8051 Programming: Arithmetic and Logic

EE4380 Fall 2002
Class 4

Pari vallal Kannan
Center for Integrated Circuits and Systems
University of Texas at Dallas
Topics

- Signed and Unsigned arithmetic
- Binary and BCD coded numbers
- Addition Instructions
- Subtraction
- Multiplication
- Division
- Logic Operations
- Rotate and Swap operations
- Bit addressable memory and single bit instructions
Unsigned Addition

- add a, source ; A = A + source
- Carry (if any) will be in CY flag
  mov A, #0F5H
  add A, #0BH ; A = F5 + B0 = 00, CY=1
- 16 bit addition
  - addc A, source ; A = A + source + CY
  - Add the lower bytes using add
  - Save the result
  - Add the upper bytes using addc
Unsigned Addition (contd.)

- Example of 16 bit addition
- Add UUVV and PPQQ
  
  ```
  clr C
  mov A, QQ
  add A, VV
  mov r6, A
  mov A, PP
  addc A, UU
  mov r7, A
  ```

- Final 16 bit result in r7:r6 and CY flag
BCD Addition

- BCD – Binary Coded Decimal
  - 4 bits are used to represent a decimal number from 0-9
- Packed BCD has two such numbers in one byte
  - 17 PBCD = 17 decimal = 11hex
- Packed BCD addition may not yield a valid BCD. Use decimal adjust instruction (da A) for correcting it
- After adding two Packed BCD numbers call da to get valid PBCD

```assembly
mov A, #47H ; first BCD = 47d
mov B, #25H ; second BCD = 25d
add A, B ; A = 6CH (binary addition of 47H and 25H)
da A ; A = 72H (BCD result of addition)
```
BCD Addition (contd.)

- To correct an invalid BCD, add 6 to the digit that is greater than 9
- What da does
  - If lower nibble is > 9 or AC=1 then add 6 (0110) to the lower nibble
  - If upper nibble is > 9 or CY=1 then add 6 to the upper nibble
- da will work for ADD only. For other operations (inc, sub etc), this correction has to be done manually
Unsigned Subtraction

- subb x, y ; x = x-y with borrow from CY
- Operation:
  - Take 2’s complement of the subtrahend (y)
  - Add it to the minuend (x)
  - If the CY flag is set after the subb operation, then the result is negative and the destination has the 2’s complement of the result

- subb performs subtract with borrow, if CY is set before the call. Used for 16bit subtraction
  - To get plain sub, clear CY before calling subb
Unsigned Subtraction (contd.)

- **Example**

  ```assembly
  clr c ; clear CY for sub operation
  mov A, #4CH ;
  subb A, #6EH ; two operands, do 4C – 6E
  jnc done ; if CY==0 result is positive
  cpl A ; CY=1, result negative. So find 2’s complement
  inc A ; by complementing A and adding 1 to it
  done: mov R1, A ; final result in R1
  ```

- **16 bit subtraction 2762H – 1296H**

  ```assembly
  clr C ; clear Cy
  mov A, #62H ;
  subb A, #96H ; 62H – 96H = CCH and CY=1
  mov R7, A ; store the lower byte of the result in R7
  mov A, #27H ; now subtract the upper bytes
  subb A, #12H ; 27H – 12H – 1 = 14H
  mov R6, A ; store upper byte of result in R6.
  ; Final 16bit result is in R6:R7
  ```
### Multiplication and Division

- **MUL AB** ; A x B, place result in BA
  
  ```
  mov A, #25H ; operand1: 25H
  mov B, #65H ; operand2: 65H
  mul AB ; 25H * 65H = E99H
  ; B = 0EH, A = 99H
  ```

- **DIV AB** ; A/B, place quotient in A and remainder in B
  
  ```
  mov A, #95H
  mov B, #10
  div AB ; A = 9 (quotient), B = 5 (remainder)
  ```
Signed Arithmetic - Concepts

● Representation of the sign
  – Allocate one bit of all numeric quantities for the sign
  – Usually MSB (most significant bit) is assigned for the sign
  – The remaining bits represent the magnitude

● 8051 has only 8-bit registers
  – Signed numbers can have only a 7 bit magnitude
  – Positive numbers in 8051 = 0 to +127 (7 bits)
  – Negative numbers -128 to -1
  – Why?
Signed Arith. – Negative Numbers

