MICROCONTROLLER

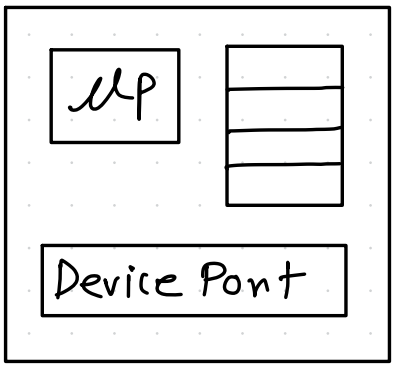
Microcontroller

- Small
- controls all components

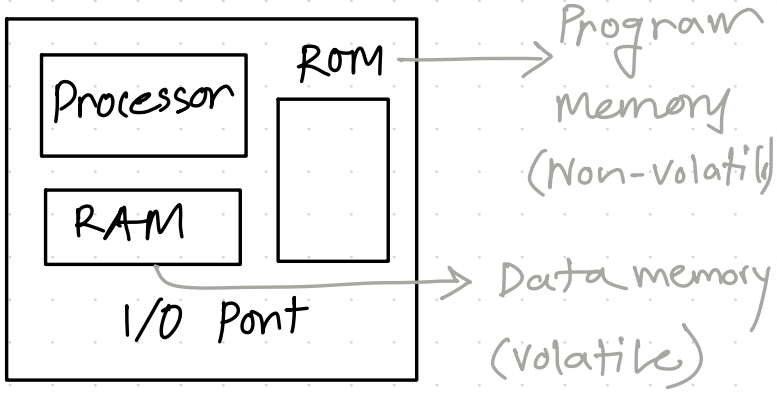
μp computer on chip (CoC)

μc system on chip (SoC)

μp



μc



Microprocessor vs Microcontroller

μp	μc
• CPU is stand-alone, RAM, ROM, I/O are separate.	• CPU, RAM, ROM, I/O all are on a single chip.
• Generally large is size.	• Small and compact in size.
• More expensive	• Cheaper than μp
• Higher power consumption.	• Lower power consumption.
• General purpose	• Built for specific/fixed purpose.

CLASSIFICATION OF μC

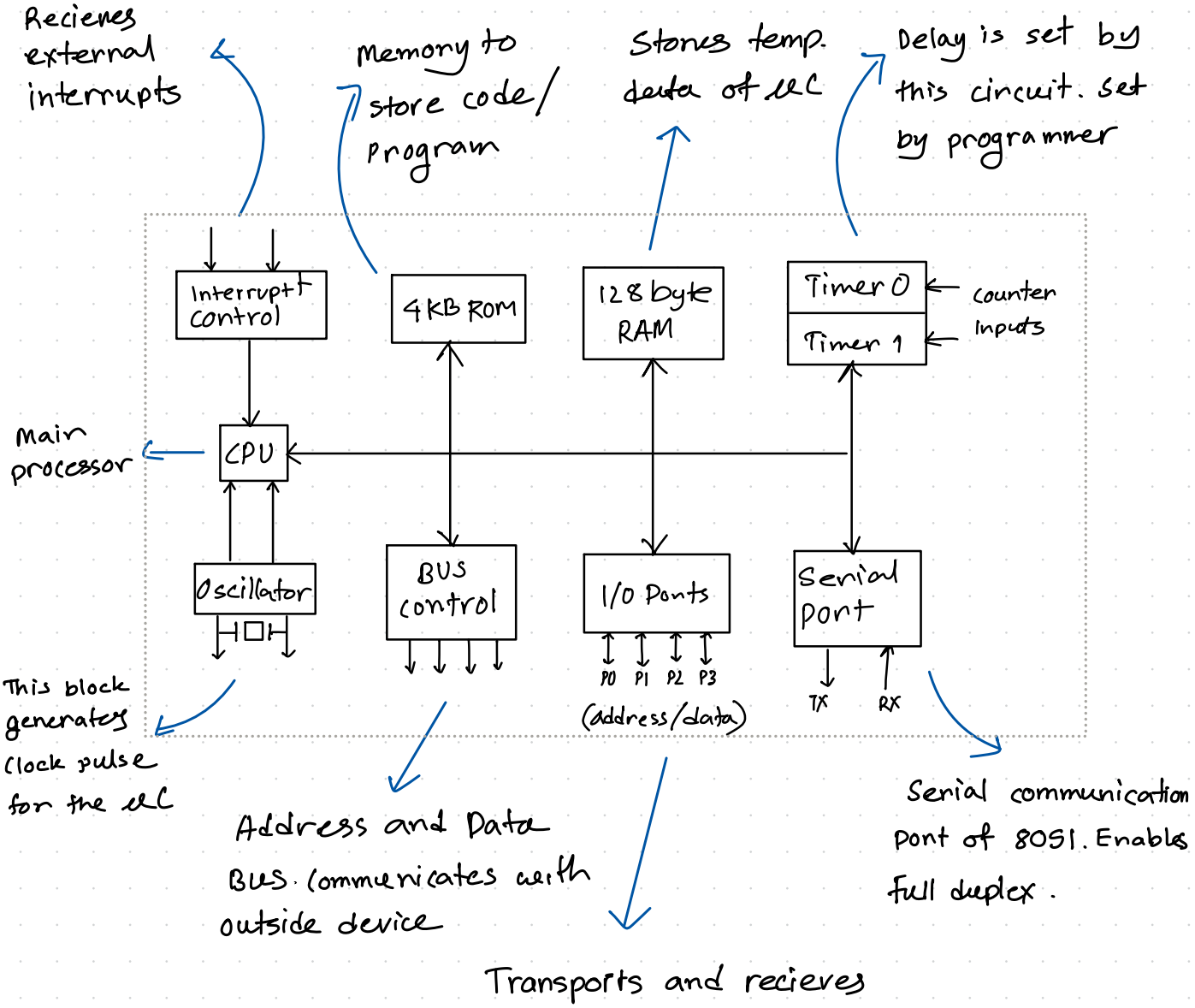
(i) Data Bus → 8 bit (8031)
→ 16 bit (8051)
→ 32 bit (8081)

