Microprocessor + Memory + I/O Interface + Timer system .... = Microcontroller

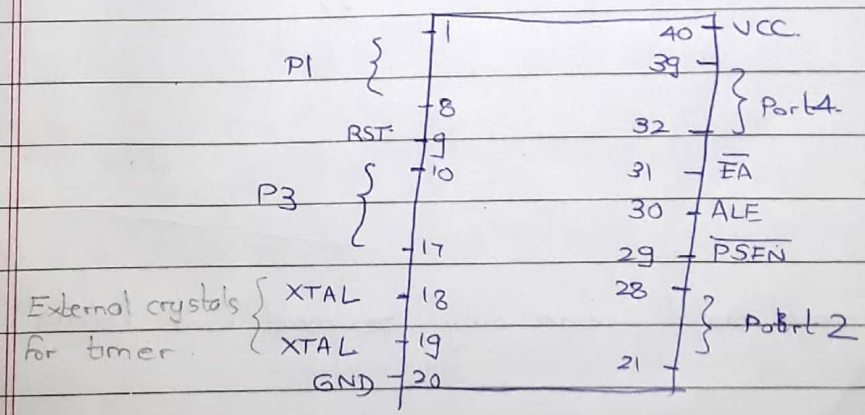
+ Interrupts

↳ like Alt+F4

Changing execution midway

→ 8051 :- (8-bit processor)

- Different memory for program, different memory for data.
- No secondary memory.
  - Memory must be non-volatile for program. Memory is erased only when overwritten (not erased when powered off)
  - Memory for data can be volatile.
- 4kB non-volatile instruction (program) memory (expandable to 64kB)
  128 B volatile data memory (expandable to 64kB)
- 2 timers
- 5 interrupts.
- Unlike 8085, supports operations on individual bits along with bytes.
  - For bit operations, uses carry bit as one operand.
- 64kB :- Maximum accessible (external) memory
- Address and data pins are multiplexed (like 8085)
  - ALE pin is present.
- 4 ports (8 bit) :- P0 - P3
- 40 pins



Scanned by CamScanner

- Port 3 is also used as following (multiplexed)

Pin	10	11	12	17.
	RxD	TxD	$\overline{INT0}$	$\overline{INT1}$	T0	T1	$\overline{WR}$	$\overline{RD}$

- Port 4 is multiplexed to be used as address/data paths (AD0 to AD
- Port 2 " " " " " " address paths (A8 to A15)

→ Memory.

- 4kB internal Read only memory (0000 to 0FFF in hex - 12 bits)
- External ~~60~~ kB ROM (1000 to FFFF)

☆

- $\overline{PSEN}$:- ~~When 0,~~ read ~~from external memory~~ For reading from external code memory every time (output signal that goes to memory)

Program Status Enable

- $\overline{EA}$:- External enable input.

When $\overline{EA} = 0$, ~~#~~ only external memory is used (64 kB)

When $\overline{EA} = 1$, 4 kB internal, 60 kB external.

- $\overline{WR}$, $\overline{RD}$

Write or read data memory. (upto 64 kB) (RAM)

27/7

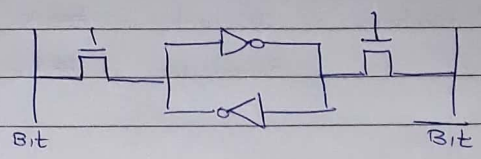
- Code memory, once burnt, can only be read from (ROM) during execution.

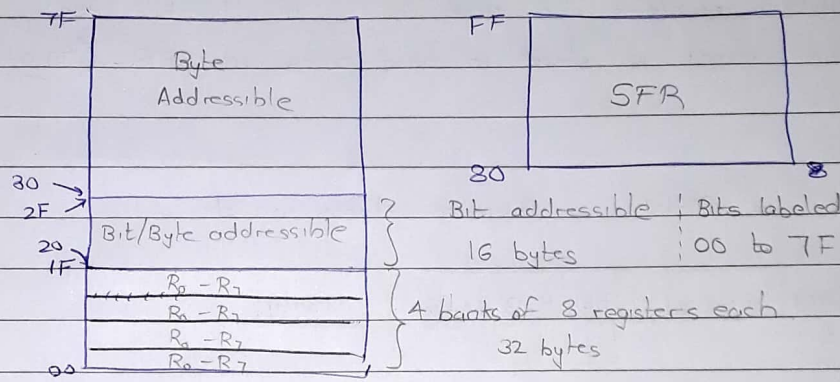
Scanned by CamScanner

→ RAM

- 128 B internal.
- Unlike 8085, RAM is like memory instead of registers in construction.

* SRAM





- 20 to 2F :- Can be used to access 16 8-bit registers (labelled 20 to 2F) or 128 bits (labelled 00 to 7F.)

SFR ↓

- RAM = ~~20~~ 00 to 7F (128 bytes) --- "General Function Register"
  Special Function Registers = 80 to FF (128 bytes)

- 4 bytes in SFR keep track of ports
  P0 = 80, P1 = 90, P2 = A0, P3 = B0

- 81 = Stack Pointer

- 82, 83 = DPTR (Data Pointer) (16-bit)
  DPH ↑ DPL ↑

Verify - ~~D~~E0 = Accumulator

P.T.O.

Scanned by CamScanner

Verify - Register 'B' = F0

On multiplying 2 8-bit numbers :- 8 LSBs in Acc,
8 MSBs in B

On dividing 2 numbers :- Quotient in B
Remainder in Acc

- The two counters can each count till 16-bit nos.
Their count is store in two pairs in SFR (TL0-TH0, TL1-TH1)
16 bit 16 bit

- D0 = PSW :- Program status word

Carry	Aux Carry	-	RS1	RS0	Overflow	-	Parity

Carry: Comes from MSB+1th bit because MSB is sign bit

RS1, RS0: Refers to which of the 4 register banks is being used (default 00)

Overflow: Comes from MSB

- Indirect memory access uses RS1, RS0 and R0-R7 to refer to registe
Direct memory access can directly use 00 to 1F

- 4 addressing modes (for the 4 register banks)
- Register, Direct, Indirect (Register), Immediate

Register	Direct	Indirect (Register)	Immediate
R0-R7 depending on RS1-RS0	00 to 1F for banks, or 00 to FF in general for RAM USFR	@Ri i=0 or 1 only Contents of register are the address to be found in RAM USFR	#n.
		@R0 means R0 of currently selected bank	

Scanned by CamScanner

⟹ Instructions

→ Arithmetic

1 Addition.

ADD A, R_n eg- ADD A, R0 :- Acc = Acc + R0

ADD A, add. ADD A, 4 :- Acc = Acc + R4

ADD A, @R_i ADD A, @R2 :- Acc = Acc + M(R2)

ADD A, #n ADD A, #4 :- Acc = Acc + 4

ADDC

2 Subtraction

There is no subtraction without borrow

SUBB

3 Multiplication.

MUL AB B = 8MSBs of A*B , A = 8LSBs of A*B

cannot use other registers

4 Division

DIV AB B = Quotient(A/B) A = Rem(A/B)

→ Logical

1 AND

ANL A, R_n

ANL A, Add

ANL A, @R_i

ANL A, #Imm

ANL Add, A

ANL Add, #Imm

{ AND of Address with A or #Imm and result is stored in Address

• Only logical operations can have results stored not in accumulator.

Scanned by CamScanner

2. OR

ORL ∴ All 6 variants as with AND

3. XOR

XRL ∴ All 6 variants as with AND

→ Other

1. Rotation

RL

RLC

RR

RRC

2. CLA Clear Accumulator (all zeroes)

3. CPL Complement Accumulator

→ Data Transfer

1. MOV

MOV A, address A = address

MOV address, A

MOV address, R_n

MOV address, address

✗ MOV R_m, R_n --- Use MOV add, add for achieving same effect

MOV address, @R_i

MOV address, #n Store n in address

MOV DPTR, #16-bit data

MOV transfers with internal RAM or SFR.

Scanned by CamScanner

A + DPTR means
add value in A to 16-bit no of DPTR
and access its ROM

- MOVX transfers from external memory to RAM or SFR, or vice versa.
  MOVX address, @DPTR
  MOVX @DPTR, address
  (both 16-bit)

- MOVC transfers from ROM to Acc only (✗ vice versa) — generates PSEN signal and not RD signal
  MOVC A, @(A + DPTR)
  MOVC A, @(A + PC)
  (PC: Program Counter)

2 Exchange.
- XCH A, R_n
  XCH A, Address
  XCH A, @R_p

→ Changing Control Flow

1 Calling a function
ACALL 11-bit-address ---- 5 MSBs are 00000, for compact instructions
LCALL 16-bit address
RET

- When first function of a nest of functions is called, stack pointer is initialized to 07 (R7 of bank 1), and the PC is stored in 08;09. Further nested functions will be stored in 0A;0B, 0C-0D and so on. Returning will return in reverse order (LIFO), and decrement stack pointer by 2.
  - While executing any function, do not overwrite registers of stack, or you won't be able to return.
  - The starting register of stack pointer can be manually changed from 07.

Scanned by CamScanner