Instruction Groups

- The 8051 has 255 instructions
  - Every 8-bit opcode from 00 to FF is used except for A5.

- The instructions are grouped into 5 groups
  - Arithmetic
  - Logic
  - Data Transfer
  - Boolean
  - Branching

Arithmetic Instructions

- Add
- Subtract
- Increment
- Decrement
- Multiply
- Divide
- Decimal adjust

Arithmetic Instructions

- ADD
  - 8-bit addition between the accumulator (A) and a second operand.
    - The result is always in the accumulator.
    - The CY flag is set/reset appropriately.

- ADDC
  - 8-bit addition between the accumulator, a second operand and the value of the CY flag.
    - Useful for 16-bit addition in two steps.
    - The CY flag is set/reset appropriately.

ADD Examples

```
mov Acc, #3Fh
add Acc, #D3h
```

- Q1. What is the value of the C, AC, OV flags after the second instruction is executed?

```
  0011 1111
  1101 0011
  ---------
  0001 0010
```

C = 1

AC = 1

OV = 0

ADD Instructions

```
add a, byte          ; a ← a + byte
addc a, byte         ; a ← a + byte + C
```

These instructions affect 3 bits in PSW:

C = 1 if result of add is greater than FF

AC = 1 if there is a carry out of bit 3

OV = 1 if there is a carry out of bit 7, but not from bit 6, or visa versa.

Program Status Word (PSW)

Bit	7	6	5	4	3	2	1	0
Flag	**CY**	**AC**	**F0**	**RS1**	**RS0**	**OV**	**F1**	**P**
Name	Carry Flag	Auxiliary Carry Flag	User Flag 0	Register Bank Select 1	Register Bank Select 0	Overflo w flag	User Flag 1	Parity Bit

add, addc addressing modes

ADD	Acc , #	Acc $\leftarrow$	Acc + Immediate
	, D		
	, R		
	, @R		
ADDC	Acc , #	Acc $\leftarrow$	Acc + Immediate + Carry
	, D		
	, R		
	, @R		

ADD Acc , #ABH
ADD Acc , 2AH
ADD Acc , R3
ADD Acc , @R0

Addition Example

```
; Computes Z = X + Y
; Adds values at locations 78h and 79h and puts them in 7Ah
;------------------------------------------------------------
X          equ      78h
Y          equ      79h
Z          equ      7Ah
;------------------------------------------------------------
           org  8000h
Main:
           mov acc, X      mov acc, 78H ; Acc ← 22H
           add acc, Y      add acc, 79H  ; Acc ← 22H + 33H
           mov Z, acc      mov 7AH, acc ; [7AH] ← 55H
           end
```

Memory picture

Address	contents
77H	11H
78H	22H
79H	33H
7AH	55H
7BH	xxH

Signed and Unsigned Numbers

- unsigned numbers
  - All values are positive

  - Eg. Considering 8 bit numbers
  - 00H to FFH all bit patterns represent positive values
- Signed numbers
  - 2’s complement
    - MSB represents the sign
      - If MSB = 0, then it’s a positive value
      - Else, it’s a negative value

  - Eg. Considering 8 bit numbers
  - 00H to 7FH represents positive values
  - 80H to FFH represents negative values

Signed Addition and Overflow issue

```
 0111 1111   (positive 127)
 0111 0011   (positive 115)
 1111 0010   (overflow
cannot represent 242 in 8
bits 2's complement)
```

```
1000 1111   (negative 113)
1101 0011   (negative  45)
0110 0010   (overflow)
```

```
0011 1111   (positive)
1101 0011   (negative)
0001 0010 (never overflows)
```

Overflow Problem

- Q2: Show how the 8051 would represent –24.
- Overflow occurs if,

  A carry from D6 to D7, CY=0

  CY=1, but NO carry from D6 to D7.
- Q3:

  MOV A, #96

  MOV R1, #70

  ADD A, R1
- Q4:

  MOV A, #-128

  MOV R4, #-2

  ADD A, R4

Overflow Problem

- Q5:

  MOV A, #-2

  MOV R1, #-5

  ADD A, R1

- Q6:

  MOV A, #7

  MOV R4, #18

  ADD A, R4

Program Status Word (PSW)

Bit	7	6	5	4	3	2	1	0
Flag	**CY**	**AC**	**F0**	**RS1**	**RS0**	**OV**	**F1**	**P**
Name	Carry Flag	Auxiliary Carry Flag	User Flag 0	Register Bank Select 1	Register Bank Select 0	Overflo w flag	User Flag 1	Parity Bit

- ***Unsigned*** Number (0 to 255)

  Addition➔ we must monitor the CY (using JNC and JC)

- ***Signed*** Number (-128 to +127)

  Addition ➔ we must monitor OV (using JB PSW.2 and JNB PSW.2)

ADDC: Addition with a Carry – multi-byte numbers

- Can represent values greater than 255 by using more that 8-bits; they are multi-byte numbers.
- Can add multi-byte numbers in multiple steps of addition.
- When we add multi-byte numbers, we need to take into account, the carry values generated in each addition steps.
- propagation of carry will be from the lower bytes to the higher bytes.
- ADDC is used when we add multi-byte numbers.

Example – 16-bit Addition

Add 1E44H to 56CAH

Let's using immediate mode of addressing to refer to the values

```
MOV   Acc, #44H    ; The lower 8-bits of the 1st number
ADD   Acc, #CAH    ; The lower 8-bits of the 2nd number
MOV   R1, Acc      ; The result 0EH will be in R1. CY = 1.
MOV   Acc, #1EH    ; The upper 8-bits of the 1st number
ADDC  Acc, #56H    ; The upper 8-bits of the 2nd number
MOV   R2, Acc      ; The result of the addition is 75H
```

The overall result: 750EH will be in R2:R1. CY = 0.

The 16-bit ADD example

```
; Computes Z = X + Y     (X,Y,Z are 16 bit values)
;-------------------------------------------------------------
X        equ      78h
Y        equ      7Ah
Z        equ      7Ch
;-------------------------------------------------------------
         org 9000h
Main:    mov acc, X
         add acc, Y
         mov Z, acc
         mov acc, X+1
         addc acc, Y+1
         mov Z+1, acc
         end
```

Same program, using Indirect addressing for X and Y variables with R0 and R1 registers

```
Mov R0, #78H
Mov R1, #7AH
Mov Acc, @R0
Add Acc, @R1
Mov Z , Acc
Inc R0
Inc R1
Mov Acc, @R0
Adc Acc , @R1
Mov Z+1 , Acc
```

Example: Increment 16-bit Word

- **Assume 16-bit word is in R3:R2**

```
mov Acc, R2
add Acc, #1     ; use add rather than increment to affect CY
mov R2, Acc
mov Acc, R3
addc Acc, #0    ; add CY to most significant byte
mov R3, Acc
```

Subtract

SUBB A, byte	subtract with borrow (carry flag value)

Example:

```
SUBB A, #0x4F        ;A ← A – 4F – CY
```

Notice that
There is no subtraction WITHOUT borrow instruction !!!
Therefore, if a subtraction without borrow is desired,
it is necessary to clear the CY flag.

Example:

```
Clr  c
SUBB A, #0x4F        ;A ← A – 4F
```

Addressing modes supported by SUBB

SUBB	A , #	Acc $\longleftarrow$	Acc-Immediate-Carry
	, D		
	, R		
	, @R		

SUBB Acc , #BDH
SUBB Acc , 16H
SUBB Acc , R2
SUBB Acc , @R1

SUBB: Subtract with Borrow

- Q6: Show values of registers after each of the instructions in the following.

CLR C	CY = 0
MOV A, #3FH	Acc = 3FH
MOV R3, #23H	R3 = 23H
SUBB A, R3	Acc = 3FH - 23H = 1CH

SUBB: Subtract with Borrow

- SUBB with CY=1 for multi-byte numbers.
- Q8: Analyze the following programs.

```
CLR   C
MOV   A, #62H
SUBB  A, #96H
MOV   R6, A
MOV   A, #27H
SUBB  A, #12H
MOV   R7, A
```

27 62H

- 12 96H

= 14 CCH

in R7 R6

Instructions that Affect PSW bits

Instructions that Affect Flag Settings[1]

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

Arithmetic instructions

- Increment and decrement operations

INC A	increment A
INC byte	increment byte in memory
INC DPTR	increment data pointer
INC @REG	increment byte pointed by REG
DEC A	decrement accumulator
DEC byte	decrement byte in memory
DEC @REG	decrement byte pointed by REG

- The increment and decrement instructions do NOT affect the C flag.
- Notice we can only INCREMENT the data pointer, not decrement.