8051

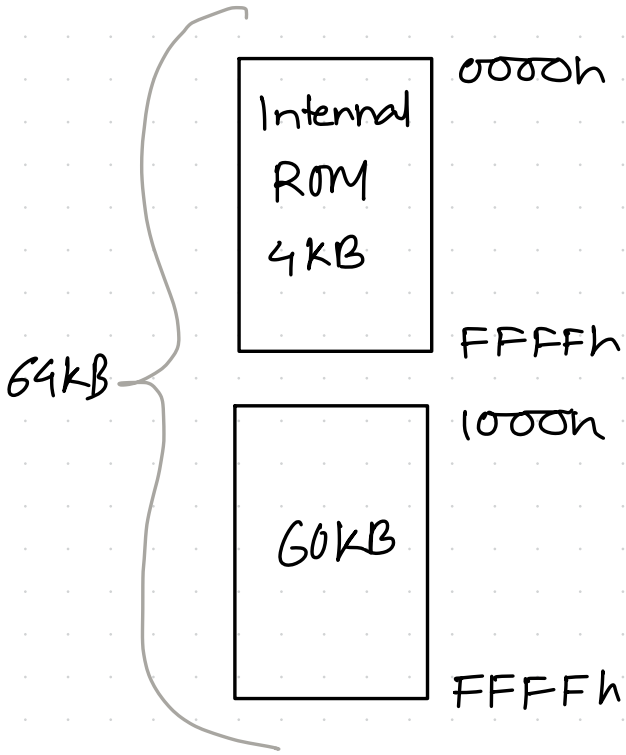
ROM → 4 KB (4096)
RAM → 128 bytes

(ii) Internal memory → External (8031)
→ Embedded (8051)

BLOCK DIAGRAM



ROM – READ ONLY MEMORY



① Internal (IROM)

② External (EROM)

③

If programmer wants a completely separate ROM to store a large program.

RAM- RANDOM ACCESS MEMORY

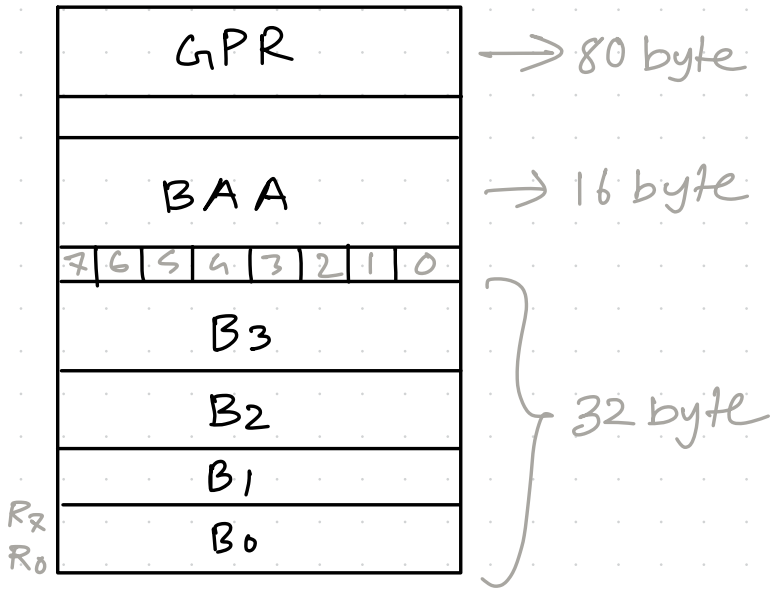
RAM — 128 byte
↳ Volatile

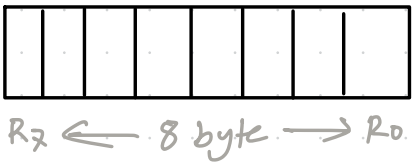
We can use external RAM if needed.

GPR → General Purpose Register

BAA → Bit Addressable Area

B_0-B_3 → Banks





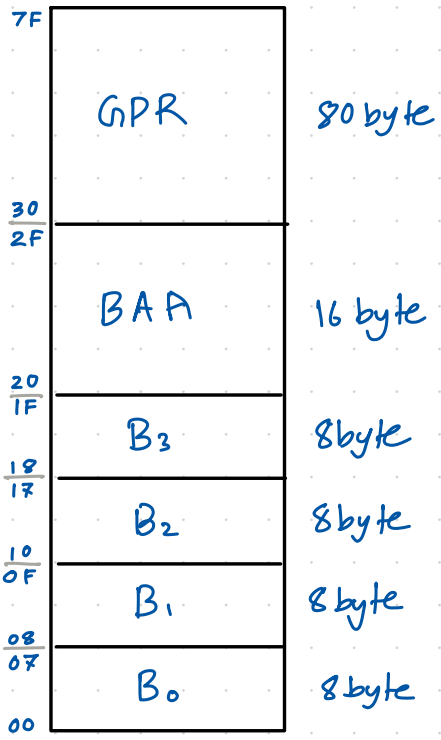
* We can do bitwise operation in BAA

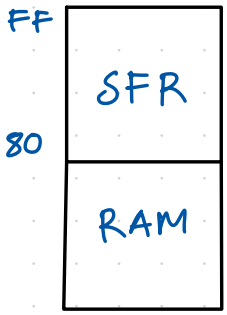
ROM vs RAM

ROM	RAM
• Stores program	• Stores Data
• Type; Non volatile	• Type: Volatile
• 4kB ROM. Can be extended upto 64KB.	• 128 byte RAM. can be extended upto 64kB
• $\overline{EA}$ pin works under ROM	• Not functional for RAM.

ROM → 4KB + 60KB (Extended) / 64K

RAM → 128 bytes (00-7F)





→ Special function Register (128 byte)

* sits on top of RAM

*SFR contains register like µP

(I) Accumulator (A & B)
(II) Status Register
(III) Pointer register
(IV) Interrupt Reg
(V) Stack pointer
(VI) I/O Register

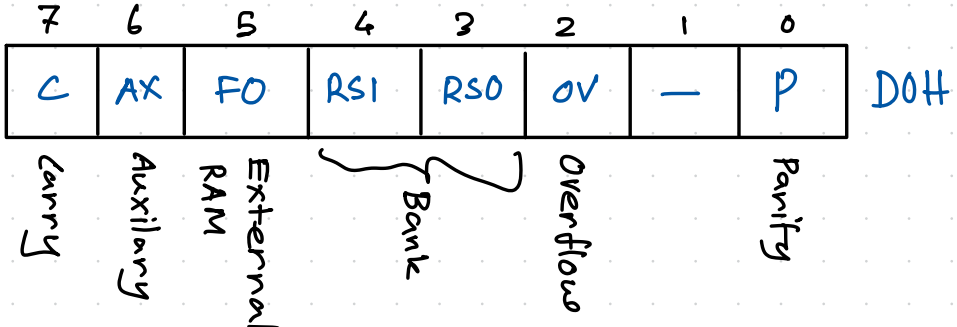
(I) Accumulator (A & B) 8 + 8 = 16 bit

→ Performs math and logical operation

(A)	(B)	
8 bit	8 bit	→ size
E0H	F0H	→ Memory location

→ Mainly works A, involves B for division.

(II) Status Register (Flag)



RS1	RS0	
0	0	→ B0
0	1	→ B1
1	0	→ B2
1	1	→ B3

(III) Pointer Register

→ To indicate CPU, which Bank we are using

DPH → Higher Bank (B_2, B_3) 83H

DPL → Lower Bank (B_0, B_1) 82H

(IV) Stack Pointer

(V) I/O port register

P_0 → 80H
P_1 → 90H
P_2 → A0H
P_3 → B0H

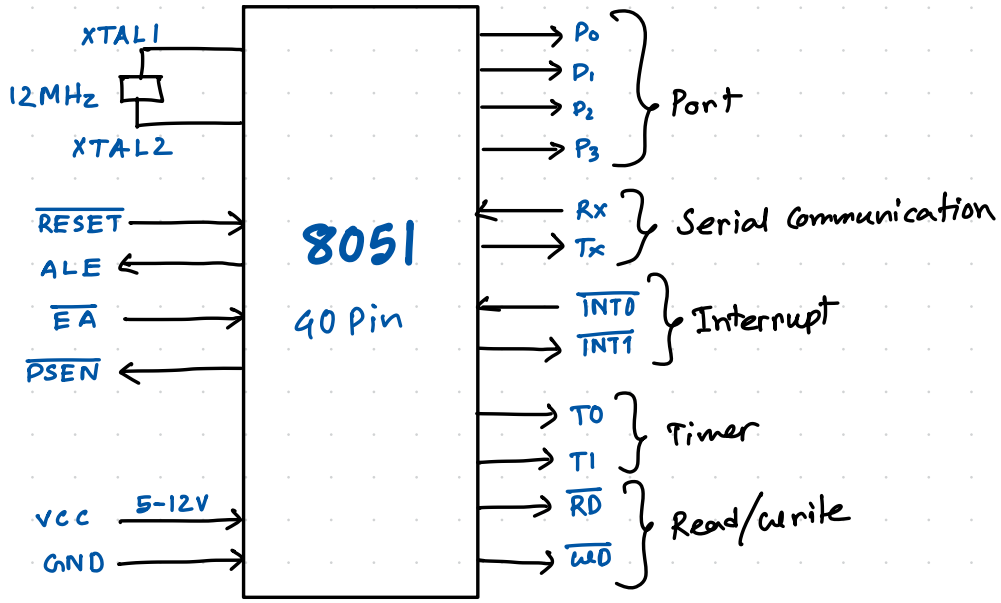
* 8 bit each

(VI) Interrupt Register

→ Primary Interrupt Register (PIR) A2H

→ Secondary " " (SIR) B2H

8051 PIN DIAGRAM



XTAL1-XTAL2 → Clock

Reset → Program Reset

ALE → Address Latch Enable
- 1 - Address
- 0 - Data

$\overline{EA}$ → Enable
- 1 - Internal (4KB) ROM
- 0 - Internal + External (4+60) kB ROM

PSEN → Program store enable
- 1 - External RAM
- 0 - No use

RX → Data receive peripheral
TX → Data send
} Serial Communication

T_0, T_1 → Timer Circuit Control

INT0 → Primary Interrupt

INT1 → Secondary "

BIT MANIPULATION

Boolean Instructions

1. set — SETB 42H → 1
2. clear — CLR 42H → 0
3. compliment — CPL 42H → 0/1

*Applicable for BAA (20H-2F)

Pont 0,1,2,3

8bit each

SETB P0.2 → 1

↙

pont location

Logical Operation

1. AND	ANL	A, source
2. OR	ORL	A, source
3. XOR	XRL	A, source
4. NOT	CPL	destination

SWAP A

A= 78H

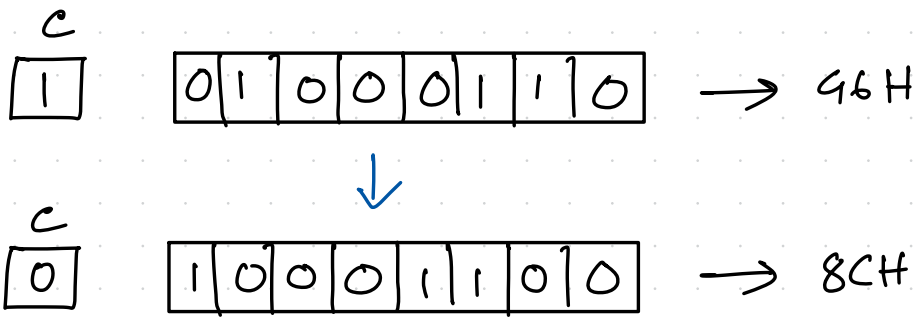
↓

A = 87H

Rotate

RL destination/A

A= 46H, carry = 1



RRC/RLC
Rotates through carry flag

RR destination/A

ADD A, #17H → Direct
ADD A, 17H → Memory
ADD A, R2 → Bank
ADD A, @R1 → ROM

ADC → Add with carry
SUB → Subtraction
INC → Increment
DEC → Decrement

ADDRESSING MODE

① Immediate Mode

MOV → Internal RAM only

```
MOV  DPH, #82H
MOV  A,   #10H
```

we put # to indicate value, not address.

⑪ Register mode

R0, R1, R2, R3 . . . A, B, DPH, DPL → Registers

MOV R1, A [A will stay in memory, copy, not cut]

MOV R1, R2 ✗ → Reg to Reg invalid

Instead we can do,

```
mov 07H, R2   → Reg to memory
mov R1, 07H   → Memory to Reg
```

⑪ Direct Addressing

```
MOV  A, 25H   → Memory location of value.
MOV  25H, A     to indicate value, we use #
MOV  15H, 25H
```

⑫ Indirect Addressing mode

To communicate with internal ROM(4KB) with RAM(128B)

R0, R1, R2, R3

```
MOV  A, @R2   → Internal ROM
MOVX A, R1    → External RAM to RAM
```

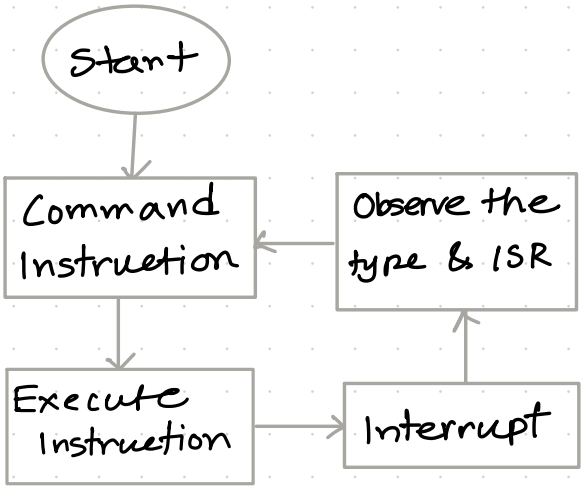
External

Direct comm. from internal ROM to, External RAM is not possible. Instead → Int ROM → Int RAM → Ex RAM

INTERRUPTS

↳ 6 Types

- RST ← ↳ RESET (1) ↗ 1st Priority
- T0/T1 ← ↳ Internal Interrupt (2)
- INT0/INT1 ← ↳ External " (2)
- TI/RI ← ↳ Serial " (1)



Vector Address

Interrupt	Address
RESET	0000H
INT0	0003H
T0	000BH
INT1	0013H
T1	001BH
Serial	0023H

IE Register

7	6	5	4	3	2	1	0
EA	—	—	ES	ET1	EX1	ET0	EX0

L H

EA → Enable All (1-on, 0-off)

6/5 → Preserve bit (functional in 8096)

ES → Enable serial

ET →

IP Register

7	6	5	4	3	2	1	0
			PS	PT1	PX1	PT0	PX0

L H

Interrupt vs Polling **

* Briefly discuss how 8051 handles interrupts? (BQ)

→

MEMORY INTERFACING

Chip Capacity → number of bits that a semicond. can store
→ how many bits it can process (16bit in 8051)

Storage Capacity → how many bits of data it can store.

* Number of memory location depends on address pin.
* Number of bits each location can hold depends on data pin

2^x → 12 Address pin
→ 4 Data pin

2^x → No of Address pin

→ 2^{12} = 4096 byte [4KB Memory location]

→ 4096 X 4 = 16K [memory size]

No of Datapin

4KB memory can hold 4bit in each location. Thus capacity 16K