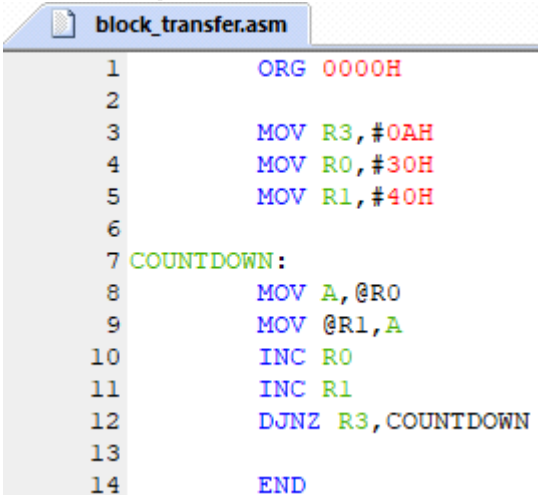


Date: 21/02/2021

| | |
|-------------------|-----------------------|
| Student's name | Anuj Shah |
| Roll Number | 18104B0024 |
| Name of Professor | Professor Mohit Gujar |

| | |
|----------------------|-----------------------------------|
| Experiment number | 2 |
| Experiment title | Block transfer and Block exchange |
| Hardware requirement | - |
| Software requirement | Keil uVision5 |

| | |
|-----------------------|--|
| Aim | To write a program to copy 10 numbers stored from 30H memory location to memory location starting from 40H |
| Theory | <p>Our primary objective in these programs is to transfer the data stored in memory locations (0x30, 0x31, 0x32, ..., 0x3A) to the memory locations (0x40, 0x41, 0x42, ..., 0x4A). There are two ways to do this:</p> <ul style="list-style-type: none">• Block transfer, which carries out one-directional flow of data; that is, only from (0x30, 0x31, 0x32, ..., 0x3A) to (0x40, 0x41, 0x42, ..., 0x4A).• Block exchange, which carries out two-directional flow of data; that is, both from (0x30, 0x31, 0x32, ..., 0x3A) to (0x40, 0x41, 0x42, ..., 0x4A); and also from (0x40, 0x41, 0x42, ..., 0x4A) to (0x30, 0x31, 0x32, ..., 0x3A). <p>Both the programs make heavy use of Pointers (Indirect addressing).</p> <p>Both the programs make use of a conditional jump statement called <u>DJNZ</u> (<u>D</u>ecrement register, and <u>J</u>ump if that register is <u>N</u>ot <u>Z</u>ero).</p> <p>The block exchange program also made use of the "XCH A,register" command, which is used to exchange the value of register A with the value contained in register.</p> <p>Source: https://www.win.tue.nl/~aeb/comp/8051/set8051.html#51xch</p> |
| Algorithm / Flowchart | <p>Block transfer:</p> <ol style="list-style-type: none">1. Because we want to transfer 10 numbers (10 in decimal = 0xA in hex) from address 0x30 to address 0x40, that is why we write the commands "MOV R3,#0AH", "MOV R0,#30H" and "MOV R1,#40H"2. "MOV A,@R0" copies the contents of address 0x30 to register A3. "MOV @R1,A" copies the contents of register A to address 0x404. "INC R0" changes 0x30 to 0x315. "INC R1" changes 0x40 to 0x416. "DJNZ R3,COUNTDOWN" changes 0xA to 0x9, and because 0x9 is not |

| | |
|---------|--|
| | <p>zero, that is why our program jumps back to step 2.</p> <p>7. Steps 2-6 are repeated continuously, until R3 finally becomes 0x0.</p> <p>Thus, our program <u>copies</u> the contents from address 0x30 to address 0x40; from 0x31 to 0x41; and so on, until 0x3A to 0x4A.</p> <p>Block exchange:</p> <ol style="list-style-type: none"> 1. Because we want to transfer 10 numbers (10 in decimal = 0xA in hex) from address 0x30 to address 0x40, that is why we write the commands "MOV R3,#0AH", "MOV R0,#30H" and "MOV R1,#40H" 2. "MOV A,@R0" copies the contents of address 0x30 to register A 3. "XCH A,@R1" exchanges the contents of register A with the contents at address 0x40 4. "MOV @R0,A" copies the contents of register A to address 0x30 5. "INC R0" changes 0x30 to 0x31 6. "INC R1" changes 0x40 to 0x41 7. "DJNZ R3,COUNTDOWN" changes 0xA to 0x9, and because 0x9 is not zero, that is why our program jumps back to step 2. 8. Steps 2-6 are repeated continuously, until R3 finally becomes 0x0. <p>Thus, our program <u>exchanges</u> the contents of addresses 0x30 and 0x40; the contents of 0x31 and 0x41; the contents of 0x32 and 0x42; and so on, until 0x3A and 0x4A.</p> <p>Source: https://www.keil.com/support/man/docs/is51/is51_mov.htm</p> |
| Program | <p>Block transfer:</p>  <pre> 1 ORG 0000H 2 3 MOV R3,#0AH 4 MOV R0,#30H 5 MOV R1,#40H 6 7 COUNTDOWN: 8 MOV A,@R0 9 MOV @R1,A 10 INC R0 11 INC R1 12 DJNZ R3,COUNTDOWN 13 14 END </pre> <p>Block exchange:</p> |

```

1  ORG 0000H
2
3  MOV R3, #0AH
4  MOV R0, #30H
5  MOV R1, #40H
6
7  COUNTDOWN:
8      MOV A, @R0
9      XCH A, @R1
10     MOV @R0, A
11     INC R1
12     INC R0
13     DJNZ R3, COUNTDOWN
14
15     END