Arithmetic Instructions

- INC
  - Increment the operand by one.
    - The operand can be a register, a direct address, an indirect address, the data pointer.
- DEC
  - Decrement the operand by one.
    - The operand can be a register, a direct address, an indirect address.
- Examples:
  - INC R3
  - INC 55H
  - DEC @R0
  - DEC R0

Arithmetic Instructions

- DA
  - Decimal adjust the accumulator.
    - Format the accumulator into a proper 2 digit packed BCD number.
    - Operates only on the accumulator.
    - Works only after the ADD instruction.

BCD: Binary Coded Decimal

- Unpacked BCD (one digit in 1 byte data)

  9 ➔ 0000 1001B (1 byte)

  5 ➔ 0000 0101B (1 byte)

- Packed BCD (two digits 1 byte data): it is twice as efficient in storing data compared to unpacked BCD.

  59H ➔ 0101 1001B (1 byte)

- Problem with adding BCD numbers

  MOV A, #17H   0001 0111

  ADD A, #28H   0010 1000

The sum is 0011 1111 =3FH

➔ This is NOT BCD number!!

- A BCD number only have digits from 0000 to 1001 (0 to 9).

Decimal Adjust

```
DA a          ; decimal adjust a
```

Used to facilitate BCD addition.
Adds “6” to either high or low nibble after an addition to create a valid BCD number.

Example:

```
mov a, #23h
mov b, #29h
add a, b      ; a ← 23h + 29h = 4Ch (wanted 52)
DA a          ; a ← a + 6 = 52
```

Examples – BCD addition

Add 51 to 46 BCD

```
MOV    A, #51H      ; Place 1st number in A
ADD    A, #46H      ; Add the 2nd number.
                    ; A = 97H
DA     A            ; A = 97H
```

Add 85 to 67 BCD

```
MOV    A, #85H      ; Place 1st number in A
ADD    A, #67H      ; Add the 2nd number.
                    ; A = ECH, answer as per BCD is 152
DA     A            ; A = 52H and CY = 1
```

Multiply

- 8051 can multiply two 8 bit unsigned numbers
- When multiplying two 8-bit numbers, the size of the maximum product is 16-bits

FF x FF = FE01

(255 x 255 = 65025)

```
MUL AB          ; BA ← A * B
```

Note : **B gets the High byte**

A gets the Low byte

Division

- 8051 can divide 8-bit unsigned number by 8-bit unsigned number

```
DIV AB        ; divide A by B

A ← Quotient(A/B)
B ← Remainder(A/B)
```

OV - used to indicate a divide by zero condition.
C – set to zero

Examples of mul and div

- If A = 78H and B = 2H
  - MUL AB ; BA = 00F0H
  - DIV AB ; quotient, A = 3CH and reminder, B = 0H
- If A = 22H and B = 5H
  - MUL AB ; BA = 00AAH
  - DIV AB ; quotient, A = 6H and reminder, B = 4H
- If A = CDH and B = 34H
  - MUL AB ; BA = 29A4H
  - DIV AB ; quotient, A = 3H and reminder, B = 31H
- If A = 45H and B = ABH
  - MUL AB ; BA = 2E17H
  - DIV AB ; quotient, A = 0 and reminder, B = 45H

Arithmetic Instructions- summary (not complete listing)

Instuction	Description
ADD A, byte	add A to byte, put result in A
ADDC A, byte	add with carry
SUBB A, byte	subtract with borrow
INC A	increment A
INC byte	increment byte in memory
INC DPTR	increment data pointer
DEC A	decrement accumulator
DEC byte	decrement byte
MUL AB	multiply accumulator by b register
DIV AB	divide accumulator by b register
DA A	decimal adjust the accumulator