EE309(S2): Microprocessors

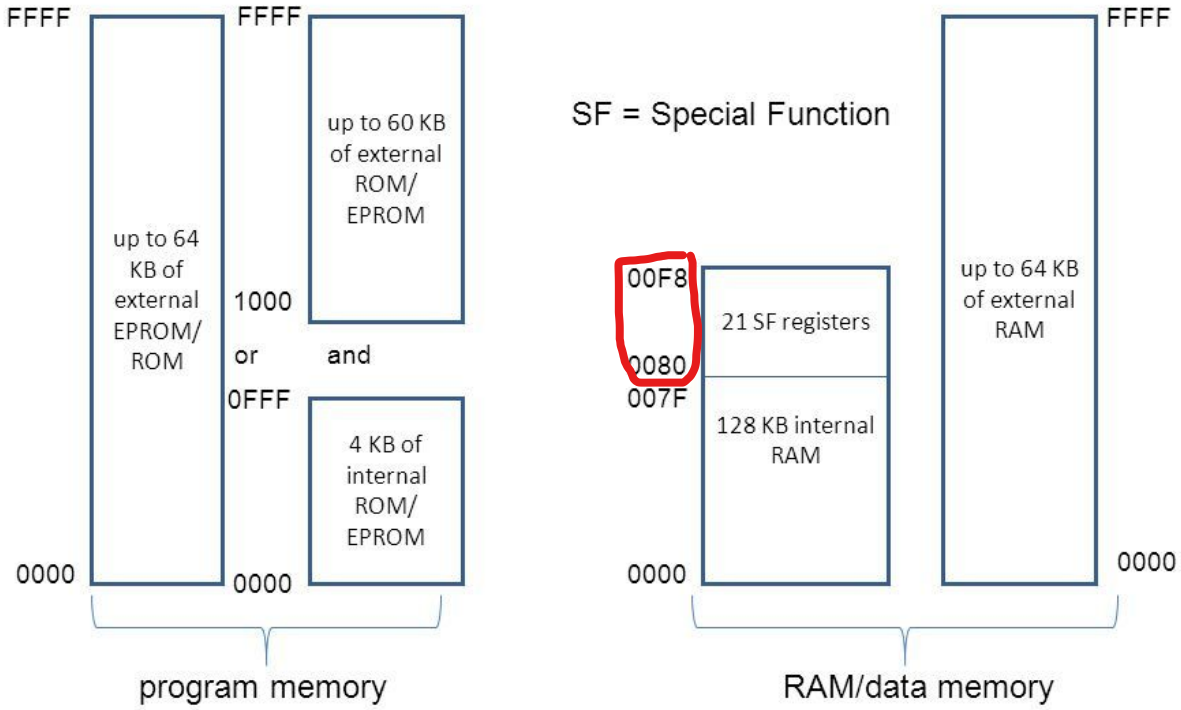
Spring 2025

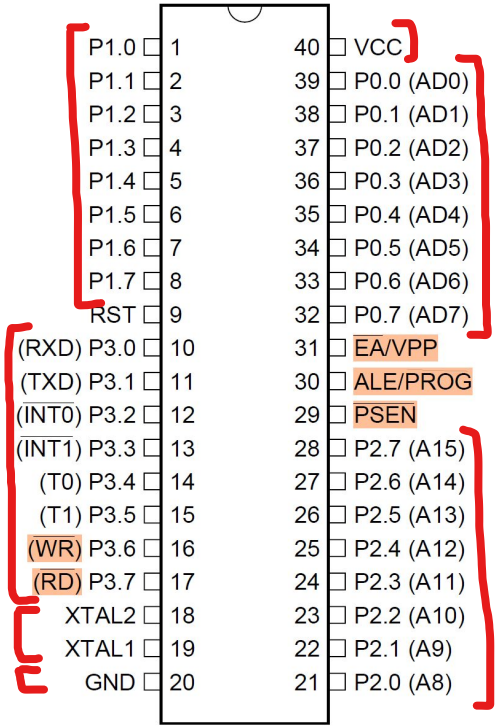
[Week #2 Slides]

Instructor: Shalabh Gupta

8051 Memory Map

Erasable Programmable Read-Only Memory





SFRs are memory-mapped registers used for controlling various hardware features of the microcontroller.

SFRs (Special Function Registers)

	Bit Addressable	8 Bytes Non-bit Addressable							
F8h									FFh
F0h	B	Used in multiplication and division operations							F7h
E8h									EFh
E0h	ACC	Used for arithmetic and logic operations.							E7h
D8h		Program Status Word (Contains flags like Carry, Auxiliary Carry,							DFh
D0h	PSW	Overflow, and the register bank selection bits)							D7h
C8h	(T2CON)		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)			CFh
C0h									C7h
B8h	IP	Interrupt Priority (Sets priority levels for interrupts.)							BFh
B0h	P3	Input/output ports for interfacing external devices							B7h
A8h	IE	Interrupt Enable (Enables or disables interrupts)							AFh
A0h	P2								A7h
98h	SCON	SBUF							9Fh
90h	P1	Timer Control (Controls timer/counter operations)							97h
88h	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	CKCON	8Fh
80h	P0	SP	DPL	DPH				PCON	87h
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

Note: Reserved

SP - Stack Pointer (Points to the current location in the stack)
TMOD - Timer Mode (Configures the mode of timers/counters)
SBUF - Serial Buffer (Used for serial communication (data transmission/reception)

DPH, DPL - Data Pointer Low/High (Used as a 16-bit data pointer for indirect addressing)
SCON - Serial Control (Configures the serial port operation)

An 8051 instruction comprises the following components:

1. **Opcode (Operation Code)**: This specifies the operation to be performed, such as addition, subtraction, data transfer, or logical operations.
2. **Operands**: These specify the data on which the operation is to be performed. Operands can be immediate values, registers, or memory addresses.
3. **Addressing Mode**: This specifies how the operands are accessed. The 8051 supports several addressing modes, including immediate, register, direct, and indirect addressing.
4. **Instruction Format**: The 8051 instructions vary in length, which can be 1 byte, 2 bytes, or 3 bytes depending on the complexity and type of operation.