```

Results /
Output

Block transfer:

C:\Users\Anuj Shah\Documents\GitHub\learning_assembly\8051\exp2- block transfer and block exchange\exp2_block_transfer.uvproj - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

RST [Icons]

Registers Disassembly

| Register | Value |
|----------|------------|
| Regs | |
| r0 | 0x3a |
| r1 | 0x4a |
| r2 | 0x00 |
| r3 | 0x00 |
| r4 | 0x00 |
| r5 | 0x00 |
| r6 | 0x00 |
| r7 | 0x00 |
| Sys | |
| a | 0x00 |
| b | 0x00 |
| sp | 0x07 |
| sp_max | 0x07 |
| PC \$ | C:0x000C |
| auxr1 | 0x00 |
| dptr | 0x0000 |
| states | 63 |
| sec | 0.00002291 |
| psw | 0x00 |

Disassembly

| Address | Hex | Assembly |
|----------|-----|----------|
| C:0x000C | 00 | NOP |
| C:0x000D | 00 | NOP |
| C:0x000E | 00 | NOP |
| C:0x000F | 00 | NOP |

block_transfer.asm

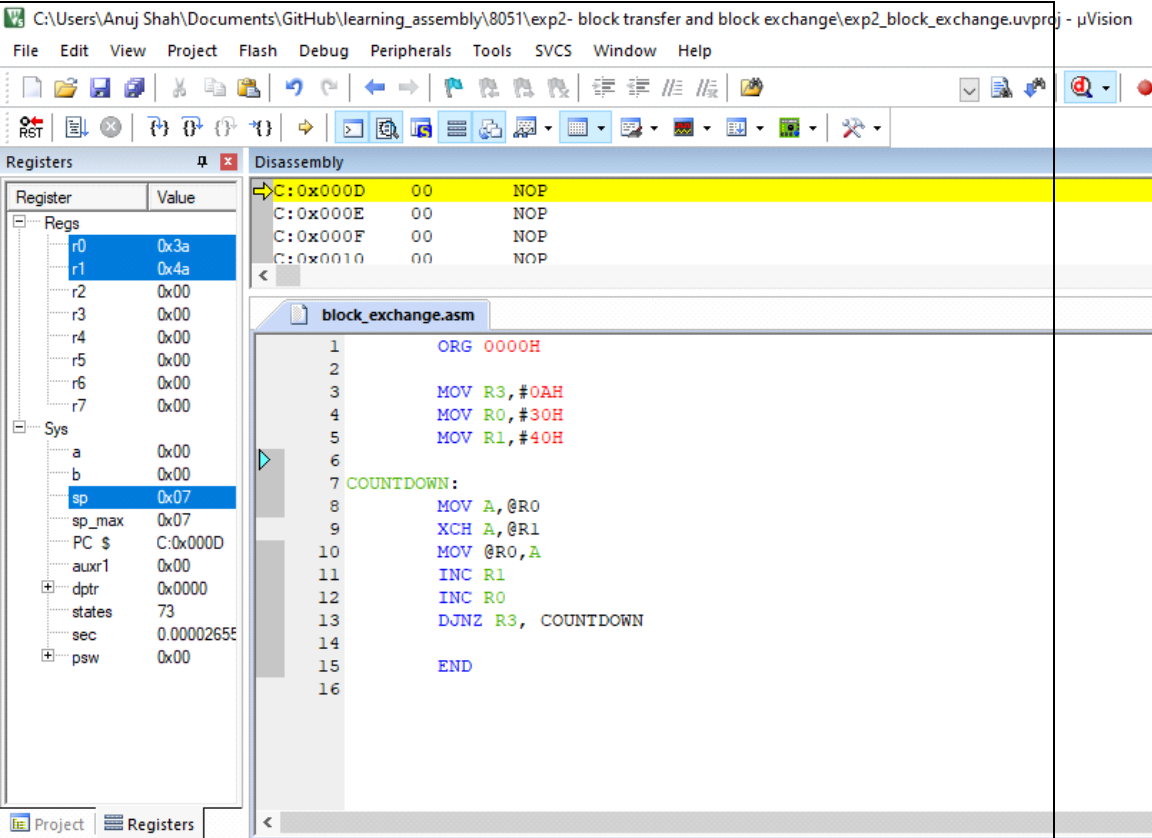
```

1  ORG 0000H
2
3  MOV R3, #0AH
4  MOV R0, #30H
5  MOV R1, #40H
6
7  COUNTDOWN:
8      MOV A, @R0
9      MOV @R1, A
10     INC R0
11     INC R1
12     DJNZ R3, COUNTDOWN
13
14     END

```

Project Registers

Block exchange:

| |  <p>C:\Users\Anuj Shah\Documents\GitHub\learning_assembly\8051\exp2- block transfer and block exchange\exp2_block_exchange.uvproj - µVision</p> <p>File Edit View Project Flash Debug Peripherals Tools SVCS Window Help</p> <p>Registers</p> <table><thead><tr><th>Register</th><th>Value</th></tr></thead><tbody><tr><td>r0</td><td>0x3a</td></tr><tr><td>r1</td><td>0x4a</td></tr><tr><td>r2</td><td>0x00</td></tr><tr><td>r3</td><td>0x00</td></tr><tr><td>r4</td><td>0x00</td></tr><tr><td>r5</td><td>0x00</td></tr><tr><td>r6</td><td>0x00</td></tr><tr><td>r7</td><td>0x00</td></tr><tr><td>sp</td><td>0x07</td></tr><tr><td>sp_max</td><td>0x07</td></tr><tr><td>PC \$</td><td>C:0x000D</td></tr><tr><td>auxr1</td><td>0x00</td></tr><tr><td>dptr</td><td>0x0000</td></tr><tr><td>states</td><td>73</td></tr><tr><td>sec</td><td>0.00002655</td></tr><tr><td>psw</td><td>0x00</td></tr></tbody></table> <p>Disassembly</p> <table><thead><tr><th>Address</th><th>Hex</th><th>Op</th><th>Comment</th></tr></thead><tbody><tr><td>C:0x000D</td><td>00</td><td>NO</td><td>NO</td></tr><tr><td>C:0x000E</td><td>00</td><td>NO</td><td>NO</td></tr><tr><td>C:0x000F</td><td>00</td><td>NO</td><td>NO</td></tr><tr><td>C:0x0010</td><td>00</td><td>NO</td><td>NO</td></tr></tbody></table> <p>block_exchange.asm</p> <pre> 1 ORG 0000H 2 3 MOV R3, #0AH 4 MOV R0, #30H 5 MOV R1, #40H 6 7 COUNTDOWN: 8 MOV A, @R0 9 XCH A, @R1 10 MOV @R0, A 11 INC R1 12 INC R0 13 DJNZ R3, COUNTDOWN 14 15 END 16 </pre> <p>Project Registers</p> | Register | Value | r0 | 0x3a | r1 | 0x4a | r2 | 0x00 | r3 | 0x00 | r4 | 0x00 | r5 | 0x00 | r6 | 0x00 | r7 | 0x00 | sp | 0x07 | sp_max | 0x07 | PC \$ | C:0x000D | auxr1 | 0x00 | dptr | 0x0000 | states | 73 | sec | 0.00002655 | psw | 0x00 | Address | Hex | Op | Comment | C:0x000D | 00 | NO | NO | C:0x000E | 00 | NO | NO | C:0x000F | 00 | NO | NO | C:0x0010 | 00 | NO | NO | |
|------------|---|----------|---------|----|------|----|------|----|------|----|------|----|------|----|------|----|------|----|------|----|------|--------|------|-------|----------|-------|------|------|--------|--------|----|-----|------------|-----|------|---------|-----|----|---------|----------|----|----|----|----------|----|----|----|----------|----|----|----|----------|----|----|----|--|
| Register | Value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| r0 | 0x3a | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| r1 | 0x4a | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| r2 | 0x00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| r3 | 0x00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| r4 | 0x00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| r5 | 0x00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| r6 | 0x00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| r7 | 0x00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| sp | 0x07 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| sp_max | 0x07 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PC \$ | C:0x000D | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| auxr1 | 0x00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dptr | 0x0000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| states | 73 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| sec | 0.00002655 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| psw | 0x00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Address | Hex | Op | Comment | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C:0x000D | 00 | NO | NO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C:0x000E | 00 | NO | NO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C:0x000F | 00 | NO | NO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C:0x0010 | 00 | NO | NO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Conclusion | <p>We learned some interesting features of 8051 assembly language in this experiment:</p> <ul style="list-style-type: none">• Indirect addressing (pointers)• Conditional jumps (eg. DJNZ) and count-downs• XCH, a slight variant of the MOV keyword | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Faculty Sign

Grade received