Online Instructor’s Manual to accompany

THE 8051 MICROCONTROLLER: A SYSTEMS APPROACH

Muhammad Ali Mazidi
Janice Gillispie Mazidi
Rolin McKinlay

With contributions from
Ardeshir Eslami
and Sepehr Naimi

PEARSON

Boston  Columbus  Indianapolis  New York  San Francisco  Upper Saddle River
Amsterdam  Cape Town  Dubai  London  Madrid  Milan  Munich  Paris  Montreal  Toronto
Delhi  Mexico City  Sao Paulo  Sydney  Hong Kong  Seoul  Singapore  Taipei  Tokyo
CHAPTER 0: INTRODUCTION TO COMPUTING

Section 0.1: Numbering and Coding Systems

1. 
   (a) \(12_{10} = 1100_2\)
   (b) \(123_{10} = 0111\ 1011_2\)
   (c) \(63_{10} = 0011\ 1111_2\)
   (d) \(128_{10} = 1000\ 0000_2\)
   (e) \(1000_{10} = 0011\ 1110\ 1000_2\)

2. 
   (a) \(100100_2 = 36_{10}\)
   (b) \(1000001_2 = 65_{10}\)
   (c) \(11101_2 = 29_{10}\)
   (d) \(1010_2 = 10_{10}\)
   (e) \(00100010_2 = 34_{10}\)

3. 
   (a) \(100100_2 = 24_{16}\)
   (b) \(1000001_2 = 41_{16}\)
   (c) \(11101_2 = 1D_{16}\)
   (d) \(1010_2 = 0A_{16}\)
   (e) \(00100010_2 = 22_{16}\)

4. 
   (a) \(2B9_{16} = 0011\ 1011\ 1001_{10},\ 697_{10}\)
   (b) \(F44_{16} = 1111\ 0100\ 0100_{10},\ 3908_{10}\)
   (c) \(912_{16} = 1001\ 0001\ 0010_{10},\ 2322_{10}\)
   (d) \(2B_{16} = 0010\ 1011_{10},\ 43_{10}\)
   (e) \(FFFF_{16} = 1111\ 1111\ 1111\ 1111_{10},\ 65535_{10}\)

5. 
   (a) \(12_{10} = 0C_{16}\)
   (b) \(123_{10} = 7B_{16}\)
   (c) \(63_{10} = 3F_{16}\)
   (d) \(128_{10} = 80_{16}\)
   (e) \(1000_{10} = 3E8_{16}\)

6. 
   (a) \(1001010 = 0011\ 0110\)
   (b) \(111001 = 0000\ 0111\)
   (c) \(1000010 = 0111\ 1110\)
   (d) \(111111001 = 0000\ 1111\)

7. 
   (a) \(2C+3F = 6B\)
   (b) \(F34+5D6 = 150A\)
   (c) \(20000+12FF = 212FF\)
   (d) \(FFFF+2222 = 12221\)
8.  (a) $24F-129 = 126_{16}$  
(b) $FE9-5CC = A1D_{16}$  
(c) $2FFFF-FFFF = 30000_{16}$  
(d) $9FF25-4DD99 = 5218C_{16}$

9.  (a) Hex: 30, 31, 32, 33, 34, 35, 36, 37, 38, 39  

<table>
<thead>
<tr>
<th>ASCII (hex)</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>31</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
</tr>
<tr>
<td>8</td>
<td>38</td>
</tr>
<tr>
<td>9</td>
<td>39</td>
</tr>
</tbody>
</table>

10.  $000000 \ 22 \ 55 \ 2E \ 53 \ 2E \ 41 \ 2E \ 20 \ 69 \ 73 \ 20 \ 61 \ 20 \ 63 \ 6F \ 75$ \quad "U.S.A. is a cou
     $000010 \ 6E \ 74 \ 72 \ 79 \ 22 \ 0D \ 0A \ 22 \ 69 \ 6E \ 20 \ 4E \ 6F \ 72 \ 74 \ 68$ \quad ntry".."in North
     $000020 \ 20 \ 41 \ 6D \ 65 \ 72 \ 69 \ 63 \ 61 \ 22 \ 0D \ 0A$ \quad America"..

Section 0.2: Digital Primer

11.

12.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
13. 

![Logic Diagram](image)

14. 

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

15. 

![Logic Diagram](image)

16. 

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
17. 

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

18. 

![LSB Diagram](image)

19. 

![LSB Diagram](image)

20. 

<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>X</td>
<td>NC</td>
</tr>
<tr>
<td>Yes</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Yes</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Section 0.3: Semiconductor Memory

21. 

(a) 4  
(b) 4  
(c) 4  
(d) $1\,048\,576, 2^{20}$  
(e) 1024K  
(f) $1\,073\,741\,824, 2^{30}$  
(g) 1048576 K  
(h) 1024M  
(i) 8388608, 8192K
22. 1 million pages
23. The storage of the chip is measured in Megabits while the Computer memory is measured in Megabytes.
24. True, the more address lines the more memory locations.
25. True, the memory location size is fixed.
26. True, the more data lines the more memory locations
27. True
28. access time
29. True
30. electrically erasable
31. True
32. DRAM
33. SRAM
34. DRAM and SRAM
35. (c)
36. (c)
37. (a) 32Kx8, 256K  (f) 8Kx1, 8K
(b) 8Kx8, 64K     (g) 4Kx8, 32K
(c) 4Kx8, 32K     (h) 2Kx8, 16K
(d) 8Kx8, 64K     (i) 256Kx4, 1M
(e) 4Mx1, 4M      (j) 64Kx8, 512K
38. (a) 128K 14 8  (f) 256K 8 4
(b) 256K 15 8     (g) 8M 20 8
(c) 512K 16 8     (h) 16M 11 4
(d) 2M 18 8       (i) 512K 16 8
(e) 512K 16 8

Section 0.4: Bus Designing and Address Decoding

39. (a) 589824 bytes
(b) 576 kbytes
40. \(2^{32} - 1 = 4294967295\)
41. (a) FFh, 255
(b) FFFFh, 65535
(c) FFFF FFFFh, 4,294,967,295
(d) FFFF FFFF FFFFh, 18,446,744,073,709,551,615
42. (a) 64K
(b) 16M
(c) 4096 Mega, 4G
(d) 256 Tera, 262144 Giga, 268435456 Mega
43. Data bus is bidirectional, address is unidirectional.
44. 4000h - 7FFFh
45. Each controls 8K bytes block

46. 8000h - 8FFFh, B000h - BFFFh, E000h - EFFFh

47. Each controls 8K bytes block

48. 6000h - 7FFFh, C000h - DFFFh, E000h - FFFFh

49. Each controls 32K bytes block

50. 4000h - 7FFFh, 8000h - BFFFh, C000h – FFFFh

51. Tri-state buffer

52. Latch

53. The linear select

54. Most often each port (or even memory) can be accessed by a single unique address. However, in address decoding, multiple addresses are assigned to a single port if some address lines are not used in the decoding circuitry. In this case, all the aliases refer to the same device just like a person with several names (aliases).
55. the linear select
56. Memory-mapped I/O uses the memory address space as I/O ports.
57. True
58. \textit{IOR}
59. \textit{IOW}
60. The diagram follows.

61. The diagram follows.
62. The following circuit shows an output circuit for Figure 0-30.

63. Here is the drawing:
Section 0.6: CPU and Harvard Architecture

64. PC (Program Counter)
65. ALU
66. Address, control and data
CHAPTER 1: THE 8051 MICROCONTROLLERS

Section 1.1: Microcontrollers and Embedded Processors

1. False: A general-purpose microprocessor does not have on-chip ROM.
2. True: A microcontroller has on-chip ROM.
3. True: A microcontroller has on-chip I/O ports.
4. True: A microcontroller has a fixed amount of RAM on the chip.
5. CPU, RAM, ROM, I/O, Timer, Serial COM port
6. RAM and ROM
7. Keyboard, mouse, printer
8. Computing power and compatibility with millions and millions of PCs
9. 8051 - Intel, 6811 – Freescale (Motorola), Z8 – Zilog, PIC 16x – Microchip Technology
10. 8051
11. Power consumption
12. The ROM area is where the executable code is stored
13. Very, in case there is a shortage by one supplier
14. Suppliers other than the manufacturer of the chip
15. Only A is true, 8 bit software will run on a 16 bit system

Section 1.1: Overview of the 8051 Family

16. 4096 bytes
17. 128 bytes
18. 2 timers
19. 256 bytes
20. 8031
21. 32 pins
22. 1 serial port
23. UV-EPROM
24. EEPROM
25. NV-RAM
26. Flash
27. (a) 4K ROM, 128 Bytes of RAM
   (b) 16K ROM, 256 Bytes RAM
   (c) 32K ROM, 256 Bytes RAM
28. The OTP version of the 8051
29. The 8031 does not have on-chip ROM.
30. DS89C420/430 are the best for home development, because you can program it serially using IBM PC.
CHAPTER 2: 8051 ASSEMBLY LANGUAGE PROGRAMMING

Section 2.1: Inside the 8051

1. 8
2. 8
3. 8
4. DPTR or PC (program counter)
5. Necessary
6. 28H, A
7. (a), (d), (f), and (g)
8. (a), (c), (d), (f), and (g)
9. 44H, A
10. 1EH, A

Section 2.2: Introduction to Assembly Programming and

Section 2.3: Assembling and Running an 8051 Program

11. Low, high
12. Assembly language
13. Assembler program
14. True
15. List file
16. False
17. True
18. No
19. They are not real code (real code produces op code). Pseudo-instructions only give instruction to the assembler and does not generate opcodes.
20. True

Section 2.4: The Program Counter and ROM Space in the 8051

21. 0000H
22. It executes whatever is at location 0000h which could be garbage in this case.
23. (a) 2 bytes
   (b) 2 bytes
   (c) 1 byte
   (d) 2 bytes
   (e) 1 byte
   (f) 1 byte
   (g) 1 byte
24.  

<table>
<thead>
<tr>
<th>Location</th>
<th>Instruction</th>
<th>Next Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>ORG 0h</td>
<td>0000</td>
</tr>
<tr>
<td>0000</td>
<td>7D25</td>
<td>MOV R5, #25h</td>
</tr>
<tr>
<td>0002</td>
<td>7F34</td>
<td>MOV R7, #34h</td>
</tr>
<tr>
<td>0004</td>
<td>7400</td>
<td>MOV A, #0</td>
</tr>
<tr>
<td>0006</td>
<td>2D</td>
<td>ADD A, R5</td>
</tr>
<tr>
<td>0007</td>
<td>2F</td>
<td>ADD A, R7</td>
</tr>
<tr>
<td>0008</td>
<td>2412</td>
<td>ADD A, #12h</td>
</tr>
<tr>
<td>000A</td>
<td>80FE</td>
<td>SJMP HERE</td>
</tr>
<tr>
<td>000C</td>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>

25.  
(a) 3FFFh (16K)  
(b) 1FFFh (8K)  
(c) 7FFFh (32K)  
(d) 1FFFh (8K)  
(e) 0FFFh (4K)  
(f) 0FFFh (4K)  
(g) FFFFh (64K)

26. 0000h, FFFFh  
27. 32K  
28. 1K

Section 2.5: 8051 Data Types and Directives

29.  
<table>
<thead>
<tr>
<th>Location</th>
<th>Instruction</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0200</td>
<td>ORG 200H</td>
<td></td>
</tr>
<tr>
<td>0202</td>
<td>45617274</td>
<td>MYDATA_1: DB &quot;Earth&quot;</td>
</tr>
<tr>
<td>0206</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>0209</td>
<td>3938372D</td>
<td>MYDATA_2: DB &quot;987-65&quot;</td>
</tr>
<tr>
<td>020B</td>
<td>3635</td>
<td></td>
</tr>
<tr>
<td>020F</td>
<td>47414245</td>
<td>MYDATA_3: DB &quot;GABEH 98&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END</td>
</tr>
</tbody>
</table>

30.  
<table>
<thead>
<tr>
<th>Location</th>
<th>Instruction</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0340</td>
<td>ORG 340H</td>
<td></td>
</tr>
<tr>
<td>0342</td>
<td>16569920</td>
<td>DAT_1: DB 22,56H,10011001B,32,0F6H,1111011B</td>
</tr>
<tr>
<td>0344</td>
<td>F6FB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END</td>
</tr>
</tbody>
</table>

All in hex where contents of each location are:  
340=(22),341=(56),342=(99),343=(20),344=(F6),345=(FB)
Section 2.6: 8051 Flag Bits and the PSW Register

31. 8
32. D7, D6
33. D2, D0
34. when there is a carry from D7
35. when there is a carry from D3 to D4
36. CY = 1
37. (a) CY = 1
   (b) CY = 0
   (c) CY = 0
38.

```assembly
ORG 0H
MOV R0, #5
MOV A, #0
HERE:
  ADD A, #55h
DJNZ R0, HERE
END
```

Section 2.7: 8051 Register Banks and Stack

39. D3, D4
40. 08
41. Bank 1
42. 8 Bits
43. Bank 0
44. Bank 0: 00h-07h, Bank 1: 08h-0Fh, Bank 2: 10h-17h, Bank 3: 18h-1Fh
45. (a) RAM Location 04
   (b) RAM Location 00
   (c) RAM Location 07
   (d) RAM Location 05
46. (a) RAM Location 14h
   (b) RAM Location 10h
   (c) RAM Location 17h
   (d) RAM Location 15h
47. SETB PSW.4
48.

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>SP after execution of the instruction</th>
<th>Contents of the Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG 0</td>
<td>07H</td>
<td>?</td>
</tr>
<tr>
<td>MOV R0,#66H</td>
<td>07H</td>
<td>?</td>
</tr>
<tr>
<td>MOV R3,#7FH</td>
<td>07H</td>
<td>66H</td>
</tr>
<tr>
<td>MOV R7,#5DH</td>
<td>08H</td>
<td>7FH</td>
</tr>
<tr>
<td>Instruction</td>
<td>SP after execution of the instruction</td>
<td>Contents of the Stack</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Push 0</td>
<td>0AH</td>
<td>5DH</td>
</tr>
<tr>
<td>Push 3</td>
<td>0AH</td>
<td>5DH</td>
</tr>
<tr>
<td>Push 7</td>
<td>0AH</td>
<td>5DH</td>
</tr>
<tr>
<td>Clr A</td>
<td>0AH</td>
<td>5DH</td>
</tr>
<tr>
<td>Mov R3,A</td>
<td>0AH</td>
<td>5DH</td>
</tr>
<tr>
<td>Mov R7,A</td>
<td>0AH</td>
<td>5DH</td>
</tr>
<tr>
<td>Pop 3</td>
<td>09H</td>
<td>7FH</td>
</tr>
<tr>
<td>Pop 7</td>
<td>08H</td>
<td>66H</td>
</tr>
<tr>
<td>Pop 0</td>
<td>07H</td>
<td>7FH</td>
</tr>
</tbody>
</table>

49. NO, pop 7
pop 3
pop 0

50. Instruction SP after execution of the instruction Contents of the Stack

SECTION 2.8: RISC ARCHITECTURE

51. RISC stands for "Reduced Instruction Set Computer". CISC stands for "Complex (or Complete) Instruction Set Computer".

52. CISC

53. RISC

54. RISC

55. CISC
CHAPTER 3: JUMP, LOOP, AND CALL INSTRUCTIONS

Section 3.1: Loop and Jump Instructions

1. 256
2. the instruction following the jump
3. PC
4. short jump, 2
5. long jump, 3
6. less ROM space is consumed
7. True
8. False
9. c, LJMP is not a short jump.
10. 2, because it can only go –128 to 127 bytes in the program
11. True
12. 

```
MOV R6, #10
BACK: MOV R5, #100
HERE: DJNZ R5, HERE
       DJNZ R6, BACK
```

13. 

```
MOV R6, #10
BACK: MOV R5, #100
AGAIN: MOV R4, #100
HERE: DJNZ R4, HERE
       DJNZ R5, AGAIN
       DJNZ R6, BACK
```

14. 20,000
15. 128
16. 127

Section 3.2: Call Instructions

17. 3
18. 2
19. 2K
20. 64K
21. 2 bytes
22. 2 bytes
23. They need to be equal (in order for the stack to remain unchanged) so that when the RET instruction is executed it goes back to place where it originated; otherwise, it returns to a wrong place.

24. When POP is executed, the contents of the stack are copied to the destination register and SP is decremented by one.

25. SP   Stack content
09   (00)
08   (0E)

26. SP   Stack content
0B   (67)
0A   (99)
09   (02)
08   (0B)

Section 3.3: Time Delay Generation for Various 8051 Chips

27. 10 MHz
28. 0.667 μs
29. 1 μs
30. 0.480 μs
31. True, both instructions take 2 machine cycles to execute.
32. 813.75 μs
33. 750 μs
34. \((4 \times 250) \times (200 \times 100)) \times 1.085 \text{ μs} = 21.7 \text{ seconds}\)
35. \((250 \times 150) \times 5 \times 0.75 \text{ μs} = 140625 \text{ μs}\)
36. Duration of each machine cycle: \(1/(11.0592\text{MHz}) = 90.422 \text{ ns}\)
   Number of machine cycles: \((1 + 1 + 1 + 4) \times 150 \times 90.422\text{ns} = 94.944 \text{ μs}\)
37. \(1 / 16\text{MHz} = 62.5 \text{ ns}\)
   \((1+1+1+4) \times 200 \times 62.5 \text{ ns} = 87.5\text{μs}\)
38. \((1+1+4) \times 250 \times 200 \times 100 \times 90.422 \text{ ns} = 2.713 \text{ seconds}\)
39. \((7 \times 250) \times 150 \times 62.5 \text{ ns} = 16.406 \text{ ms}\)
40. Although the period of each machine cycle is twelve times shorter in DS89C420/30, the fact that the execution of many instructions take more number of machine cycles than in AT89C51, does not allow the performance to become twelve times faster in DS89C420/30. As an example, if you divide the answer of Problem 35 by the answer of Problem 39, we can figure out that in this particular case, DS89C420/30 is almost 8.6 times faster than AT89C51, not 12 times faster.
CHAPTER 4: I/O PORT PROGRAMMING

Section 4.1: 8051 I/O Programming

1. 40
2. Pins 40, 20 are assigned to Vcc and Gnd, respectively.
3. 32
4. 8 Pins, 39-32, are designated for P0.
5. 8 Pins, 01-08, are designated for P1.
6. 8 Pins, 21-28, are designated for P2.
7. 8 Pins, 10-17, are designated for P3.
8. output
9. P0
10. P1

11. ORG 0000h
    MOV A, #0FFH
    MOV P1, A
    MOV A, P1
    MOV P0, A
    MOV P2, A
    MOV P3, A
    HERE: SJMP HERE
    END

12. ORG 0000h
    MOV A, #0FFH
    MOV P2, A
    MOV A, P2
    MOV P0, A
    MOV P1, A
    HERE: SJMP HERE
    END

13. 10 and 11
14. 0000h; This means that the starting of the code must begin at location 0000h. In other words it belongs to code ROM.
Section 4.2: I/O Bit Manipulation Programming

16. All ports of the 8051 are bit addressable.
17. The advantage of the bit addressing is that it allows each bit to be modified without affecting the other bits.
18. P1.x
19. No, CPL P1 is not a valid instruction because CPL works with register A and individual bits.
20. ORG 0000h
    AGAIN:
    SETB P1.2
    SETB P1.5
    ACALL DELAY
    CLR P1.2
    CLR P1.5
    ACALL DELAY
    SJMP AGAIN
    END
21.

```
ORG 0000h
AGAIN:
SETB P1.3
SETB P1.7
SETB P2.5
ACALL DELAY
CLR P1.3
CLR P1.7
CLR P2.5
ACALL DELAY
SJMP AGAIN
END
```

22.

```
ORG 0000h
SETB P1.3
AGAIN:
JNB P1.3, AGAIN
MOV A, #55h
MOV P2, A
HERE: SJMP HERE
END
```

23.

```
ORG 0000h
SETB P1.3
AGAIN:
JB P1.3, AGAIN
HERE:
MOV A, #55h
MOV P0, A
ACALL DELAY
MOV A, #0AAh
MOV P0, A
ACALL DELAY
SJMP HERE
END
```
24. ORG 0000h
    SETB P2.0
    AGAIN:
        JB   P2.0, HERE
        MOV  A, #66H
        MOV  P1, A
        SJMP AGAIN
    HERE:
        MOV  A, #99H
        MOV  P1, A
        SJMP AGAIN
    END

25. ORG 0000h
    SETB P1.5
    AGAIN:
        JNB P1.5, AGAIN
        CLR  P1.3
        SETB P1.3
        CLR  P1.3
    HERE: SJMP HERE
    END

26. ORG 0000h
    SETB P1.3
    AGAIN:
        JB   P1.3, HERE
        CLR  P1.4
        SJMP AGAIN
    HERE: SETB P1.4
    SJMP AGAIN
    END

27. The fifth bit of port 1.
ORG 0000H

SETB P1.7
SETB P1.6

AGAIN:
   JB P1.7, HERE
   CLR P1.0
   SJMP NEXT

HERE:
   SETB P1.0

NEXT:
   JB P1.6, COPY
   CLR P1.7
   SJMP AGAIN

COPY:
   SETB P1.7
   SJMP AGAIN
END
CHAPTER 5: 8051 ADDRESSING MODES

Section 5.1: Immediate and Register Addressing Modes
Section 5.2: Accessing Memory using Various Addressing Modes

1. (b)
2. (a) immediate   (g) direct addressing
  (b) direct addressing  (h) direct addressing
  (c) direct addressing  (i) register indirect
  (d) immediate   (j) direct addressing
  (e) direct addressing  (k) register indirect
  (f) immediate
3. (a) 00h    (g) 0Ch
  (b) 0E0h   (h) 82h
  (c) 07h    (i) 0Eh
  (d) 13h    (j) 18h
  (e) 0F0h   (k) 83h
  (f) 1Fh    (l) 80h
4. Bank 1
5. direct
6. “MOV A, 0F0h” moves the value stored in the B register into register A.
7. “MOV A, 1Fh” moves the value of R7 of bank 3 into register A.
8. 

```
PUSH 00h
PUSH 01h
PUSH 03h

POP 1Fh
POP 1Eh
POP 1Dh
```

9. R0 and R1
10. 
```
MOV A, #0FFh
MOV R1, #50h
MOV R7, #20h
AGAIN:
  MOV @R1, A
  INC R1
  DJNZ R7, AGAIN
```
11.

```
ORG 0000h
MOV DPTR, #400h
MOV R7, #10
MOV R1, #30h
AGAIN:
  CLR A
  MOVC A, @A+DPTR
  MOV @R1, A
  INC R1
  INC DPTR
  DJNZ R7, AGAIN
END
```

12.

```
ORG 0000h
MOV DPTR, #250h
MOV A, #0FFh
MOV P1, A
AGAIN:
  MOV A, P1
  MOVC A, @A+DPTR
  MOV P2, A
  SJMP AGAIN
ORG 250h
TABLE: DB 5,8,13,20,29,40,53,68,85,104
END
```

13.

```
ORG 0000h
MOV DPTR, #250h
CLR A
MOV 30H, A
MOV R0, #5
AGAIN:
  CLR A
  MOVC A, @A+DPTR
  INC DPTR
  ADD 30H, A
  DJNZ AGAIN
HERE: SJMP HERE
ORG 250h
TABLE: DB 06,09,02,05,07
END
```
Section 5.3: Bit Addresses for I/O and RAM

14. invalid
15. valid
16. valid
17. All of the ports
18. B, PSW, IP, IE, ACC, TCON, SCON (and all the ports P0 - P3)
19. (a) Invalid     (e) D4 of B
(b) D3 of P2      (f) P0.0
(c) D5 of A       (g) D3 of PSW
(d) P1.0          (h) P0.7

20. HERE:
    SETB P1.5
    LCALL DELAY
    LCALL DELAY
    LCALL DELAY
    CLR P1.5
    LCALL DELAY
    SJMP HERE

21. HERE:
    SETB P2.7
    LCALL DELAY
    LCALL DELAY
    LCALL DELAY
    LCALL DELAY
    CLR P2.7
    LCALL DELAY
    SJMP HERE

22. CLR P2.7
    SETB P1.4
HERE:
    JNB P1.4, HERE
    SETB P2.7
    LCALL DELAY
    CLR P2.7
    LCALL DELAY
    SJMP HERE

23. SETB P2.1
HERE:
    JB P2.1, HERE
    MOV P0,#55H

24
24. 80h – 87h
25. 90h – 97h
26. A0h – A7h
27. B0h – B7h
28. None
29. 88h – 8Fh
30. E0h – E7h
31. F0h – F7h
32. D0h – D7h
33. (a) P0.5  
    (b) P0.7  
    (c) TCON.0  
    (d) TCON.5  
    (e) P1.3  
    (f) P2.5  
    (g) P2.7  
    (h) P3.3  
    (i) PSW.4  
    (j) PSW.7  
    (k) D3 of B
34. 
   CLR PSW.3  
   SETB PSW.4  
   MOV R3, A  
   MOV R5, B
35. CLR PSW.7
36. The difference between the two, is that the CLR C command is a 1 byte instruction, whereas the CLR PSW.7 is a two byte instruction.
37. JNB PSW.2,...
38. JB PSW.7,...
39. JB PSW.0,...
40. JB PSW.6
41. CY = D7H  
    P = D0H  
    AC = D6H  
    OV = D2H
42. 16, 20h – 2Fh
43. 00h – 7Fh
44. 00h – 7Fh
45. All the SFR are located in the address space of 80h – FFh, but not all of these addresses are used. The unused portion is reserved for future.
46. (a) 00-7Fh  
    (b) b 80-FFh
   There are five gaps in the SFR portion. By referring to figure 5.2, these gaps are: C0h~C7h, C8~CFh, D8h~DFh, E8h~EFh, and F8h~FFh. These gaps are reserved for future improvements like T2CON in 8052 which occupies C8h~CFh bit locations.
47. They belong to byte address space of 20h-2Fh as shown below:
   (a) D5 of byte 20h  
   (b) D7 of byte 28h  
   (c) D0 of byte 23h  
   (g) D7 of byte 2Ch  
   (h) D5 of byte 2Ah  
   (i) D4 of byte 22h
(d) D5 of byte 25h  (j) D7 of byte 26h
(e) D3 of byte 2Ah  (k) D7 of byte 2Fh
(f) D5 of byte 22h

48. True
49. True
50. MOV 04, C

51. MOV C, PSW.6
   MOV 16H, C

52. MOV C, PSW.0
   MOV 12H, C

53. JNB ACC.0, OVER
    JNB ACC.1, OVER
    MOV B, #4
    DIV AB
    OVER:

54. MOV C, ACC.7
    JNC EXIT
    MOV DPTR, #300h
    LCALL DISPLAY

EXIT: SJMP EXIT

ORG 300h
DATA1: DB "A Negative Number"

55. MOV A,B
    JB ACC.7, OVER
    ACALL DISPLAY
    ...
    ...
    OVER:
Section 5.4: Extra 128-Byte On-Chip RAM in 8052

56. MOV R1, #10h
    MOV A, #0FFh
    MOV R0, #20h
    KEEP: MOV @R0, A
    INC R0
    DJNZ R1, KEEP

57.  JB ACC.0, OVER
    JB ACC.1, OVER
    JB ACC.2, OVER
    MOV B, #8
    DIV AB
    ........

OVER:

58. MOV R4, #8
    MOV R3, #8
    MOV A, R2
    HERE:
    RRC A
    JC, NEXT
    INC R4
    NEXT: DJNZ R3, HERE
    ;NOW R4 Contains the number of zeros

59. 256 bytes of RAM + 128 SFR bytes = 384 bytes. In 8051, we have only 256 bytes (128 byte SFR + 128 byte RAM).
60. Direct addressing mode.
61. Register indirect addressing mode.
62. Lower 128-bytes: 00h-7Fh
    Upper 128-bytes: 80h-FFh
63. Physically they are two different memory spaces, otherwise it would be meaningless to say that there is another 128-byte memory space in 8052. And that is why we need to use two different addressing modes to access each of these memory spaces.
64. (a) Puts 99h in the address 80h of the SFRs (which is P1)
    (b) Writes 99h into the address 80h of the upper 128-byte memory space (which is the first byte of the upper part of memory)
65. R1 and R0
67.

```
ORG 0000H
MOV R0,#0C0H
MOV R2,#16
MOV A,#55H
AGAIN:
  MOV @R0,A
  INC R0
  DJNZ R2,AGAIN

SJMP $
END
```

68.

```
ORG 0000H
MOV R0,#60H
MOV R1,#0D0H
MOV R2,#16
AGAIN:
  MOV A,@R0
  MOV @R1,A
  INC R0
  INC R1
  DJNZ R2,AGAIN

SJMP $
END
```
CHAPTER 6: ARITHMETIC, LOGIC INSTRUCTIONS, AND PROGRAMS

Section 6.1: Arithmetic Instructions

1. (a) CY=0, AC=1
   (b) CY=0, AC=1
   (c) CY=1, AC=0
   (d) CY=1, AC=1
   (e) CY=1, AC=0
   (f) CY=0, AC=0

2.

```
ACALL TRANSF
ACALL ADD_IT
...
TRANSF:
  MOV  DPTR, #300H
  MOV  R3, #09H
  MOV  R0,#40H
BACK: CLR  A
  MOVC  A, @A+DPTR
  MOV   @R0,A
  INC R0
  INC DPTR
  DJNZ R3,BACK
  RET
;-----------------
ADD_IT:
  CLR A
  MOV  R3,#9
  MOV  R0,#40H
RE:  ADD A,@R0
  DA A
  INC R0
  DJNZ R3,RE
  MOV  R3,A
  RET
ORG 300H
NUMBERS:    DB 9,9,9,9,9,9,6,8,2
```
3.

; The transfer part is the same as in Problem 2
; ADD_IT is modified as follows:
ADD_IT:
   CLR A
   MOV B,A
   MOV R3,#9
   MOV R0,#40H
   RE:  ADD A,@R0
        JNC OVER
        INC B
   OVER:
        INC R0
        DJNZ R3,RE
        MOV R2,A
        MOV R3,B
        RET

4.

; in the add subroutine we simply put
; the “DA A” instruction

ADD_IT:
   CLR A
   MOV B,A
   MOV R3,#9
   MOV R0,#40H
   RE:  ADD A,@R0
        DA A
        JNC OVER
        INC B
   OVER:
        INC R0
        DJNZ R3,RE
        MOV R2,A
        MOV R3,B
        RET
5. (a) Take the 2’s complement of $12_{h}$. Add the result to $23_{h}$. Invert the carry.

\[
\begin{array}{c|c|c}
23 & 0010 0011 & 0010 0011 \\
-12 & 0001 0010 & + 1110 1110 \\
\end{array}
\]

where $1 \rightarrow 0$ for carry

(b) $53 \rightarrow 0101 0011 \rightarrow 1010 1101 \\
+ 0100 0011 \\
0 1111 0000 \rightarrow F0 \\
0 \rightarrow 1$ for carry
The result ($F0_{h}$) is negative, which is the 2’s complement of $10_{h}$

(c) $1001 1001 \rightarrow 0110 0111 \\
+ 1001 1001 \\
1 0000 0000 \\
1 \rightarrow 0$ for carry, and the result is zero.

6. (a) Take the 2’s complement of $12_{h}$. Add the result to $23_{h}$. Invert the carry.

```
MOV R3, #16
MOV R1, #40H
MOV A, #55H
BACK: MOV @R1, A
INC R1
DJNZ R3, BACK

MOV R4, #16
MOV R1, #40H
CLR A
MOV B,A
HERE: ADD A, @R1
JNC OVER
INC B
OVER: INC R1
DJNZ R4, HERE
MOV 60H,A
MOV 61H,B
```
7. 

```asm
; (a) --
ORG 0H
CLR C
MOV A, #23H
MOV R2, #12H
SUBB A, R2
; (b) -----------------
CLR C
MOV A, #43H
MOV R2, #53H
SUBB A, R2
; (c) ------------------
CLR C
MOV A, #99
MOV R2, #99
SUBB A, R2
```

8. True, it must be used after the ADD or ADDC instruction.

9. 

```asm
CLR C
MOV A, #9AH
ADDC A, #48H
MOV 40H, A
MOV A, #7FH
ADDC A, #0BCH
MOV 41H, A
MOV A, #19H
ADDC A, #34H
MOV 42H, A
```

10. 

```asm
CLR C
MOV A, #48H
SUBB A, #9AH
MOV 40H, A
MOV A, #0BC
SUBB A, #7FH
MOV 41H, A
MOV A, #34H
SUBB A, #19H
MOV 42H, A
```
11. 

```
CLR C
MOV A, #95H
ADDC A, #48H
DA A
MOV 40H, A
MOV A, #77H
ADDC A, #45H
DA A
MOV 41H, A
MOV A, #19H
ADDC A, #34H
DA A
MOV 42H, A
```

12. 

```
MOV A, #77
MOV B, #34
MUL AB
```

13. 

```
MOV A, #77
MOV B, #3
DIV AB
```

14. False, it only works with A and B.
Section 6.2: Signed Number Concepts and Arithmetic Operations

16. (a) 1110 1001
(b) 0000 1100
(c) 1110 0100
(d) 0110 1111
(e) 1000 0000
(f) 0111 1111

17. unsigned

18. 

<table>
<thead>
<tr>
<th>(a) MOV A, #15</th>
<th>(b) MOV A, #123</th>
<th>(c) MOV A, #25H</th>
<th>(d) MOV A, #127</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV R1, #12</td>
<td>MOV R1, #127</td>
<td>MOV R1, #34H</td>
<td>MOV R1, #127</td>
</tr>
<tr>
<td>ADD A, R1</td>
<td>ADD A, R1</td>
<td>ADD A, R1</td>
<td>ADD A, R1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OV = 0</th>
<th>OV = 1</th>
<th>OV = 0</th>
<th>OV = 0</th>
</tr>
</thead>
</table>
19. The CY flag is raised when there is carry beyond bit D7 and used for unsigned numbers. The OV is raised when there is a carry from D6 to D7 but not from D7 to carry. Or when there is a carry from D7 to carry but not from D6 to D7 and it is used for signed numbers.

20. The OV flag is raised when there is a carry from D6 to D7 but not from D7 to carry. Or when there is a carry from D7 to carry but not from D6 to D7.

21. PSW.2

22. The OV flag is detected using the instruction “JNB PSW.2,...” The carry flag can be detected by the JNC instruction.

Section 6.3: Logic and Compare Instructions

23. (a) 40h
    (b) F6h
    (c) 86h
    (d) 90h
    (e) 60h
    (f) F0h
    (g) F0h
    (h) F9h
    (i) 1Eh
    (j) 5Ah

24. (a) 64h
    (b) 7Bh
    (c) 3Fh
    (d) 58h
    (e) D7h
    (f) 04h
    (g) 37h

25. False, we do not have to use the accumulator.

26. Yes

27. No, there is not a CJE instruction.

28. (a) CY=1
    (b) CY=0
    (c) CY=0
    (d) CY=0
    (e) CY=1
    (f) CY=0

29. (a) Yes
    (b) Yes
    (c) No
    (d) No
    (e) Yes
    (f) No
Section 6.4: Rotate Instruction and Data Serialization

30. (a) 01011001b ; 59h
    (b) 11100100b; E4h
    (c) 00011010b; 1Ah
    (d) 10011111b; 9Fh

31. 
   (a) RR A
       RR A
       RR A
       RR A
   (b) RL A
       RL A
       RL A
       RL A

32. 
   MOV R1, #0
   MOV R7, #8
   MOV A, VALUE
   CLR C
   AGAIN: RLC A
   JC NEXT
   INC R1
   NEXT: DJNZ R7, AGAIN

33. 
   MOV R1, #0
   MOV R7, #8
   MOV A, #68h
   CLR C
   AGAIN:
   RRC A
   JC FIRST
   INC R1
   DJNZ R7, AGAIN
   FIRST:
   ;NOW R1=3 FOR 68h.