Example:

- **1-byte instruction**: `ADD A, Rn` (Add the contents of Rn [n: 0 - 7] into the accumulator A)
- **2-byte instruction**: `MOV A, 30H` (Move the value from memory location 30H into the accumulator A)
- **3-byte instruction**: `LJMP 1234H` (Long jump to address 1234H)

Types of instructions

DATA TRANSFER	***ARITHMETIC***	***LOGICAL***	***BOOLEAN***	***PROGRAM BRANCHING***
MOV	ADD	ANL	CLR	LJMP
MOVC	ADDC	ORL	SETB	AJMP
MOVX	SUBB	XRL	MOV	SJMP
PUSH	INC	CLR	JC	JZ
POP	DEC	CPL	JNC	JNZ
XCH	MUL	RL	JB	CJNE
XCHD	DIV	RLC	JNB	DJNZ
	DA A	RR	JBC	NOP
		RRC	ANL	LCALL
		SWAP	ORL	ACALL
			CPL	RET
				RETI
				JMP

Courtesy:
https://oercommons.org/courseware/lesson/69411/student-old/?task=4

8051 Addressing Modes

Register Addressing
ADD A, Rn — Adds the register to the accumulator

Direct Addressing
ADD A, direct — Adds the direct byte to the accumulator

Indirect Addressing
ADD A, @Ri — Adds the indirect RAM to the accumulator

Immediate Constant
ADD A, #data — Adds the immediate data to the accumulator

Instructions by opcode

Instructions by opcode

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0a	0x0b	0x0c	0x0d	0x0e	0x0f
0x00	NOP	AJMP	LJMP	RR	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC
0x10	JBC	ACALL	LCALL	RRC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
0x20	JB	AJMP	RET	RL	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD
0x30	JNB	ACALL	RETI	RLC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC
0x40	JC	AJMP	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL
0x50	JNC	ACALL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL
0x60	JZ	AJMP	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL
0x70	JNZ	ACALL	ORL	JMP	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0x80	SJMP	AJMP	ANL	MOVC	DIV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0x90	MOV	ACALL	MOV	MOVC	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB
0xa0	ORL	AJMP	MOV	INC	MUL	?	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0xb0	ANL	ACALL	CPL	CPL	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE
0xc0	PUSH	AJMP	CLR	CLR	SWAP	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH
0xd0	POP	ACALL	SETB	SETB	DA	DJNZ	XCHD	XCHD	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ
0xe0	MOVX	AJMP	MOVX	MOVX	CLR	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0xf0	MOVX	ACALL	MOVX	MOVX	CPL	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV

Courtesy: https://oercommons.org/courseware/lesson/69411/student-old/?task=4

Examples: Multiple opcodes for an instruction

Can we give these OpCodes instead of the instruction to make the code work??

Instruction	OpCode	Bytes
MOVX @DPTR,A	0xF0	1
MOVX @R0,A	0xF2	1
MOVX @R1,A	0xF3	1
MOVX A,@DPTR	0xE0	1
MOVX A,@R0	0xE2	1
MOVX A,@R1	0xE3	1

Accumulator to External Memory Address at specified register

External Memory to Accumulator

Operation: MOVX

Function: Move Data To/From External Memory (XRAM)

Syntax: MOVX operand1,operand2

Description: MOVX moves a byte to or from External Memory into or from the Accumulator.

If *operand1* is @DPTR, the Accumulator is moved to the 16-bit External Memory address indicated by DPTR. This instruction uses both P0 (port 0) and P2 (port 2) to output the 16-bit address and data. If *operand2* is DPTR then the byte is moved from External Memory into the Accumulator.

If *operand1* is @R0 or @R1, the Accumulator is moved to the 8-bit External Memory address indicated by the specified Register. This instruction uses only P0 (port 0) to output the 8-bit address and data. P2 (port 2) is not affected. If *operand2* is @R0 or @R1 then the byte is moved from External Memory into the Accumulator.

Courtesy: https://oercommons.org/courseware/lesson/69411/student-old/?task=4

Instructions by opcode

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0a	0x0b	0x0c	0x0d	0x0e	0x0f
0x00	NOP	AJMP	LJMP	RR	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC
0x10	JBC	ACALL	LCALL	RRC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
0x20	JB	AJMP	RET	RL	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD
0x30	JNB	ACALL	RETI	RLC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC
0x40	JC	AJMP	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL
0x50	JNC	ACALL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL
0x60	JZ	AJMP	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL
0x70	JNZ	ACALL	ORL	JMP	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0x80	SJMP	AJMP	ANL	MOVC	DIV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0x90	MOV	ACALL	MOV	MOVC	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB
0xa0	ORL	AJMP	MOV	INC	MUL	?	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0xb0	ANL	ACALL	CPL	CPL	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE
0xc0	PUSH	AJMP	CLR	CLR	SWAP	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH
0xd0	POP	ACALL	SETB	SETB	DA	DJNZ	XCHD	XCHD	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ
0xe0	MOVX	AJMP	MOVX	MOVX	CLR	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0xf0	MOVX	ACALL	MOVX	MOVX	CPL	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV