6 01                          ADD A,1
181A 27                             DAA
181B 5F                             LD E,A
181C 7A                             LD A,D
181D CE 00                          ADC A,0
```

```
0025   181F 27                          DAA
0026   1820
0027   1820 57            SKIP2         LD D,A
0028   1821 7B                          LD A,E
0029   1822
0030   1822 21 00 20                    LD HL,BUFFER
0031   1825 CD 78 06                    CALL HEX7SEG
0032   1828 36 02                       LD (HL),2
0033   182A 23                          INC HL
0034   182B 7A                          LD A,D
0035   182C CD 78 06                    CALL HEX7SEG
0036   182F 36 00                       LD (HL),0
0037   1831
0038   1831 C3 07 18                    JP LOOP
0039   1834
0040   1834                             .END
0041   1834
tasm: Number of errors = 0
```



Since SCAN1 subroutine also scans the keyboard. It returns hardware wiring position of the key being pressed. We will learn how to use SCAN1 more in later LAB. Let us try detecting key CBR having position key of 18H to be a reset counter key. We use compare instruction to check if key CBR has been pressed or not. It is was pressed, register DE and A are cleared to zero.

Procedure

1. Enter the hex code from address 1800 to 1833.

2. Run the program, press key PC and key GO.

3. Try press any key, it will stop counting.

4. Try press key CBR, did you see the counter is zero?

Exercise

1. We can try change the control key to a given key. The example uses CBR for resetting the counter to zero. Try to change another key for stop function.

2. Have the real stop watch for checking time error, with the accumulating time for one minute, our program is run faster or slower in seconds unit? Why?

3. Can you change the counting from incrementing to decrementing using SUB and SBC instructions.

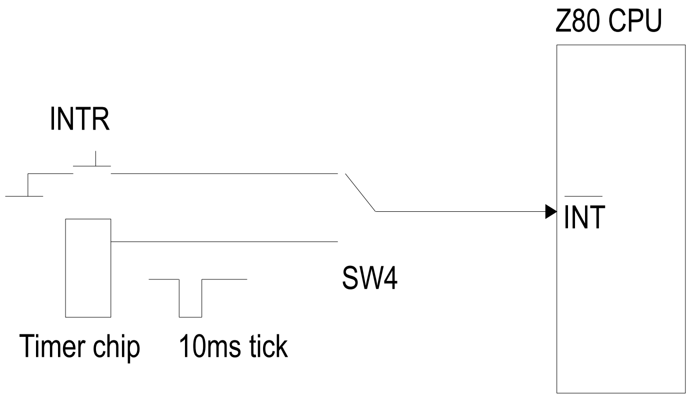
Summary

We can use monitor subroutine scan display for displaying counter running. The count is updated every approx. 1/100 sec. Registers DE was used for four digits BCD counting. SCAN1 subroutine also returns hardware wiring position of the key. We can have a given key by detecting its position for the STOP function.

LAB 7 INTERRUPT

Hardware interrupt

We will learn a maskable interrupt process and write the code for testing interrupt. The Z80 Kit has two sources of hardware interrupt. One for INTR key and a 10ms tick signal produced by timer chip, AT89C2051 microcontroller. SW4 is a slide switch. We can select which one will be the interrupt source.

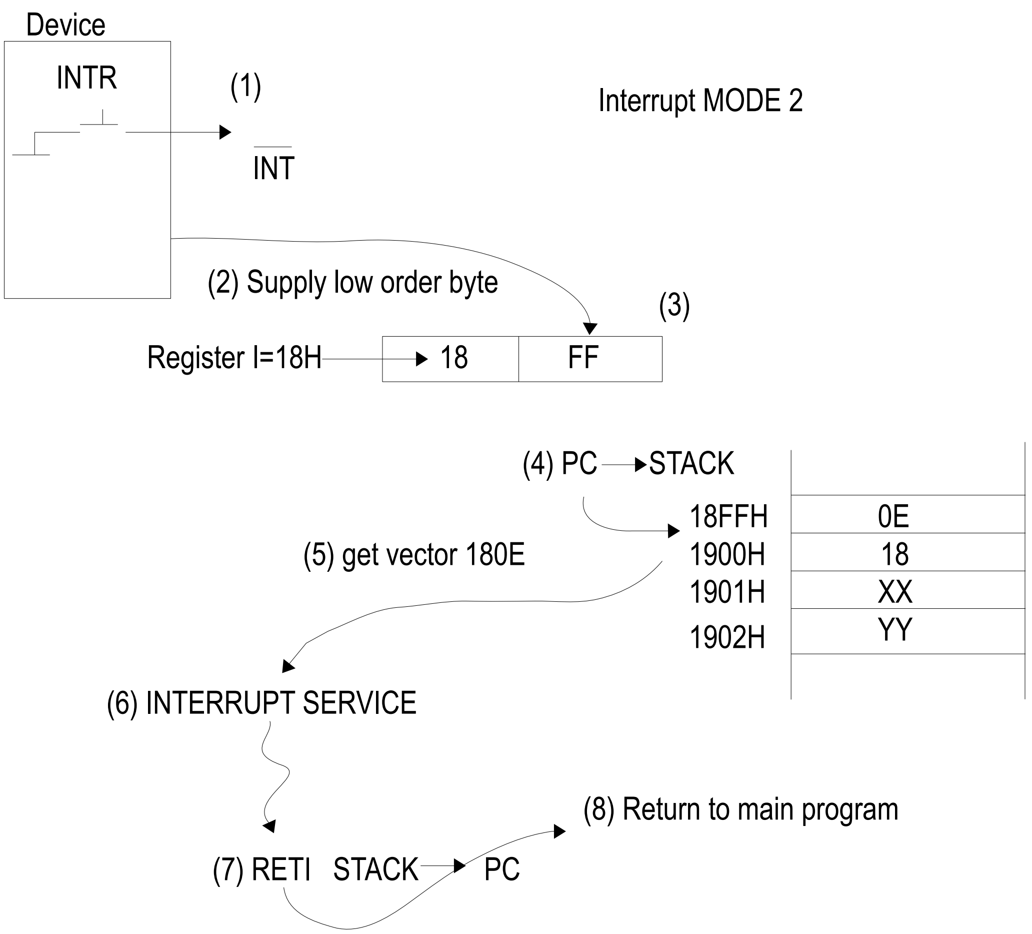



```
1800                        .ORG 1800H
1800
1800             ADDISP     .EQU 1FDEH
1800             PORT2      .EQU 02H
1800             GPIO1      .EQU 40H
1800
1800       ; main code
1800
1800 21 0E 18    MAIN       LD HL,SERVICE
1803 22 FF 18               LD (18FFH),HL ; insert vector
1806 3E 18                  LD A,18H
1808 ED 47                  LD I,A     ; high order byte
180A
180A ED 5E                  IM 2
180C FB                     EI
180D
180D FF                     RST 38H
180E
180E       ; interrupt service subroutine
180E
180E E5          SERVICE    PUSH HL
180F F5                     PUSH AF
1810
1810 AF                     XOR A
1811 D3 02                  OUT (PORT2),A
1813 2A DE 1F               LD HL,(ADDISP)
1816 7E                     LD A,(HL)
```

```
0028  1817 D3 40                    OUT (GPIO1),A
0029  1819
0030  1819 F1                       POP AF
0031  181A E1                       POP HL
0032  181B
0033  181B FB                       EI
0034  181C ED 4D                    RETI
0035  181E
0036  181E                          .END
0037  181E
tasm: Number of errors = 0
```

Above program demonstrates how the maskable interrupt mode 2 is responding to the trigger signal by key INTR.

When Z80 was triggered by low level logic at INT pin, it will acknowledge by requesting the low order byte from device. The memory address that stores service vector is formed by register I for high byte and low order byte from the device. Z80 will save current program counter to stack memory, get the vector address then jump to interrupt service routine. When interrupt process was finished, the program counter that saved in stack memory will be retrieved and then return to the main program.



Our Kit has no circuit that supplies the low order byte, however with the pull-up resistor at the data bus, the CPU will read byte as FF. We then load the register I with 18H, thus the location that stores interrupt vector will be 18FFH.

Procedure

1. Slide SW4 to select INTR key. The INTR key will be used as the interrupt source.

2. Enter the hex code from address 1800 to 181D.

3. Press RESET key, PC key. Try press INTR key. Did you see any change at GPIO1?

____

4. Now press key RESET, PC then GO. Did you see any change?

____

5. Press key INTR again, what value will be displayed on GPIO1?

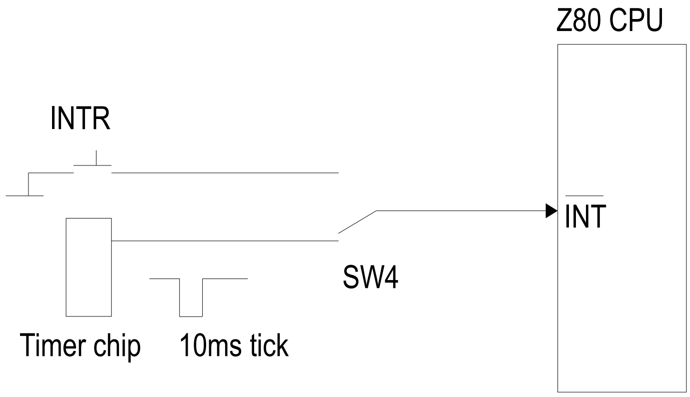
____

6. Change address to 0000, press key INTR again, write down the hex content and draw GPIO1 LED. Press key + for next address and key INTR to make interrupt..

Address	Data (HEX)	GPIO1 LED (binary)
0000		
0001		
0002		
0003		
0004		
0005		
0006		
0007		
0008		
0009		
000A		
000B		
000C		
000D		
000E		
000F		

10ms Tick Timer interrupt

Let us do experiment with 10ms tick using timer chip. The 89C2051 microcontroller is used to produce 10ms tick signal. Slide SW4 to 10ms tick position, we can trigger the Z80 CPU to jump to interrupt service routine every 10ms. The demonstration program will display BCD counting at GPIO1 every one second while the monitor program is functioning.




