

Data Moving: onchip memory

```
; 8-bit between onchip memory
MOV A,30H ; direct mode
MOV 40H,A
MOV R0,#30H ; indirect mode
MOV A,@R0
MOV R1,#40H
MOV @R1,A
; [40h] = ?
```

Data Moving: bit addressable

; 1-bit addressable registers

MOV ACC,#10101000B

CLR ACC.7 ; ACC == Accumulator

SETB ACC.7

MOV C,ACC.7

MOV ACC.0,C

MOV C,32.0;copy bit 00H to c

MOV ACC.1,C; copy to ACC.1

Data Moving: external memory

; 8-bit external memory

MOV DPTR,#9000H

MOV A,#0AAH

MOVX @DPTR,A

INC DPTR

MOVX A,@DPTR

; A = ?

; [9000]=? [9001] = ?

Data Moving: code memory

; table processing

MOV DPTR,#table

CLR A

loop: PUSH A ; save A to stack

MOVC A,@A+DPTR ; read byte

POP A ; restore A from stack

INC A

JMP loop

Data Moving: code memory

Table: DB '1'

DB '2'

DB 33H ; ASCII code for '3'

DB 34H

DB '5'

; run program with single-step

; If A = 0,1,2,3,4,5 after open

; table with MOVC A,@A+DPTR, A=?

Data Moving: via STACK

MOV A,#79H

MOV B,#97H

PUSH B

PUSH A

ADD A,B ; A = ?

POP A ; A = ?

POP B ; B = ?

Data Moving: nibble swapping

MOV A,#79H

SWAP A ; A = ?

MOV R0,#32H

XCHD A,@R0 ; A = ?

SWAP A ; A = ?

MOV R1,A ; R1 = ?

Data Moving: byte swapping

MOV A,#254

XCH A,45H ; A = ?, [45H]=?

MOV R0,#33H

XCH A,R0 ; A = ?

MOV R1,#46H

XCH A,@R1; R1 = ?, A= ?

; direct and indirect modes!

Copy onchip RAM to external RAM

```
; subroutine copy1  
; entry: R0, DPTR  
COPY1:      MOV R7,128  
LOOP:       MOV A,@R0  
              MOVX @DPTR,A  
              INC R0  
              INC DPTR  
              DJNZ R7, LOOP  
              RET
```

Calling subroutine copy1

```
Main:      MOV R0,#0
           MOV DPTR,#0E000H
           CALL COPY1
           JMP MONITOR

; check the internal memory before
; run above code with jump command
; dump the external memory after
; run the main program
```

Boolean Processing:

```
Main:      MOV A,#10101011B
           SETB C
           ANL C,ACC.0
           MOV ACC.0,C
           ANL C,/ACC.1
           ORL C,/ACC.7
           MOV 32.0,C
           CPL 32.1
           CLR 32.2
```

Boolean processing:

```
Main:      MOV  A,#01010000B
           ADD  A,#0FEH
           JC   skip1
           CLR  ACC.7
skip1:     CPL  ACC.7
           MOV  C,ACC.7
           JNC  skip2
           JB   ACC.7,skip3
```

Boolean processing:

skip2: JBC ACC.7,skip4

JB ACC.7,skip5

skip3: CLR ACC.7

skip4: SETB ACC.7

skip5: ORL C, /ACC.7

CPL C

CPL ACC.0

MOV ACC.1,C

Relative Offset:

```
delay:      MOV R7,#10
            DJNZ R7,$
            RET

main:       CLR A

loop:      MOV P1,A
            ACALL delay
            LCALL delay
            SJMP  loop
```

Relative Offset:

8100 7F0A delay: MOV R7,#10

8102 DFFE DJNZ R7,\$

8104 22 RET

;offset byte can be computed by

;destination - current PC

;8102-8104 = -2

;negative number is two complement!

; 0000 0010 > 1111 1101 > 1111 1110

Relative Offset:

- ; 8-bit signed number +127 to -128
- ; or 7FH to 80H
- ; current PC + offset
- ; 8104H + 0FEH = ?
- ; offset byte will be positive
- ; number 0 to +127 or 0-7FH for forward jump and will be negative
- ; -1 to -128 for backward jump!

Relative Offset:

```
8106 F590 loop: MOV P1,A
8108 3100 ACALL delay
810A 128100 LCALL delay
810D 80XX SJMP loop

; let's compute XX
; destination = ?
; current Program Counter = ?
; offset byte XX = ?
```