34. 
   MOV R1, #7 ;FOR D7 BIT
   MOV R7, #8 ;COUNTER
   MOV A, #68h
   CLR C
   AGAIN:
   RLC A
   JC FIRST
   DEC R1
   DJNZ R7, AGAIN
   FIRST:
   ;NOW R1=6 FOR 68H
35.

```assembly
MOV A, #0CCh
KEEPGOING:
    RR A
    MOV P1, A
    SJMP KEEPGOING

;There is no need for the ;top nibble, it is just for convenience.
```

Section 6.5: BCD, ASCII and Other Application Programs

36.

```assembly
ACALL TRANSFER
ACALL CONVERT
```

```assembly
TRANSFER:
    MOV DPTR, MYDATA
    MOV R1, #60h
    MOV R3, #4h
NEXT: CLR A
    MOVC A, @A+DPTR
    MOV @R1, A
    INC R1
    INC DPTR
    DJNZ R3, NEXT
    RET

;---------------
CONVERTING ;FROM PACKED BCD TO ASCII
CONVERT:
    MOV R1,#60H
    MOV R0,#40H
    MOV R3,#4
BACK: MOV A,@R1
    ANL A, #0Fh
    ORL A,#30H
    MOV @R0,A
    INC R0
    MOV A,@R1
    ANL A, #0F0h
    SWAP A
    ORL A,#30H
    MOV @R0,A
    INC R0
    INC R1
    DJNZ R3, BACK
    RET

;---------------
ORG 300h
MYDATA: DB 76H,87H,98H,43H
ACALL TRANSFER
ACALL CONVERT

TRANSFER:
MOV DPTR, MYDATA
MOV R1, #40h
MOV R3, #8h
NEXT: CLR A
      MOVC A, @A+DPTR
      MOV @R1,A
      INC R1
      INC DPTR
      DJNZ R3, NEXT
      RET

CONVERTING ;FROM ASCII TO PACKED BCD
CONVERT:
MOV R1,#40H
MOV R0,#60H
MOV R3,#4
BACK: MOV A,@R1
      ANL A, #0Fh
      MOV B,A
      INC R1
      MOV A,@R1
      ANL A, #0Fh
      SWAP A
      ORL A,B
      MOV @R0,A
      INC R0
      INC R1
      DJNZ R3, BACK
      RET

ORG 300h

MOV A, #0FFH
MOV P1, A
MOV A, P1
MOV B, #10
DIV AB
MOV R0, B
MOV B, #10
MOV 40H, R0
DIV AB
MOV R0, B
MOV 41H, R0
MOV 42H, A
ORL 40H, #30H
ORL 41H, #30H
ORL 42H, #30H

;Since 8D=141, we have 31h,34h, ;and 31h.
39. Point 1 if $A = 50$
   Point 2 if $A > 50$
   Point 3 if $A < 50$

40.

\[
\text{ORG 0000h} \\
\text{MOV A, #0FFh} \\
\text{MOV P1, A} \\
\text{MOV A, P1} \\
\text{ANL A, #0FH} \\
\text{MOV DPTR, TABLE} \\
\text{MOVC A, @A+DPTR} \\
\text{MOV P2, A} \\
\text{SJMP $} \\
\text{END} \\
\text{TABLE:} \\
\text{DB “0123456789ABCDEF”}
\]

41. ‘H’ = 48h , ‘e’ = 65h , ‘l’ = 6Ch , ‘o’ = 6Fh
   $48h + 65h + 6Ch + 6Ch + 6Fh = 1F4h \rightarrow \text{F4h} \rightarrow \text{Check-sum byte = 0Ch}$

42. True ( The carry should be discarded )

43.

\[
\text{ORG 0000H} \\
\text{ACALL LOAD} \\
\text{ACALL CALCULATE} \\
\text{ACALL TEST} \\
\text{JNZ ERROR} \\
\text{... ERROR:} \\
\text{...} \\
\text{;----------------------} \\
\text{LOAD:} \\
\text{MOV R0, #30H} \\
\text{MOV DPTR, SENTENCE} \\
\text{NEXT1:} \\
\text{CLR A} \\
\text{MOVC A, @A+DPTR} \\
\text{MOV @R0, A} \\
\text{JZ EXIT1} \\
\text{INC R0} \\
\text{INC DPTR} \\
\text{SJMP NEXT1} \\
\text{EXIT1:} \\
\text{RET} \\
\text{;----------------------} \\
\text{CALCULATE:} \\
\text{MOV R0, #30H} \\
\text{CLR A} \\
\text{NEXT2:} \\
\text{ADD A, @R0} \\
\text{;--CONTINUED ON NEXT PAGE--}
\]
40. (a) Modules can easily be used in other programs when needed.
   (b) It is easier to find a bug in the program when it is in a modular format.
   (c) It is much easier to break a heavy task into smaller tasks and get them done one by one in modules.

45. ASCII

46. ORG 0000H
    MOV A, #0FFH
    MOV P1, A
    MOV A, P1
    ANL A, #0FH
    CLR C
    SUBB A, #9
    JNC CHARACTER
    ADD A, #9
    ORL A, #30H
    MOV P2, A
    SJMP EXIT

    CHARACTER:
    ORL A, #40H
    MOV P2, A

    EXIT:
    ...