```
1800                            .ORG 1800H
1800
1800                GPIO1       .EQU 40H
1800                TICK        .EQU 2000H
1800                COUNTER     .EQU 2001H
1800
1800        ; main code
1800
1800 21 0E 18       MAIN        LD HL,SERVICE
1803 22 FF 18                   LD (18FFH),HL ; insert vector
1806 3E 18                      LD A,18H
1808 ED 47                      LD I,A      ; high order byte
180A
180A ED 5E                      IM 2
180C FB                         EI
180D
180D FF                         RST 38H
180E
180E        ; interrupt service subroutine
180E
180E E5             SERVICE     PUSH HL
180F F5                         PUSH AF
1810
1810 21 00 20                   LD HL,TICK
1813 34                         INC (HL)
1814 3A 00 20                   LD A,(TICK)
1817 FE 64                      CP 100
1819 38 10                      JR C,SKIP
181B 3E 00                      LD A,0
181D 32 00 20                   LD (TICK),A
```

```
0031   1820
0032   1820 3A 01 20                          LD A,(COUNTER)
0033   1823 C6 01                             ADD A,1
0034   1825 27                                DAA
0035   1826 32 01 20                          LD (COUNTER),A
0036   1829 D3 40                             OUT (GPIO1),A
0037   182B
0038   182B F1              SKIP              POP AF
0039   182C E1                                POP HL
0040   182D
0041   182D FB                                EI
0042   182E ED 4D                             RETI
0043   1830
0044   1830                                   .END
0045   1830
tasm: Number of errors = 0
```

Main code is the same as hardware interrupt. When Z80 is triggered by logic low signal at pin INT, it will get the vector at location 18FFH, where we insert the interrupt service address 180EH. The same as hardware interrupt, but now we use 10ms tick as the interrupt source.

The code for interrupt service may called timer interrupt, will increment the tick variable, if it is equal or greater than 100, then increment the counter variable.

Procedure

1. Slide SW4 to select TICK. 10ms Tick will be used as the interrupt source.

2. Enter the hex code from address 1800 to 182F.

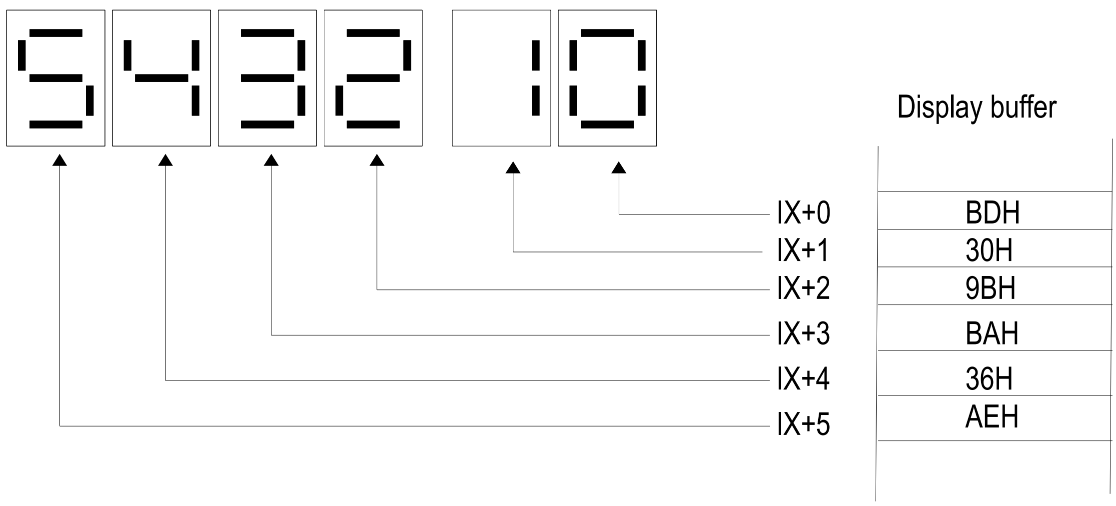
3. Press RESET key, PC key. Then key GO. Did you see any change at GPIO1? Explain.

---

4. Try press key ADDR, + or – to change the address. Can you hear the beep sound that slightly change in frequency? Explain why?

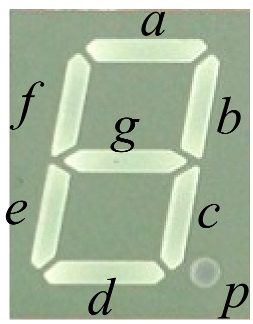
LAB 8 7-SEGMENT DISPLAY

The kit has 6-digit 7-segment LED display. We will learn how to use it for displaying hex or BCD data.



Display buffer uses 6 bytes of RAM for storing the data to be scanned. We can use IX register to point to buffer memory. Example above shows the rightmost digit pattern, BDH for number 0, and the left most digit pattern, AEH for number 5.

Bit pattern for LED display and bit position for each segment is shown below. For example number 1, segment b and c are logic one , the data will be 30H.



d p c b	*a f g e*

8-bit segment designation, e.g. 30H for '1'

Let us try this program for testing the display.

```
1800                              .ORG 1800H
1800
1800                 SCAN         .EQU 0624H
1800
1800        ; main code
1800
1800 DD 21 00 20     MAIN         LD IX,DISPLAYBUF
1804 CD 24 06        LOOP         CALL SCAN
1807 18 FB                        JR LOOP
1809
1809
2000                              .ORG 2000H
2000
2000                 DISPLAYBUF   .BLOCK 6
2006
2006                              .END
2006
tasm: Number of errors = 0
```

We use IX register for pointing the display buffer memory begin at 2000H. And then call the subroutine that scans display.

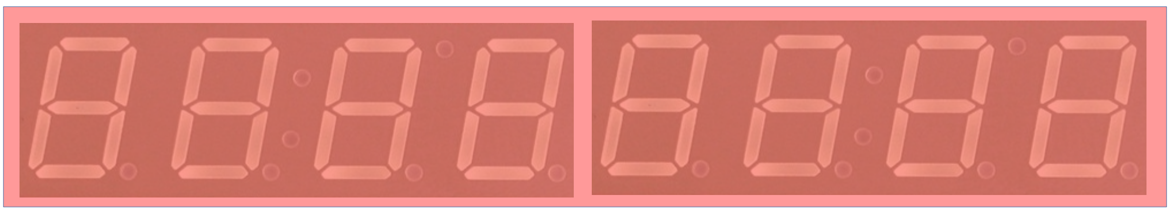
Procedure

1. Enter the hex code from address 1800 to 1808.

2. Press PC and GO, Did you see any display on the 7-segment LED? How it looks like?

---

3. Edit the memory address 2000 to 2005 with these value, 30, 02, 02, 0F, 1F, A1. Press PC and GO. Draw the pattern on the segment below. Try also your pattern.



Exercise

1. Edit the bit pattern in display buffer, then show TA with the your name on the LED display.

Now let us try display the 16-bit data and 8-bit data using monitor subroutines.