- Negative Number representation in signed arithmetic
  - The sign bit (MSB) is 1
  - Magnitude is in 2’s complement form

- Examples

<table>
<thead>
<tr>
<th>Represent –5</th>
<th>Represent –34H</th>
<th>Represent –128</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 = 0000 0101</td>
<td>34H = 0011 0100</td>
<td>128 = 1000 0000</td>
</tr>
<tr>
<td>Cpl = 1111 1010</td>
<td>Cpl = 1100 1011</td>
<td>Cpl = 0111 1111</td>
</tr>
<tr>
<td>+1 = 1111 1011</td>
<td>+1 = 1100 1100</td>
<td>+1 = 1000 0000</td>
</tr>
<tr>
<td>Hex = FBH</td>
<td>Hex = CCH</td>
<td>Hex = 80H</td>
</tr>
<tr>
<td>Hence –5 = FBh</td>
<td>Hence –34H = CCH</td>
<td>Hence –128 = 80H</td>
</tr>
</tbody>
</table>

- Range
  - -128 = 80H
  - -127 = 81H
  - -1 = FFH
  - 0 = 00H
  - 1 = 01H
  - +127 = 7FH
Signed Numbers - Usage

- Application may require a specific quantity be represented as a signed number
  - Temperature measurement –20deg, +10deg etc
  - Water/Gas/Beer level measurement in a tank

- Data is collected and stored as an array of signed numbers
  - Some of the array elements can be negative, while others are positive
  - Identify negative numbers by the MSB. If MSB=1, the number is negative

- Same arithmetic operations (add, sub, mul, div etc) may need to be performed on the array elements, and the result can be positive or negative.
8051 – Signed Arithmetic

- 8051 uses negative number representation in the sub instruction. Not enough!
- When signed numbers are needed, programmer has to take care of signed arithmetic
- Overflow has to be dealt with. Carry flag is not enough, because only 7 bits carry the magnitude in signed numbers
- The 8051 provides another flag – OV (Overflow) for this purpose.
8051 - Signed Arithmetic (contd.)

- **Addition**
  
  \[
  \begin{align*}
  A + B & = 01H, B = FFH \\
  A & = +1, B = -1 \\
  A & = 0000 0001 \\
  B & = 1111 1111 \\
  + & = 1 0000 0000 \\
  A + B & = 0H \\
  \\
  A + B & = FEH, B = FFH \\
  A & = -2, B = -1 \\
  A & = 1111 1110 \\
  B & = 1111 1111 \\
  + & = 1 1111 1101 \\
  A + B & = FDH = -3
  \end{align*}
  \]

- **Subtraction**
  
  \[
  \begin{align*}
  A - B & = 01H, B = FFH \\
  A & = +1, B = -1 \\
  2's(B) & = 0000 0000 +1 = 0000 0001 \\
  A & = 0000 0001 \\
  2's(B) & = 0000 0001 \\
  + & = 0 0000 0010 \\
  A - B & = 02H \\
  \\
  A - B & = FEH, B = 01H \\
  A & = -2, B = +1 \\
  2's(B) & = 1111 1110 +1 = 1111 1111 \\
  A & = 1111 1110 \\
  2's(B) & = 1111 1111 \\
  + & = 1 1111 1101 \\
  A - B & = FDH = -3
  \end{align*}
  \]
Overflow can occur from the magnitudes of the signed numbers, which can change the sign bit.

OV Flag is to be checked for error in signed arithmetic

Example

A+B, A=+96 (60H), B=+70(46H)

A = 0110 0000
B = 0100 0110
+ = 1010 0110 = A6H = -90 (wrong)
OV = 1, CY=0
96+70 = 166 > +127
8051 – OV Flag

- After arithmetic operations, OV is set if
  - Carry from D6 to D7 but no carry from D7
  - Carry from D7 but no carry from D6 to D7
  - These cases indicate a wrong result due to signed arithmetic

- After arithmetic operation involving signed numbers, check OV flag, for error detection
  - Use jb PSW.2 or jnb PSW.2
  - PSW.2 = OV
8051 Logic Instructions

- **AND**
  - `anl dest, source ; dest = dest AND source`
  - Commonly used to mask out (set to 0) a few bits in an operand

- **OR**
  - `orl dest, source ; dest = dest OR source`
  - Commonly used to set a few bits in an operand

- **XOR**
  - `xrl dest, source ; dest = dest XOR source`
  - Commonly used to clear a register, check if two registers have the same value and toggle a few bits

- **Complement**
  - `cpl A ; A = A’`

- None of these instructions affect any flags
8051 – Compare Instruction

- **CJNE**
  - Cjne dest, source, rel address
  - Compare dest and source and jump to relative address if not equal
  - Basically a subtract operation which does not change the operands but affects the CY flag
  - dest > source \(\Rightarrow\) CY=0
  - dest < source \(\Rightarrow\) CY=1

- **Example**
  - Monitor P1 continuously and
  - exit if P1=63H

```
Loop:   mov A, P1
        cjne A, #63, loop
```

```
Cmp:    cjne R5, #80, NEQ
EQ:     …. ;R5= #80
NEQ:    jnc GREAT
LESS:   … ;R5< #80
GREAT:  …. ;R5 > #80
```
8051 – Rotate and Swap

- Bitwise rotation is required in many apps like serial comm., control etc.
- **Rotate right**
  - \( rr \ A \); rotate right \( A \)
- **Rotate left**
  - \( rl \ A \); rotate left \( A \)
- **Rotate right/left with Carry**
  - Use \( CY \) in the rotate sequence (9 bit rotate)
  - \( rlc \ A : D7 \rightarrow CY \rightarrow D0 \)
  - \( rrc \ A : D0 \rightarrow CY \rightarrow D7 \)
- **Swap nibbles**
  - \( swap \ A \); swaps \( D7-D4 \) with \( D3-D0 \)

**Example for RR**

\[
\begin{align*}
\text{mov} & \ A, \#\text{AAH} \\
\text{rr} & \ A \quad \text{; now } A = 55H \\
\text{rr} & \ A \quad \text{; now } A = 2AH
\end{align*}
\]

**Example for RL**

\[
\begin{align*}
\text{mov} & \ A, \#55H \\
\text{rl} & \ A \quad \text{; now } A = \text{AAH} \\
\text{rl} & \ A \quad \text{; now } A = 54H
\end{align*}
\]
8051 – Single Bit Instructions

- Set a bit
  - set bit_name ;bit = 1

- Clear a bit
  - clr bit_name ;bit = 0

- Complement a bit
  - cpl bit_name ;bit = \overline{bit}

- Conditional Jump on bit value
  - jb (jump if bit=1), jnb (jump if bit=0), jbc (jump if bit=1 and clear the bit)
Bit addressable Regs and Memory

- I/O ports (P0 – P3), B, PSW, IP, IE, ACC, SCON and TCON are bit addressable registers (BARs)
- The bits of BARs can be referred to as Register.bitnum (P0.1, PSW.2, IE.4 etc) or by their bit address
- Bit address is the base address of the register + the bit number
  - ACC Base address is E0H, hence ACC.1=E1H, ACC.7=E7H
  - P0, Base address is 80H, hence P0.0=80H, P0.5=84H and so on
- 16 bytes of the internal RAM is bit addressable
  - 20H to 2FH has a bit address of 00H to 7FH
  - clr 67H ; clear bit D7H of RAM location 2CH
  - setb 05H ; set bit 5 of RAM location 20H
Single Bit Operations with CY flag

- 8051 has special instructions that directly manipulate CY flag
  - setb C; clr C; cpl C; mov b,C; mov C,b; jnc, jc, anl C,b; anl C,/b; orl C,b; orl C,/b
  - anl C, /b ; C = CY AND b'

- Example: Turn ON fan (P2.2) and turn OFF light (P2.3)

| Fan_on:        | setb C
|               | orl C,P2.2 ; CY = CY OR P2.2
|               | mov P2.2, C ; turn on fan if not already ON
| Light_off:     | clr C
|               | anl C,P2.3 ; CY = CY AND P2.3
|               | mov P2.3,C ; turn off light if not already OFF
Class-4: Review

- Signed and Unsigned arithmetic
- Binary and BCD coded numbers
- Addition, Subtraction, Multiplication, Division
- Signed Number representation and arithmetic
- Logic Operations
- Rotate and Swap operations
- Bit addressable memory and single bit instructions
Thanks