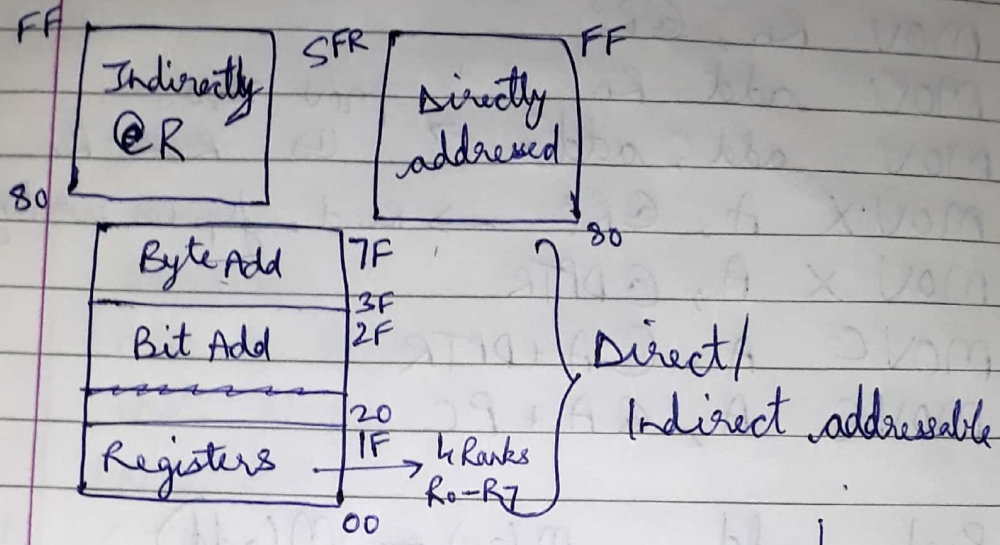
30/07 Instn's

Internal RAM 8052



Arithmetic

R0 or Ri only

Add A, Rn/@Ri/add/#n
AddC A, Rn/@Ri/add/#n
Subb A, Rn/@Ri/add/#n
Mul AB
Div AB
INC Rn/@Ri/add
DEC Rn/@Ri/add

Logical

ANL A, Rn/@Ri/add/#n
ORL & XRL
RL, RLC, RR, RRC
CLR A, CPL A

Bit

ANL C, bit
ORL C, bit
CLR C
CLR bit ⇒ 8bit addr
SETB C ⇒ carry = 1
SETB bit
CPL C
CPL bit

ANL add, A/#n

dest^n is add.

Page No.:

Data transfer

MOV Rn, add
MOV Rn, @Ri
MOV add, Rn
MOV add, add → MOV 2,0 ↳ R2 ← R0

ext. RAM
- MOVX A, @Ri ⇒ Read, flip for writing
- MOVX A, @DPTR

ext. ROM
- MOVC A, @A+DPTR
- MOVC A, @A+PC

Push add M(SP) ← M(add)
Pop add M(add) ← M(SP)
XCH A, Rn / @Ri / add
SWAP A ⇒ 86 → 68 (Interchanges the 2 nibbles)

control flow

ACALL add11
LCALL add16
AJMP add11 - 2byte
LJMP add16 - 3byte
SJMP rel - 2byte
RET - 1byte
RETI
JZ rel
JNZ rel
JC rel
JNC rel
CJNE A, add, rel → compare A & add content & if not equal, jump to

CJNE Rn/@Ri/add, #n, rel
DJNZ Rn, rel
↳ 1st decrement Rn, check if zero, if not zero then jump

Page No.:

10 values, 1st stored at 50H, ~~search for 50~~

→
```
    MOV R3, #10
    MOV R0, #50H
    MOV R5, #0
    MOV A, @R0
Loop: INC R0
    DEC R3
    ADD A, @R0
    JNC Next
    INC R5
Next: CJNE R3, #0, loop
    MOV R4, A
Stop: SJMP Stop
```

31/01 40 values, 1st stored in 50H, check if 25 present.

```
    MOV R0, #50H
    MOV R2, #40
loop: MOV A, @R0
    CJNE A, #25, Forward
    MOV R3, 0 ⇒ 0 refers to R0. We can't write R0 directly
    SJMP STOP
forward: INC R0
    DJNZ R2, loop
    CLR A
STOP: SJMP STOP
```

Assembler directive

EQU Pi, #3.14

Sort 40 nos., starting from 50H

```
      MOV R2, #40
loop2: MOV R0, #50H
      MOV R3, 2
      DEC R3
loop1: MOV A, @R0
      INC R0
      MOV R5, A
      CLR C
      SUBB A, @R0
      JC Nent
      MOV A, R5
      XCH A, @R0
      DEC R0
      MOV @R0, A
      INC R0
Nent: DJNZ R3, loop1
      DJNZ R2, loop2
STOP: SJMP STOP
```

SWAP

Subroutine

```
ACALL SWAPPER

08-09
SP→09
```

```
SWAPPER: MOV A, R5
      XCH A, @R0
      DEC R0
      MOV @R0, A
      INC R0
      RET
```

Page No.:

```
LCALL WAIT        WAIT: PUSH 04
                        MOV R4, #50
                  WAIT2: DJNZ R4, WAIT2
                        RET
                        POP 04
```

2/07 **Subroutines**

To preserve value in R1 ⇒ MOV R3, I n1

```
MOV R1, #50
MOV R2, #60
LCALL Delay
```

```
Delay: DJNZ RB, Delay
       RET          n2
```

⇒ $n_1 \cdot 50 + n_2$

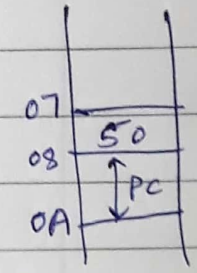
Passing parameters to subroutines by reference (memory)

Led blink

```
MOV R1, #50
Loop: MOV A, #00H
      MOV P1, A
      LCALL Delay
      MOV A, #FFH
      MOV P1, A
      LCALL Delay
      SJMP Loop
```

Passing by ~~value~~ stack

```
MOV  R1, #50
MOV  R2, #60
PUSH 01
LCALL Delay
```




```
Delay: DEC SP
       DEC SP
       POP 3
loop:  DJNZ R3, loop
       INC SP
       INC SP
       INC SP
       RET
```

By changing RS1 & RS0, we can change the register bank which will be used.

5 4 — PSW SETB PSW 5

Page No.:

If we don't know how high stack will go & if we plan to change the register bank, then its good to change SP location.

MOV SP, #20.