```
1800                                    .ORG 1800H
1800
1800                    SCANDISP        .EQU 0624H
1800                    DATADISP        .EQU 0671H  ; A=nn
1800                    ADDRDISP        .EQU 0665H  ; DE=nnnn
1800                    DISPLAYBUF      .EQU 1FB6H
1800
1800          ; main code
1800
1800 3A 00 20           MAIN            LD A,(2000H)
1803 CD 71 06                           CALL DATADISP
1806
1806 ED 5B 01 20                        LD DE,(2001H)
180A CD 65 06                           CALL ADDRDISP
180D
180D DD 21 B6 1F                        LD IX,DISPLAYBUF
1811 CD 24 06           LOOP            CALL SCANDISP
1814 18 FB                              JR LOOP
1816
1816                                    .END
1816
tasm: Number of errors = 0
```

Subroutine DATADISP will convert the 8-bit data in register A to two digits 7-segment pattern and save to display buffer memory at DATA field.

Subroutine ADDRDISP will convert the 16-bit data in register DE to four digits 7-segment pattern and save to display buffer memory at ADDRESS field.

We use memory address 2000H for storing 8-bit data and 2001H-2002H for 16-bit data.

Procedure

1. Enter the hex code from address 1800 to 1815.

2. Edit memory contents at location 2000H to 2002H with 12H, 34H and 56H.

3. Press PC and GO. Did you see the number on the display?

Summary

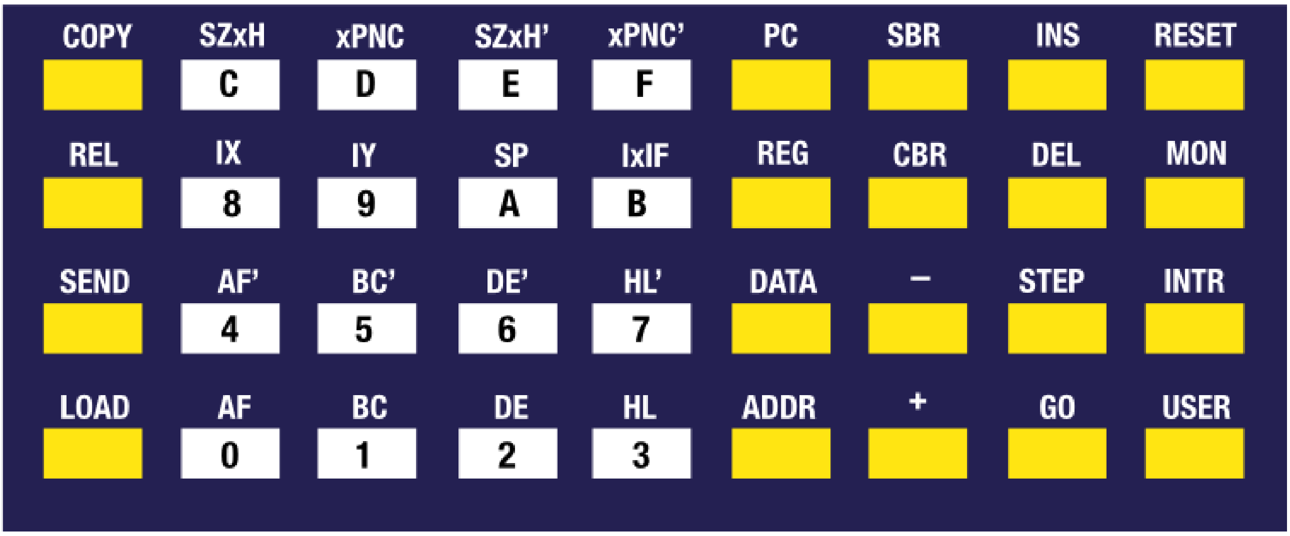
The onboard 6-digit LED can be used to display 8-bit or 16-bit data. Each digit uses one byte memory which contains the pattern for a given number to be displayed. Z80 kit provides monitor subroutines for use with applications program by calling them with proper preset registers.

The example of code for displaying number and A to Z letters. The code is hex number.

Letter	CODE (hex)	Letter	CODE (hex)
0	BD	I	89
1	30	J	B1
2	9B	K	97
3	BA	L	85
4	36	M	2B
5	AE	N	23
6	AF	O	A3
7	38	P	1F
8	BF	Q	3E
9	BE	R	03
A	3F	S	A6
B	A7	T	87
C	8D	U	B5
D	B3	V	B7
E	8F	W	A9
F	0F	X	07
G	AD	Y	B6
H	37	Z	8A

LAB 9 KEYBOARD

The Z80 kit has 36 keys for hex numbers, function keys and CPU control keys. We can do experiments that demonstrates the use of keyboard. The monitor subroutine for keyboard scanning is also provided for program testing.



Each key has its position by hardware wiring. But to use it with applications program, we may arrange it to provide proper code that suitable for logical programming. For example after key scanning, we will get key position code 12H for hex key '0'. We then can change it to binary data 0 for key '0' by table indexing.

Let us do the first experiment to find the position code for each key.

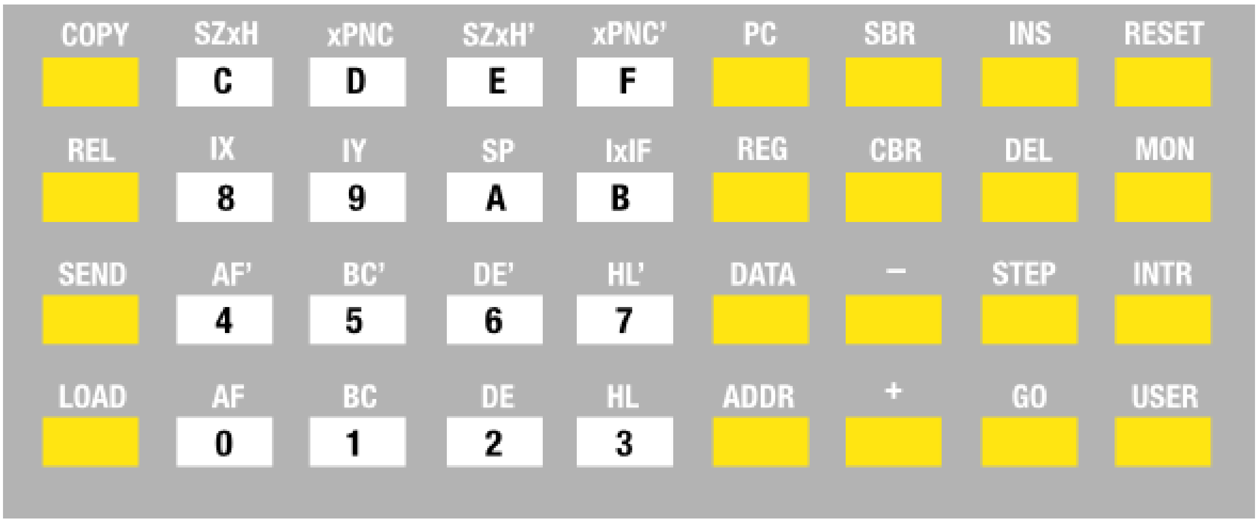
```
1800                        .ORG 1800H
1800
1800            SCAN1       .EQU 0624H
1800            HEX7SEG     .EQU 0678H
1800
1800 DD 21 13 18 MAIN       LD IX,BUFFER
1804
1804 CD 24 06   LOOP        CALL SCAN1
1807 30 02                  JR NC,SKIP
1809 3E FF                  LD A,0FFH
180B
180B 21 13 18   SKIP        LD HL,BUFFER
180E CD 78 06               CALL HEX7SEG
1811 18 F1                  JR LOOP
1813
1813 00         BUFFER      .BYTE 0
1814 00                     .BYTE 0
1815 00                     .BYTE 0
1816 00                     .BYTE 0
1817 00                     .BYTE 0
1818 00                     .BYTE 0
```

```
1819
1819                             .END
1819
tasm: Number of errors = 0
```

Procedure

1. Enter the hex code from address 1800 to 1812.

2. Press PC and GO, write down the position key in the box or close to the key when the key was pressed (except CPU control and user keys).



To make the key position useful and proper functioning, we change it to internal code then.

The monitor subroutine SCAN, the upper level subroutine uses KEYTABLE with offset byte (key position) for pointing to the internal code. The IX register will load start address of the table.

For example position code is 0. It will be used as the offset byte. The instruction LD A,(IX+0) will get value of 03 for hex key 3.

```
                KEYTABLE:
077B 03         K0              .BYTE   03H     ;HEX_3
077C 07         K1              .BYTE   07H     ;HEX_7
077D 0B         K2              .BYTE   0BH     ;HEX_B
077E 0F         K3              .BYTE   0FH     ;HEX_F
077F 20         K4              .BYTE   20H     ;N/A
0780 21         K5              .BYTE   21H     ;N/A
0781 02         K6              .BYTE   02H     ;HEX_2
0782 06         K7              .BYTE   06H     ;HEX_6
0783 0A         K8              .BYTE   0AH     ;HEX_A
```

```
0784 0E      K9           .BYTE  0EH   ;HEX_E
0785 22      K0A          .BYTE  22H   ;N/A
0786 23      K0B          .BYTE  23H   ;N/A
0787 01      K0C          .BYTE  01H   ;HEX_1
0788 05      K0D          .BYTE  05H   ;HEX_5
0789 09      K0E          .BYTE  09H   ;HEX_9
078A 0D      K0F          .BYTE  0DH   ;HEX_D
078B 13      K10          .BYTE  13H   ;STEP
078C 1F      K11          .BYTE  1FH   ;DOWNLOAD
078D 00      K12          .BYTE  00H   ;HEX_0
078E 04      K13          .BYTE  04H   ;HEX_4
078F 08      K14          .BYTE  08H   ;HEX_8
0790 0C      K15          .BYTE  0CH   ;HEX_C
0791 12      K16          .BYTE  12H   ;GO
0792 1E      K17          .BYTE  1EH   ;UPLOAD
0793 1A      K18          .BYTE  1AH   ;CBR
0794 18      K19          .BYTE  18H   ;PC
0795 1B      K1A          .BYTE  1BH   ;REG
0796 19      K1B          .BYTE  19H   ;ADDR
0797 17      K1C          .BYTE  17H   ;DEL
0798 1D      K1D          .BYTE  1DH   ;RELA
0799 15      K1E          .BYTE  15H   ;SBR
079A 11      K1F          .BYTE  11H   ;-
079B 14      K20          .BYTE  14H   ;DATA
079C 10      K21          .BYTE  10H   ;+
079D 16      K22          .BYTE  16H   ;INS
079E 1C      K23          .BYTE  1CH   ;COPY
```

Let us try the second experiment to find the internal code for each key.

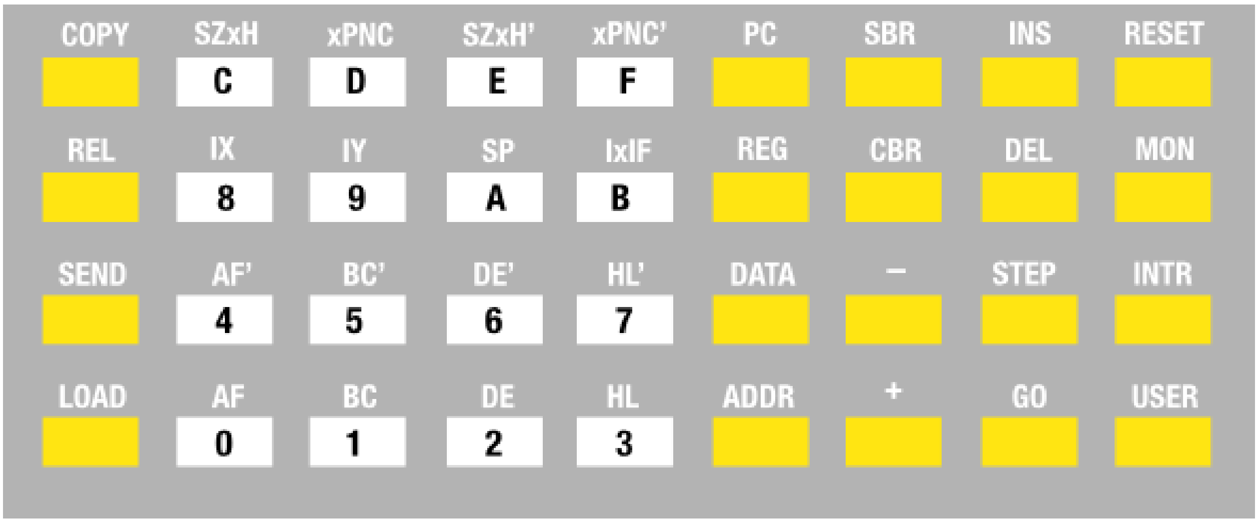
```
1800                             .ORG 1800H
1800
1800                 SCAN        .EQU 05FEH
1800                 HEX7SEG     .EQU 0678H
1800
1800 DD 21 0F 18     MAIN        LD IX,BUFFER
1804
1804 CD FE 05        LOOP        CALL SCAN
1807
1807 21 0F 18        SKIP        LD HL,BUFFER
180A CD 78 06                    CALL HEX7SEG
180D 18 F5                       JR LOOP
180F
180F 00              BUFFER      .BYTE 0
1810 00                          .BYTE 0
1811 00                          .BYTE 0
1812 00                          .BYTE 0
1813 00                          .BYTE 0
```

```
0019   1814 00                        .BYTE 0
0020   1815
0021   1815                           .END
0022   1815
tasm: Number of errors = 0
```

Procedure

1. Enter the hex code from address 1800 to 1814.

2. Press PC and GO, write down the internal key in the box or close to the key when the key was pressed (except CPU control and user keys).



Summary

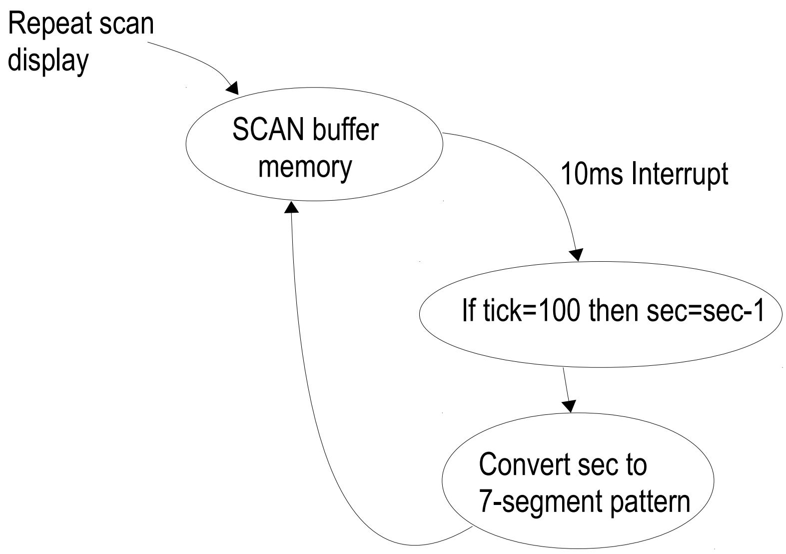
The Z80 kit has keyboard for inputting hex number or set the functions. The monitor subroutine scans the key switch in sequential manner. When it was pressed, the low level subroutine, SCAN1 will return hardware wiring position code. The upper level SCAN subroutine returns the internal code that suitable for logical programming. Students may try using the keyboard for many applications.

LAB 10 DIGITAL TIMER

This lab will use timer interrupt to be timebase for making a digital timer.

```
1800                                .ORG 1800H
1800
1800                   SCAN          .EQU 05FEH
1800                   DATADISP      .EQU 0671H  ; A=nn
1800                   ADDRDISP      .EQU 0665H  ; DE=nnnn
1800                   DISPLAYBUF    .EQU 1FB6H
1800
1800                   TICK          .EQU 2000H
1800                   SEC            .EQU 2001H
1800
1800
1800
1800          ; main code
1800
1800 21 20 18          MAIN         LD HL,SERVICE
1803 22 FF 18                       LD (18FFH),HL ; insert vector
1806 3E 18                          LD A,18H
1808 ED 47                          LD I,A      ; high order byte
180A
180A ED 5E                          IM 2
180C FB                             EI
180D
180D          ; CLEAR DISPLAY BUFFER
180D
180D 21 B6 1F                       LD HL,DISPLAYBUF
1810 06 06                          LD B,6
1812 36 00             CLEAR        LD (HL),0
1814 23                             INC HL
1815 10 FB                          DJNZ CLEAR
1817
1817 DD 21 B6 1F                    LD IX,DISPLAYBUF
181B CD FE 05          LOOP         CALL SCAN
181E 18 FB                          JR LOOP
1820
1820
1820          ; interrupt service subroutine
1820
1820 E5                SERVICE      PUSH HL
1821 F5                             PUSH AF
1822
1822 21 00 20                       LD HL,TICK
1825 34                             INC (HL)
1826 3A 00 20                       LD A,(TICK)
1829 FE 64                          CP 100
182B 38 11                          JR C,SKIP
182D 3E 00                          LD A,0
182F 32 00 20                       LD (TICK),A
1832
```

```
1832
1832 3A 01 20                    LD A,(SEC)
1835 D6 01                       SUB 1
1837 27                          DAA
1838 32 01 20                    LD (SEC),A
183B
183B CD 71 06                    CALL DATADISP
183E
183E F1            SKIP          POP AF
183F E1                          POP HL
1840
1840 FB                          EI
1841
1841 ED 4D                       RETI
1843
1843
1843
1843                             .END
1843
tasm: Number of errors = 0
```



We use interrupt mode 2 and have the interrupt service routine now with timer function. Every 10ms, the CPU will jump to interrupt service routine at location 1820H. The tick variable will be incremented by one and checked if it was equal or larger than 100, i.e. 100x10ms = 1 second, the variable sec will be decremented by one. Again we use DAA to adjust the result from subtraction to be BCD number. The subroutine DATADISP will convert the accumulator A and save the LED pattern to the display buffer in data field. Main code is repeat scan the display. So we will see the counting down every one second.

Procedure

1. Ensure SW4 is set to 10ms tick position.

2. Enter the hex code from address 1800 to 1842.

3. Set the variable sec at 2001 to 99.

4. Press PC and GO, Did you see any display on the 7-segment LED?

Exercise

1. The counting is decremented every one second. Can you modify the program to make counting up every one second.

2. Try change the code to make counting every 0.1 second.

3. If we want to stop count when time is zero, give the idea how to do that?

Summary

We see that the main code is only scan the display. It reads the display buffer memory and writes to the LED. But when the CPU was interrupted by 10ms tick, Z80 stop running the main code and jump to interrupt service routine. We use variable to count the number of interrupt, if it is equal to 100 ticks, time has passed for one second. We can use another variable for counting 60 seconds for one minute as well.

LAB 11 LCD MODULE INTERFACE

In the applications that need more details readout, the LCD module is more suitable. One of the example is LCD text module having HD44780 controller chip. Our kit prepares the 16-pin header for easy interface. We can learn how to use it without the need of extra interfacing circuit. The LCD module will need only female 16-pin header for direct plugging into the on-board 16-pin header. Left hand is pin 1, and right hand is pin 16.

The example is 16 characters x 2 lines or 16x2 text LCD.

Each character's position has its location depending on the size of LCD module.

The kit provides a hardware driver for the LCD module. The hardware driver will set the LCD mode, clear display, set cursor position, write the ASCII data to display registers. User may try many size of LCD, the driver can set the position for up to 4 lines LCD module.

The list of hardware driver is shown below.

Subroutine	Address	Parameters	Function
INITLCD	0B64H	none	Initialize LCD
PRINTCHAR	0CCEH	L=ASCII	Print letter on LCD
PRINTTEXT	0CC8H	HL=pointer	Print text
GOTOXY	0CC2H	H=X, L=Y	Set cursor position
CLEAR	0B06H	none	Clear screen
PRINTHEX	0CD4H	A=nn	Print hex
PRINTINT	0CDBH	HL=nnnn	Print decimal number
DELAY	0CE1H	none	Delay ~0.5s

Steps to enable the LCD module and print a letter or text are as follows.

1. Initialize the module.

2. Write the ASCII code or text to LCD.

3. Set position, if need another location to be printed.

4. Clear display if needed.

Character code for LCD module is the same as ASCII code. For example letter 'A', the code is 41H, 'B' is 42H. For numeric, letter '1', code is 31H.

For scientific or mathematics symbols, different manufacturers may provide different code. We may test it by writing the code to the LCD and check it then.

Character Table for HD44780 LCD Module

Let us try print text “Hello worlds” on the first line and letter 'A' on the second line.

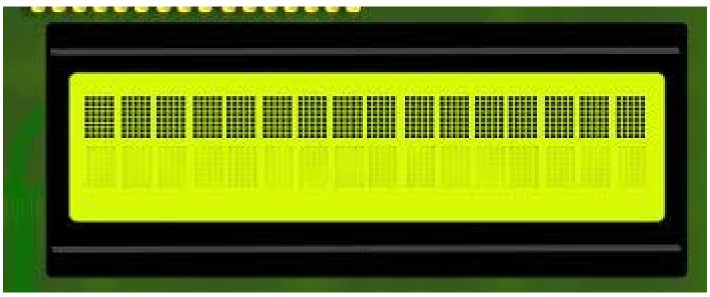
```
1800                            .ORG 1800H
1800
1800                 INITLCD    .EQU 0B64H
1800                 PRINTCHAR  .EQU 0CCEH
1800                 PRINTTEXT  .EQU 0CC8H
1800                 GOTOXY     .EQU 0CC2H
1800
1800 CD 64 0B        MAIN       CALL INITLCD
1803
1803 21 16 18                   LD HL,HELLO
1806 CD C8 0C                   CALL PRINTTEXT
1809
1809 26 01                      LD H,1
180B 2E 00                      LD L,0   ; X=0, Y=1
180D CD C2 0C                   CALL GOTOXY
1810
1810 2E 41                      LD L,'A'
1812 CD CE 0C                   CALL PRINTCHAR
1815
1815 FF                         RST 38H
1816
1816 48 65 6C 6C 6F 20 HELLO    .BYTE "Hello worlds",0
181C 77 6F 72 6C 64 73 00
1823
1823                            .END
1823
tasm: Number of errors = 0
```

Main code begins with initialize the LCD module, then print text “hello worlds” and print letter 'A' on LCD screen.

Procedure

1. Turn the kit's power off, then plug the LCD module to the 16-pin header.

2. Turn power on, adjust R18 until the first line becomes black.



3. Enter the hex code from 1800 to 1822.

4. Press PC and GO, Did you see any text on LCD screen?

Exercise

1. The message “hello worlds” is called ASCII string. At the end, it has terminator, 0. If you want to show your name, how to do that?

---

2. Try shifting letter 'A' to center of 2nd line. How to do that?

We see that the LCD module has more space for displaying many meaningful values. Suppose we want to display the memory content 4-byte on the LCD. How to do that?

```
1800                                .ORG 1800H
1800
1800
1800                  INITLCD       .EQU 0B64H
1800                  PRINTCHAR     .EQU 0CCEH
1800                  PRINTTEXT     .EQU 0CC8H
1800                  GOTOXY        .EQU 0CC2H
1800                  CLEAR         .EQU 0B06H
1800                  PRINTHEX      .EQU 0CD4H
1800
1800                  PRINTINT       .EQU 0CDBH
1800
1800                  DELAY         .EQU 0CE1H
1800
1800
1800
1800 CD 64 0B         MAIN          CALL INITLCD
1803
1803 21 00 18                       LD HL,1800H
1806 7C                             LD A,H
1807 E5                             PUSH HL
1808 CD D4 0C                       CALL PRINTHEX
180B D1                             POP DE
180C
180C 7B                             LD A,E
180D CD D4 0C                       CALL PRINTHEX
1810 2E 3A                          LD L,’:’
1812 CD CE 0C                       CALL PRINTCHAR
```

```
1815
1815 06 04                    LD B,4
1817 1A          LOOP         LD A,(DE)
1818 CD D4 0C                 CALL PRINTHEX
181B 2E 20                    LD L,' '
181D CD CE 0C                 CALL PRINTCHAR
1820
1820 13                       INC DE
1821 10 F4                    DJNZ LOOP
1823
1823 FF                       RST 38H
1824
1824                          .END
1824
tasm: Number of errors = 0
```

We use PRINTHEX subroutine for printing one byte in HEX number. The value to be printed is loaded into register A. For symbol, we use PRINTCHAR subroutine that prints the symbol from ASCII code directly.

The printout on the LCD will be 1800: CD 64 0B 21.

Procedure

1. Enter the hex code from 1800 to 1823.

2. Press PC and GO, did you see the printout as described above?

Exercise

1. Modify the program to display the 2nd line for the next four bytes. Show the printout result. Compare the printout with the memory contents by monitor key ADDR and key +.

The LCD driver also provides subroutine that prints unsigned 16-bit integer in decimal number, PRINTINT. Let us test it.

```
1800                           .ORG 1800H
1800
1800             INITLCD       .EQU 0B64H
1800             GOTOXY        .EQU 0CC2H
1800             PRINTINT      .EQU 0CDBH
1800             DELAY         .EQU 0CE1H
1800
1800 CD 64 0B    MAIN          CALL INITLCD
1803
1803 21 00 00                  LD HL,0
1806
1806 E5          LOOP          PUSH HL
```

```
0013   1807 CD DB 0C             CALL PRINTINT
0014   180A CD E1 0C             CALL DELAY
0015   180D
0016   180D 21 00 00             LD HL,0
0017   1810 CD C2 0C             CALL GOTOXY
0018   1813
0019   1813 E1                   POP HL
0020   1814 23                   INC HL
0021   1815
0022   1815 C3 06 18             JP LOOP
0023   1818
0024   1818                      .END
0025   1818
tasm: Number of errors = 0
```

The HL register is loaded with initial value of 0000 and printed it to LCD in decimal representation. The actual incrementing by INC HL is binary counting up. Delay provides approx. 0.5s delay, so we can see the incrementing clearly. To preserve the HL contents, instruction PUSH HL and POP HL are used to save HL contents to stack memory.

Procedure

1. Enter the hex code from 1800 to 1817.

2. Press PC and GO, did you see the printout of number counting?

Summary

The LCD module is suitable for applications that need more details readout. The Z80 kit provides both hardware interface and drivers subroutine for experimenting. The drivers is composed of hardware level subroutine. To use the LCD we must first initialize it then we can write the ASCII letter or string to the LCD.

LAB 12 SERIAL COMMUNICATION

The Z80 kit has RS232C communication port. We can test sending/receiving data between two computers using the RS232C. The serial data is asynchronous format. The speed is 2400 bit/s, 8-data bit, no parity and one stop bit. The RS232C is designed for connecting between two computers or between computer and data communication equipment (DCE) such as MODEM.

The kit has two keys for upload and download Intel hex file. The Intel hex file is generated from assembler or compiler program. We will learn how serial data is sent over the serial port. Since the kit has no UART chip, serial data streams are generated by software control. The monitor subroutines provide two basic functions, i.e. 1) COUT for sending a byte and 2) CIN for receiving a byte.

Connecting the kit to VT100 terminal (computer to computer)

VT100 terminal can be built using PC running communication program, like teraterm. The RS232C cable is cross cable.

List of subroutines for serial communication.

Subroutine	Address	Parameters	Function
CIN	08B9H	A=byte	Receive byte from 2400 bit/s terminal
COUT	0861H	A=byte	Send byte to 2400 bit/s terminal
PSTRING	0852H	IX=pointer	Print ASCII string to terminal
OUT2X	0875H	A=byte	Write hex to terminal
NEWLINE	088CH	none	Send CR LF to terminal

Let us test the code that prints “hello worlds” to terminal.

```
0001   1800                          .ORG 1800H
0002   1800
0003   1800              PSTRING     .EQU 0852H
0004   1800              CIN         .EQU 08B9H
0005   1800              COUT        .EQU 0861H
0006   1800              OUT2X       .EQU 0875H
0007   1800              NEWLINE     .EQU 088CH
0008   1800              DELAY       .EQU 0CE1H
0009   1800
0010   1800
0011   1800 CD 10 18     START       CALL INIT ; SET UART MARK LEVEL
0012   1803
0013   1803 CD 8C 08      MAIN       CALL NEWLINE
0014   1806 DD 21 19 18              LD IX,HELLO
0015   180A CD 52 08                 CALL PSTRING
0016   180D C3 03 18                 JP MAIN
0017   1810
0018   1810 3E FF         INIT       LD A,0FFH
0019   1812 CD 61 08                 CALL COUT
0020   1815 CD E1 0C                 CALL DELAY
0021   1818 C9                       RET
0022   1819
0023   1819 48 65 6C 6C 6F 20 HELLO  .BYTE "Hello worlds",0
0023   181F 77 6F 72 6C 64 73 00
0024   1826
0025   1826                           .END
0026   1826
tasm: Number of errors = 0
```

The kit shares TXD pin with on-board speaker, thus to send data correctly, we must set mark level before sending the serial data. This can be done with subroutine INIT. Main code sends NEWLINE and print text using PSTRING.

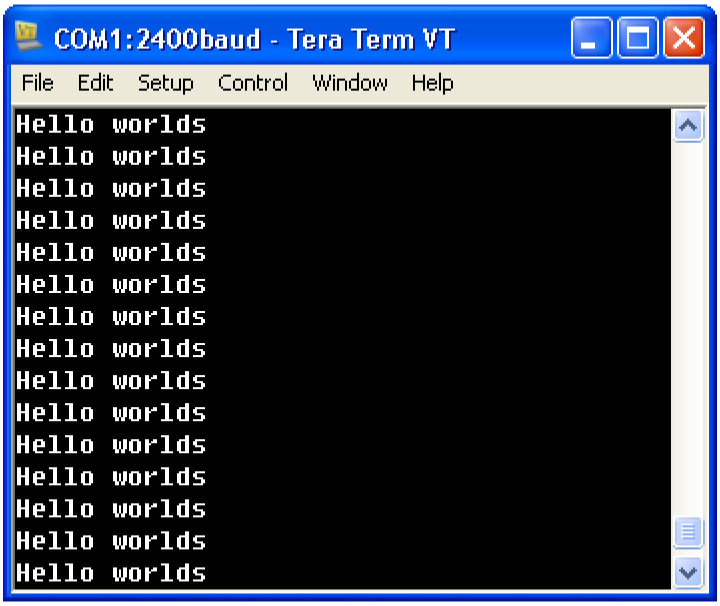
Procedure

1. Enter the hex code from 1800 to 1825.

2. Connect the kit to terminal COM port using RS232C cross cable.

3. Run teraterm with 2400 bit/s, 8-data bit, no parity and one stop bit.

4. Run the program with key RESET, PC and GO. We will get the message on terminal screen.



Exercise

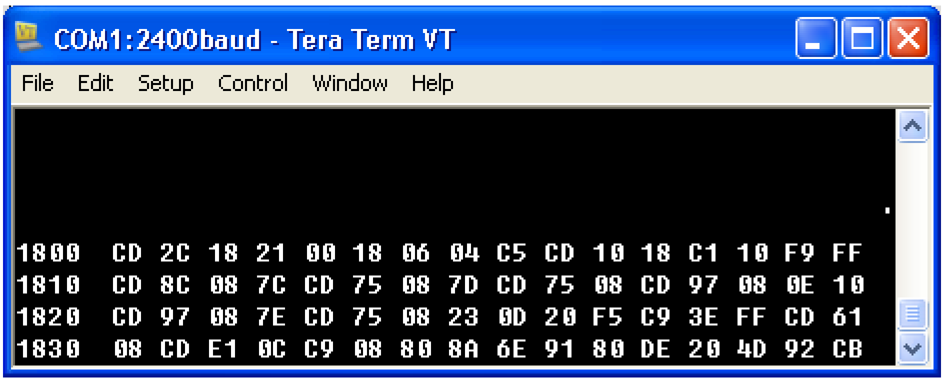
1. Modify the program to send your message like, Hello friends.

Now we will try print the memory contents with OUT2X subroutine.

```
1800                        .ORG 1800H
1800
1800            PSTRING     .EQU 0852H
1800            CIN         .EQU 08B9H
1800            COUT        .EQU 0861H
1800            OUT2X       .EQU 0875H
1800            NEWLINE     .EQU 088CH
1800            ONESPACE    .EQU 0897H
1800            DELAY       .EQU 0CE1H
1800
1800
1800 CD 2C 18   START       CALL INIT ; SET UART MARK LEVEL
1803 21 00 18                LD HL,1800H
1806
1806 06 04                  LD B,4
1808
1808 C5         LOOP        PUSH BC
1809 CD 10 18               CALL HEXDUMP
180C C1                     POP BC
180D 10 F9                  DJNZ LOOP
180F
180F FF                     RST 38H
1810
1810
1810
```

```
0026   1810 CD 8C 08        HEXDUMP     CALL NEWLINE
0027   1813 7C                          LD A,H
0028   1814 CD 75 08                    CALL OUT2X
0029   1817 7D                          LD A,L
0030   1818 CD 75 08                    CALL OUT2X
0031   181B CD 97 08                    CALL ONESPACE
0032   181E
0033   181E 0E 10                       LD C,16
0034   1820
0035   1820 CD 97 08        HEXDUMP2    CALL ONESPACE
0036   1823 7E                          LD A,(HL)
0037   1824 CD 75 08                    CALL OUT2X
0038   1827 23                          INC HL
0039   1828
0040   1828 0D                          DEC C
0041   1829 20 F5                       JR NZ,HEXDUMP2
0042   182B C9                          RET
0043   182C
0044   182C
0045   182C
0046   182C 3E FF           INIT        LD A,0FFH
0047   182E CD 61 08                    CALL COUT
0048   1831 CD E1 0C                    CALL DELAY
0049   1834 C9                          RET
0050   1835
0051   1835                             .END
0052   1835
tasm: Number of errors = 0
```

The program begins with writing ADDRESS, 1800 then prints one space, followed with 16 bytes memory contents in hex number using OUT2X. Register C is inner loop counter for 16 bytes. The outer loop counter is register B at line 0015. The printout on terminal screen would look like this.



Procedure

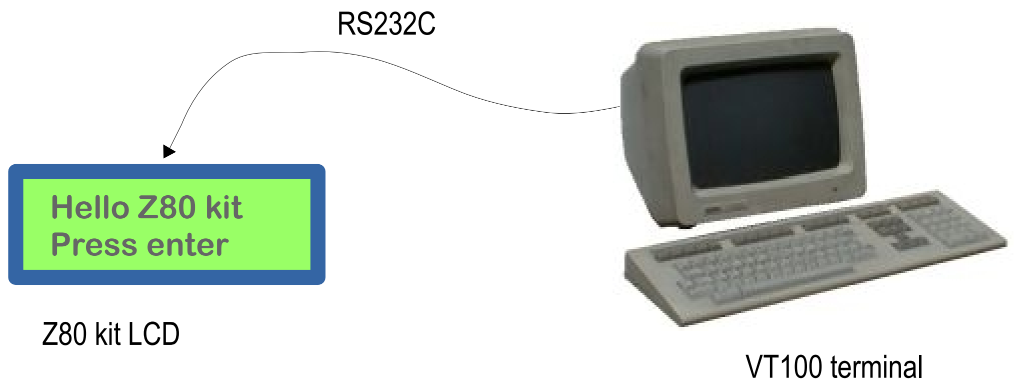
1. Enter the hex code from 1800 to 1835.

2. Run the program with key RESET, PC and GO. Did you get similar screen like above?

Exercise

1. Modify the program to print more, says 16 lines.

The kit can receive serial data as well. We will see how to type message on VT10 terminal and display it on Z80 Kit's LCD.




```
1800                             .ORG 1800H
1800
1800                INITLCD      .EQU 0B64H
1800                PRINTCHAR    .EQU 0CCEH
1800                GOTOXY       .EQU 0CC2H
1800                CLEAR        .EQU 0B06H
1800
1800                CIN          .EQU 08B9H
1800                COUT         .EQU 0861H
1800                CR           .EQU 0DH
1800                ESC          .EQU 1BH
1800
1800
1800 CD 64 0B       MAIN         CALL INITLCD
1803
1803 CD B9 08       LOOP         CALL CIN
1806 FE 0D                       CP CR
1808 20 08                       JR NZ,SKIP1
180A 21 00 01                    LD HL,0100H
180D CD C2 0C                    CALL GOTOXY
1810 18 0D                       JR SKIP3
1812
1812 FE 1B          SKIP1        CP ESC
1814 20 05                       JR NZ,SKIP2
1816 CD 06 0B                    CALL CLEAR
1819 18 04                       JR SKIP3
181B
181B 6F              SKIP2       LD L,A
181C CD CE 0C                    CALL PRINTCHAR
181F
```

```
0031   181F 18 E2          SKIP3        JR LOOP
0032   1821
0033   1821                             .END
0034   1821
tasm: Number of errors = 0
```

Main code reads a character received from terminal and check if it is CR or ESC. For CR it will set the new line, for ESC, it will clear the LCD display. The message that typed on the terminal will be displayed on LCD concurrently.

Procedure

1. We will need 16x2 LCD, 2400 bit/s VT100 terminal and RS232C cable for this experiment.

2. Enter hex code from 1800 to 1820.

3. Press PC and GO, then type a message on VT100 terminal. The message will show on the Kit's LCD.

4. To enter newline, press key Enter or clear LCD screen with key ESC.

Summary

Sending or receiving digital data between computers mostly uses serial communication. The Z80 kit has software control UART for asynchronous communication. Signal level of the serial port is RS232C. The low level driver subroutines for serial port are COUT and CIN. COUT is for sending 8-bit data in register A and CIN for receiving data to register A. Upper level subroutines e.g., PSTRING will print ASCII string, OUT2X will print register in HEX number.

We can use PC running terminal program to emulate VT100 terminal. The terminal uses ASCII code for data exchange. Printable ASCII letters are from 20H to 7FH. Control codes are from 00 to 1FH. The example of control codes are CR and LF for entering the new line.

The kit also has text file downloading key. The Intel hex file generated from assembler or c compile programs is ASCII text file. For binary file transaction, the upper level software will provide the protocol for byte sequence, acknowledgment, and error checking. The popular are XMODEM, YMODEM.

APPENDIX A

MONITOR SUBROUTINES

Subroutine	Address	Parameters	Function
INITLCD	0B64H	none	Initialize LCD
PRINTCHAR	0CCEH	L=ASCII	Print letter on LCD
PRINTTEXT	0CC8H	HL=pointer	Print text on LCD
GOTOXY	0CC2H	H=X, L=Y	Set cursor position
CLEAR	0B06H	none	Clear LCD screen
PRINTHEX	0CD4H	A=nn	Print hex on LCD
PRINTINT	0CDBH	HL=nnnn	Print decimal on LCD
SCAN1	0624H	IX=buffer	Scan display one cycle
SCAN	05FEH	IX=buffer	Scan until key pressed
HEX7SEG	0678H	A=nn	Convert A to 7-segment
TONE1K	05DEH	HL=period	Produce tone 1kHz
TONE2K	05E2H	HL=period	Produce tone 2kHz
CIN	08B9H	A=byte	Receive byte from 2400 bit/s terminal
COUT	0861H	A=byte	Send byte to 2400 bit/s terminal
PSTRING	0852H	IX=pointer	Print ASCII string to terminal
OUT2X	0875H	A=byte	Write hex to terminal
NEWLINE	088CH	none	Send CR LF to terminal
DELAY	0CE1H	none	Delay ~0.5s

APPENDIX B

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Examples:

1. Letter 'A', the ASCII code is 41H.
2. Letter '1', the ASCII code is 31H.
3. CR code is 0DH, LF is 0AH, ESC is 1BH.

Hexadecimal Representation

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Example: FF = 1111 1111, 0D= 0000 1101

REDUCED Z80 INSTRUCTION SET

Edited by Phil Townshend 2003

Conditional jumps show 2 values: e.g. "RET Z = 1/3" (cycles if condition not met)/(cycles if condition met)

FLAG DEF's	
	* = affected
	? = unknown
	x = no change

INSTR.	HEX CODE			CYC	FREG - Flags register							
					b7	b6	b5	b4	b3	b2	b1	b0
					S	Z	-	H	-	-	N	C
ADC A,C	89			1	*	*		*			0	*
ADC A,n	CE	nn		2	*	*		*			0	*
ADD A,(HL)	86			2	*	*		*			0	*
ADD A,A	87			1	*	*		*			0	*
ADD A,B	80			1	*	*		*			0	*
ADD A,C	81			1	*	*		*			0	*
ADD A,D	82			1	*	*		*			0	*
ADD A,E	83			1	*	*		*			0	*
ADD A,H	84			1	*	*		*			0	*
ADD A,L	85			1	*	*		*			0	*
ADD A,n	C6	nn		2	*	*		*			0	*
ADD HL,BC	09			3	x	x		x			0	*
ADD HL,DE	19			3	x	x		x			0	*
ADD HL,HL	29			3	x	x		x			0	*
AND (HL)	A6			2	*	*		1			0	0
AND A	A7			1	*	*		1			0	0
AND B	A0			1	*	*		1			0	0
AND C	A1			1	*	*		1			0	0
AND D	A2			1	*	*		1			0	0
AND E	A3			1	*	*		1			0	0
AND H	A4			1	*	*		1			0	0
AND L	A5			1	*	*		1			0	0
AND n	E6	nn		2	*	*		1			0	0
CALL C,mn	DC	nn	mm	3/5	x	x		x			x	x
CALL mn	CD	nn	mm	5	x	x		x			x	x
CALL NC,mn	D4	nn	mm	3/5	x	x		x			x	x
CALL NZ,mn	C4	nn	mm	3/5	x	x		x			x	x
CALL Z,nn	CC	nn	mm	3/5	x	x		x			x	x
CCF	3F			1	x	x		x			0	*
CP (HL)	BE			2	*	*		*			1	*
CP A	BF			1	*	*		*			1	*
CP B	B8			1	*	*		*			1	*
CP C	B9			1	*	*		*			1	*
CP D	BA			1	*	*		*			1	*
CP E	BB			1	*	*		*			1	*
CP H	BC			1	*	*		*			1	*
CP L	BD			1	*	*		*			1	*
CP n	FE	nn		2	*	*		*			1	*
CPL	2F			1	x	x		1			1	x
DEC (HL)	35			3	*	*		*			1	x
DEC A	3D			1	*	*		*			1	x
DEC B	05			1	*	*		*			1	x
DEC BC	0B			1	x	x		x			x	x
DEC C	0D			1	*	*		*			1	x
DEC D	15			1	*	*		*			1	x
DEC DE	1B			1	x	x		x			x	x
DEC E	1D			1	*	*		*			1	x
DEC H	25			1	*	*		*			1	x
DEC HL	2B			1	x	x		x			x	x
DEC L	2D			1	*	*		*			1	x
DEC SP	3B			1	x	x		x			x	x
DJNZ,d	10	dd		2/3	x	x		x			x	x
EX AF,AF'	08			1	x	x		x			x	x
EX DE,HL	EB			1	x	x		x			x	x
EXX	D9			1 - (2)	x	x		x			x	x

INSTR.	HEX CODE			CYC	FREG - Flags register							
					b7	b6	b5	b4	b3	b2	b1	b0
					S	Z	-	H	-	-	N	C
HALT	76			1	x	x		x			x	x
IN A,(nn)	DB	nn		3	x	x		x			x	x
INC (HL)	34			3	*	*		*			0	x
INC A	3C			1	*	*		*			0	x
INC B	04			1	*	*		*			0	x
INC BC	03			1	x	x		x			x	x
INC C	0C			1	*	*		*			0	x
INC D	14			1	*	*		*			0	x
INC DE	13			1	x	x		x			x	x
INC E	1C			1	*	*		*			0	x
INC H	24			1	*	*		*			0	x
INC HL	23			1	x	x		x			x	x
INC L	2C			1	*	*		*			0	x
INC SP	33			1	x	x		x			x	x
JP (HL)	E9			1	x	x		x			x	x
JP C,mn	DA	nn	mm	3	x	x		x			x	x
JP mn	C3	nn	mm	3	x	x		x			x	x
JP N,mn	FA	nn	mm	3	x	x		x			x	x
JP NC,mn	D2	nn	mm	3	x	x		x			x	x
JP NZ mn	C2	nn	mm	3	x	x		x			x	x
JP Z,mn	CA	nn	mm	3	x	x		x			x	x
JR C d	38	dd		2/3	x	x		x			x	x
JR dd	18	dd		3	x	x		x			x	x
JR NC,d	30	dd		2/3	x	x		x			x	x
JR NZ,d	20	dd		2/3	x	x		x			x	x
JR Z d	28	dd		2/3	x	x		x			x	x
LD (BC),A	02			2	x	x		x			x	x
LD (DE),A	12			2	x	x		x			x	x
LD (HL),A	77			2	x	x		x			x	x
LD (HL),B	70			2	x	x		x			x	x
LD (HL),C	71			2	x	x		x			x	x
LD (HL),D	72			2	x	x		x			x	x
LD (HL),E	73			2	x	x		x			x	x
LD (HL),H	74			2	x	x		x			x	x
LD (HL),L	75			2	x	x		x			x	x
LD (HL),n	36	nn		3	x	x		x			x	x
LD (mn),A	32	nn	mm	4	x	x		x			x	x
LD (mn),HL	22	nn	mm	5	x	x		x			x	x
LD A,(BC)	0A			2	x	x		x			x	x
LD A,(DE)	1A			2	x	x		x			x	x
LD A,(HL)	7E			2	x	x		x			x	x
LD A,(mn)	3A	nn	mm	4	x	x		x			x	x
LD A,A	7F			1	x	x		x			x	x
LD A,B	78			1	x	x		x			x	x
LD A,C	79			1	x	x		x			x	x
LD A,D	7A			1	x	x		x			x	x
LD A,E	7B			1	x	x		x			x	x
LD A,H	7C			1	x	x		x			x	x
LD A,L	7D			1	x	x		x			x	x
LD A,n	3E	nn		2	x	x		x			x	x
LD B,(HL)	46			2	x	x		x			x	x
LD B,A	47			1	x	x		x			x	x
LD B,B	40			1	x	x		x			x	x
LD B,C	41			1	x	x		x			x	x
LD B,D	42			1	x	x		x			x	x

REDUCED Z80 INSTRUCTION SET

Edited by Phil Townshend 2003

Conditional jumps show 2 values: e.g. "RET Z = 1/3" (cycles if condition not met)/(cycles if condition met)

FLAG DEF's	
	* = affected
	? = unknown
	x = no change

INSTR.	HEX CODE			CYC	FREG - Flags register							
					b7	b6	b5	b4	b3	b2	b1	b0
					S	Z	-	H	-	-	N	C
ADC A,C	89			1	*	*		*			0	*
ADC A,n	CE	nn		2	*	*		*			0	*
ADD A,(HL)	86			2	*	*		*			0	*
ADD A,A	87			1	*	*		*			0	*
ADD A,B	80			1	*	*		*			0	*
ADD A,C	81			1	*	*		*			0	*
ADD A,D	82			1	*	*		*			0	*
ADD A,E	83			1	*	*		*			0	*
ADD A,H	84			1	*	*		*			0	*
ADD A,L	85			1	*	*		*			0	*
ADD A,n	C6	nn		2	*	*		*			0	*
ADD HL,BC	09			3	x	x		x			0	*
ADD HL,DE	19			3	x	x		x			0	*
ADD HL,HL	29			3	x	x		x			0	*
AND (HL)	A6			2	*	*		1			0	0
AND A	A7			1	*	*		1			0	0
AND B	A0			1	*	*		1			0	0
AND C	A1			1	*	*		1			0	0
AND D	A2			1	*	*		1			0	0
AND E	A3			1	*	*		1			0	0
AND H	A4			1	*	*		1			0	0
AND L	A5			1	*	*		1			0	0
AND n	E6	nn		2	*	*		1			0	0
CALL C,mn	DC	nn	mm	3/5	x	x		x			x	x
CALL mn	CD	nn	mm	5	x	x		x			x	x
CALL NC,mn	D4	nn	mm	3/5	x	x		x			x	x
CALL NZ,mn	C4	nn	mm	3/5	x	x		x			x	x
CALL Z,nn	CC	nn	mm	3/5	x	x		x			x	x
CCF	3F			1	x	x		x			0	*
CP (HL)	BE			2	*	*		*			1	*
CP A	BF			1	*	*		*			1	*
CP B	B8			1	*	*		*			1	*
CP C	B9			1	*	*		*			1	*
CP D	BA			1	*	*		*			1	*
CP E	BB			1	*	*		*			1	*
CP H	BC			1	*	*		*			1	*
CP L	BD			1	*	*		*			1	*
CP n	FE	nn		2	*	*		*			1	*
CPL	2F			1	x	x		1			1	x
DEC (HL)	35			3	*	*		*			1	x
DEC A	3D			1	*	*		*			1	x
DEC B	05			1	*	*		*			1	x
DEC BC	0B			1	x	x		x			x	x
DEC C	0D			1	*	*		*			1	x
DEC D	15			1	*	*		*			1	x
DEC DE	1B			1	x	x		x			x	x
DEC E	1D			1	*	*		*			1	x
DEC H	25			1	*	*		*			1	x
DEC HL	2B			1	x	x		x			x	x
DEC L	2D			1	*	*		*			1	x
DEC SP	3B			1	x	x		x			x	x
DJNZ,d	10	dd		2/3	x	x		x			x	x
EX AF,AF'	08			1	x	x		x			x	x
EX DE,HL	EB			1	x	x		x			x	x
EXX	D9			1 - (2)	x	x		x			x	x

INSTR.	HEX CODE			CYC	FREG - Flags register							
					b7	b6	b5	b4	b3	b2	b1	b0
					S	Z	-	H	-	-	N	C
HALT	76			1	x	x		x			x	x
IN A,(nn)	DB	nn		3	x	x		x			x	x
INC (HL)	34			3	*	*		*			0	x
INC A	3C			1	*	*		*			0	x
INC B	04			1	*	*		*			0	x
INC BC	03			1	x	x		x			x	x
INC C	0C			1	*	*		*			0	x
INC D	14			1	*	*		*			0	x
INC DE	13			1	x	x		x			x	x
INC E	1C			1	*	*		*			0	x
INC H	24			1	*	*		*			0	x
INC HL	23			1	x	x		x			x	x
INC L	2C			1	*	*		*			0	x
INC SP	33			1	x	x		x			x	x
JP (HL)	E9			1	x	x		x			x	x
JP C,mn	DA	nn	mm	3	x	x		x			x	x
JP mn	C3	nn	mm	3	x	x		x			x	x
JP N,mn	FA	nn	mm	3	x	x		x			x	x
JP NC,mn	D2	nn	mm	3	x	x		x			x	x
JP NZ mn	C2	nn	mm	3	x	x		x			x	x
JP Z,mn	CA	nn	mm	3	x	x		x			x	x
JR C d	38	dd		2/3	x	x		x			x	x
JR dd	18	dd		3	x	x		x			x	x
JR NC,d	30	dd		2/3	x	x		x			x	x
JR NZ,d	20	dd		2/3	x	x		x			x	x
JR Z d	28	dd		2/3	x	x		x			x	x
LD (BC),A	02			2	x	x		x			x	x
LD (DE),A	12			2	x	x		x			x	x
LD (HL),A	77			2	x	x		x			x	x
LD (HL),B	70			2	x	x		x			x	x
LD (HL),C	71			2	x	x		x			x	x
LD (HL),D	72			2	x	x		x			x	x
LD (HL),E	73			2	x	x		x			x	x
LD (HL),H	74			2	x	x		x			x	x
LD (HL),L	75			2	x	x		x			x	x
LD (HL),n	36	nn		3	x	x		x			x	x
LD (mn),A	32	nn	mm	4	x	x		x			x	x
LD (mn),HL	22	nn	mm	5	x	x		x			x	x
LD A,(BC)	0A			2	x	x		x			x	x
LD A,(DE)	1A			2	x	x		x			x	x
LD A,(HL)	7E			2	x	x		x			x	x
LD A,(mn)	3A	nn	mm	4	x	x		x			x	x
LD A,A	7F			1	x	x		x			x	x
LD A,B	78			1	x	x		x			x	x
LD A,C	79			1	x	x		x			x	x
LD A,D	7A			1	x	x		x			x	x
LD A,E	7B			1	x	x		x			x	x
LD A,H	7C			1	x	x		x			x	x
LD A,L	7D			1	x	x		x			x	x
LD A,n	3E	nn		2	x	x		x			x	x
LD B,(HL)	46			2	x	x		x			x	x
LD B,A	47			1	x	x		x			x	x
LD B,B	40			1	x	x		x			x	x
LD B,C	41			1	x	x		x			x	x
LD B,D	42			1	x	x		x			x	x

REDUCED Z80 INSTRUCTION SET

FLAG DEF's	* = affected
	? = unknown
	x = no change

Conditional jumps show 2 values: e.g. "RET Z = 1/3" (cycles if condition not met/condition met)

INSTR.	HEX CODE			CYC	FREG - Flags register							
					b7	b6	b5	b4	b3	b2	b1	b0
					S	Z	-	H	-	-	N	C
LD B,E	43			1	x	x		x			x	x
LD B,H	44			1	x	x		x			x	x
LD B,L	45			1	x	x		x			x	x
LD B,n	06	nn		2	x	x		x			x	x
LD BC.mn	01	nn	mm	3	x	x		x			x	x
LD C,(HL)	4E			2	x	x		x			x	x
LD C,A	4F			1	x	x		x			x	x
LD C,B	48			1	x	x		x			x	x
LD C,C	49			1	x	x		x			x	x
LD C,D	4A			1	x	x		x			x	x
LD C,E	4B			1	x	x		x			x	x
LD C,H	4C			1	x	x		x			x	x
LD C,L	4D			1	x	x		x			x	x
LD C,n	0E	nn		2	x	x		x			x	x
LD D,(HL)	56			2	x	x		x			x	x
LD D,A	57			1	x	x		x			x	x
LD D,B	50			1	x	x		x			x	x
LD D,C	51			1	x	x		x			x	x
LD D,D	52			1	x	x		x			x	x
LD D,E	53			1	x	x		x			x	x
LD D,H	54			1	x	x		x			x	x
LD D,L	55			1	x	x		x			x	x
LD D,n	16	nn		2	x	x		x			x	x
LD DE,mn	11	nn	mm	3	x	x		x			x	x
LD E,(HL)	5E			2	x	x		x			x	x
LD E,A	5F			1	x	x		x			x	x
LD E,B	58			1	x	x		x			x	x
LD E,C	59			1	x	x		x			x	x
LD E,D	5A			1	x	x		x			x	x
LD E,E	5B			1	x	x		x			x	x
LD E,H	5C			1	x	x		x			x	x
LD E,L	5D			1	x	x		x			x	x
LD E,n	1E	nn		2	x	x		x			x	x
LD H,(HL)	66			2	x	x		x			x	x
LD H,A	67			1	x	x		x			x	x
LD H,B	60			1	x	x		x			x	x
LD H,C	61			1	x	x		x			x	x
LD H,D	62			1	x	x		x			x	x
LD H,E	63			1	x	x		x			x	x
LD H,H	64			1	x	x		x			x	x
LD H,L	65			1	x	x		x			x	x
LD H,n	26	nn		2	x	x		x			x	x
LD HL,(mn)	2A	nn	mm	5	x	x		x			x	x
LD HL,mn	21	nn	mm	3	x	x		x			x	x
LD L,(HL)	6E			2	x	x		x			x	x
LD L,A	6F			1	x	x		x			x	x
LD L,B	68			1	x	x		x			x	x
LD L,C	69			1	x	x		x			x	x
LD L,D	6A			1	x	x		x			x	x
LD L,E	6B			1	x	x		x			x	x
LD L,H	6C			1	x	x		x			x	x
LD L,L	6D			1	x	x		x			x	x
LD L,n	2E	nn		2	x	x		x			x	x
NOP	00			1	x	x		x			x	x

INSTR.	HEX CODE		CYC	FREG - Flags register							
				b7	b6	b5	b4	b3	b2	b1	b0
				S	Z	-	H	-	-	N	C
NEG	ED	44	2	*	*		*			1	*
OR (HL)	B6		2	*	*		0			0	0
OR A	B7		1	*	*		0			0	0
OR B	B0		1	*	*		0			0	0
OR C	B1		1	*	*		0			0	0
OR D	B2		1	*	*		0			0	0
OR E	B3		1	*	*		0			0	0
OR H	B4		1	*	*		0			0	0
OR L	B5		1	*	*		0			0	0
OR n	F6	nn	2	*	*		0			0	0
OUT (n),A	D3	nn	3	x	x		x			x	x
POP AF	F1		3	x	x		x			x	x
POP BC	C1		3	x	x		x			x	x
POP DE	D1		3	x	x		x			x	x
POP HL	E1		3	x	x		x			x	x
PUSH AF	F5		3	x	x		x			x	x
PUSH BC	C5		3	x	x		x			x	x
PUSH DE	D5		3	x	x		x			x	x
PUSH HL	E5		3	x	x		x			x	x
RET	C9		3	x	x		x			x	x
RET C	D8		1/3	x	x		x			x	x
RET NC	D0		1/3	x	x		x			x	x
RET NZ	C0		1/3	x	x		x			x	x
RET Z	C8		1/3	x	x		x			x	x
RLA	17		1	x	x		0			0	*
RLCA	07		1	x	x		0			0	*
RRA	1F		1	x	x		0			0	*
RRCA	0F		1	x	x		0			0	*
SBC A,C	99		1	*	*		*			1	*
SBC A,n	DE	nn	2	*	*		*			1	*
SCF	37		1	x	x		0			0	1
SUB A,(HL)	96		2	*	*		*			1	*
SUB A,B	90		1	*	*		*			1	*
SUB A,C	91		1	*	*		*			1	*
SUB A,D	92		1	*	*		*			1	*
SUB A,E	93		1	*	*		*			1	*
SUB A,H	94		1	*	*		*			1	*
SUB A,L	95		1	*	*		*			1	*
SUB n	D6	nn	2	*	*		*			1	*
XOR (HL)	AE		2	*	*		0			0	0
XOR B	A8		1	*	*		0			0	0
XOR C	A9		1	*	*		0			0	0
XOR D	AA		1	*	*		0			0	0
XOR E	AB		1	*	*		0			0	0
XOR H	AC		1	*	*		0			0	0
XOR L	AD		1	*	*		0			0	0
XOR n	EE	nn	2	*	*		0			0	0

REDUCED Z80 INSTRUCTION SET

FLAG DEF's	* = affected
	? = unknown
	x = no change

Conditional jumps show 2 values: e.g. "RET Z = 1/3" (cycles if condition not met/condition met)

INSTR.	HEX CODE			CYC	FREG - Flags register							
					b7	b6	b5	b4	b3	b2	b1	b0
					S	Z	-	H	-	-	N	C
LD B,E	43			1	x	x		x			x	x
LD B,H	44			1	x	x		x			x	x
LD B,L	45			1	x	x		x			x	x
LD B,n	06	nn		2	x	x		x			x	x
LD BC.mn	01	nn	mm	3	x	x		x			x	x
LD C,(HL)	4E			2	x	x		x			x	x
LD C,A	4F			1	x	x		x			x	x
LD C,B	48			1	x	x		x			x	x
LD C,C	49			1	x	x		x			x	x
LD C,D	4A			1	x	x		x			x	x
LD C,E	4B			1	x	x		x			x	x
LD C,H	4C			1	x	x		x			x	x
LD C,L	4D			1	x	x		x			x	x
LD C,n	0E	nn		2	x	x		x			x	x
LD D,(HL)	56			2	x	x		x			x	x
LD D,A	57			1	x	x		x			x	x
LD D,B	50			1	x	x		x			x	x
LD D,C	51			1	x	x		x			x	x
LD D,D	52			1	x	x		x			x	x
LD D,E	53			1	x	x		x			x	x
LD D,H	54			1	x	x		x			x	x
LD D,L	55			1	x	x		x			x	x
LD D,n	16	nn		2	x	x		x			x	x
LD DE,mn	11	nn	mm	3	x	x		x			x	x
LD E,(HL)	5E			2	x	x		x			x	x
LD E,A	5F			1	x	x		x			x	x
LD E,B	58			1	x	x		x			x	x
LD E,C	59			1	x	x		x			x	x
LD E,D	5A			1	x	x		x			x	x
LD E,E	5B			1	x	x		x			x	x
LD E,H	5C			1	x	x		x			x	x
LD E,L	5D			1	x	x		x			x	x
LD E,n	1E	nn		2	x	x		x			x	x
LD H,(HL)	66			2	x	x		x			x	x
LD H,A	67			1	x	x		x			x	x
LD H,B	60			1	x	x		x			x	x
LD H,C	61			1	x	x		x			x	x
LD H,D	62			1	x	x		x			x	x
LD H,E	63			1	x	x		x			x	x
LD H,H	64			1	x	x		x			x	x
LD H,L	65			1	x	x		x			x	x
LD H,n	26	nn		2	x	x		x			x	x
LD HL,(mn)	2A	nn	mm	5	x	x		x			x	x
LD HL,mn	21	nn	mm	3	x	x		x			x	x
LD L,(HL)	6E			2	x	x		x			x	x
LD L,A	6F			1	x	x		x			x	x
LD L,B	68			1	x	x		x			x	x
LD L,C	69			1	x	x		x			x	x
LD L,D	6A			1	x	x		x			x	x
LD L,E	6B			1	x	x		x			x	x
LD L,H	6C			1	x	x		x			x	x
LD L,L	6D			1	x	x		x			x	x
LD L,n	2E	nn		2	x	x		x			x	x
NOP	00			1	x	x		x			x	x

INSTR.	HEX CODE		CYC	FREG - Flags register							
				b7	b6	b5	b4	b3	b2	b1	b0
				S	Z	-	H	-	-	N	C
NEG	ED	44	2	*	*		*			1	*
OR (HL)	B6		2	*	*		0			0	0
OR A	B7		1	*	*		0			0	0
OR B	B0		1	*	*		0			0	0
OR C	B1		1	*	*		0			0	0
OR D	B2		1	*	*		0			0	0
OR E	B3		1	*	*		0			0	0
OR H	B4		1	*	*		0			0	0
OR L	B5		1	*	*		0			0	0
OR n	F6	nn	2	*	*		0			0	0
OUT (n),A	D3	nn	3	x	x		x			x	x
POP AF	F1		3	x	x		x			x	x
POP BC	C1		3	x	x		x			x	x
POP DE	D1		3	x	x		x			x	x
POP HL	E1		3	x	x		x			x	x
PUSH AF	F5		3	x	x		x			x	x
PUSH BC	C5		3	x	x		x			x	x
PUSH DE	D5		3	x	x		x			x	x
PUSH HL	E5		3	x	x		x			x	x
RET	C9		3	x	x		x			x	x
RET C	D8		1/3	x	x		x			x	x
RET NC	D0		1/3	x	x		x			x	x
RET NZ	C0		1/3	x	x		x			x	x
RET Z	C8		1/3	x	x		x			x	x
RLA	17		1	x	x		0			0	*
RLCA	07		1	x	x		0			0	*
RRA	1F		1	x	x		0			0	*
RRCA	0F		1	x	x		0			0	*
SBC A,C	99		1	*	*		*			1	*
SBC A,n	DE	nn	2	*	*		*			1	*
SCF	37		1	x	x		0			0	*
SUB A,(HL)	96		2	*	*		*			1	*
SUB A,B	90		1	*	*		*			1	*
SUB A,C	91		1	*	*		*			1	*
SUB A,D	92		1	*	*		*			1	*
SUB A,E	93		1	*	*		*			1	*
SUB A,H	94		1	*	*		*			1	*
SUB A,L	95		1	*	*		*			1	*
SUB n	D6	nn	2	*	*		*			1	*
XOR (HL)	AE		2	*	*		0			0	0
XOR B	A8		1	*	*		0			0	0
XOR C	A9		1	*	*		0			0	0
XOR D	AA		1	*	*		0			0	0
XOR E	AB		1	*	*		0			0	0
XOR H	AC		1	*	*		0			0	0
XOR L	AD		1	*	*		0			0	0
XOR n	EE	nn	2	*	*		0			0	0

NOTE

NOTE

NOTE