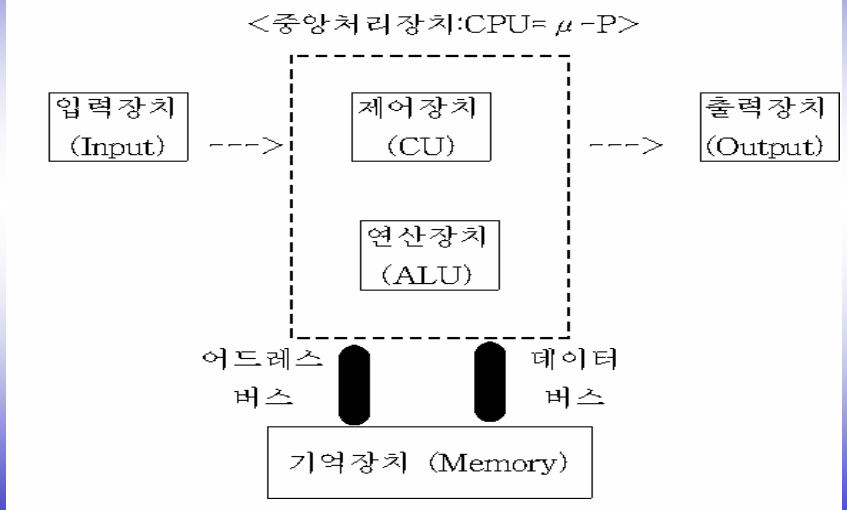
임베디드 기초 프로그래밍
(마이크로프로세서 실습)
ch1. MCS-51의 구조와 특성

정보네트워크공학전공
이훈재
hjlee@dongseo.ac.kr
http://crypto.dongseo.ac.kr
http://kowon.dongseo.ac.kr/~hjlee

MCS-51 마이크로프로세서 내부 블록 구조



MCS-51 마이크로프로세서 소개

표 1·1 MCS-51 마이크로 프로세서

명 칭	ROMLESS	EPROM	ROM(바이트)	RAM(바이트)	16비트 타이머	회로형식
8051	8031	(8751)	4 K	128	2	HMOS
8051AH	8031AH	8751H	4 K	128	2	HMOS
8052AH	8032AH	8752BH	8 K	256	3	HMOS
80C51BH	80C31BH	87C51	8 K	128	2	CHMOS

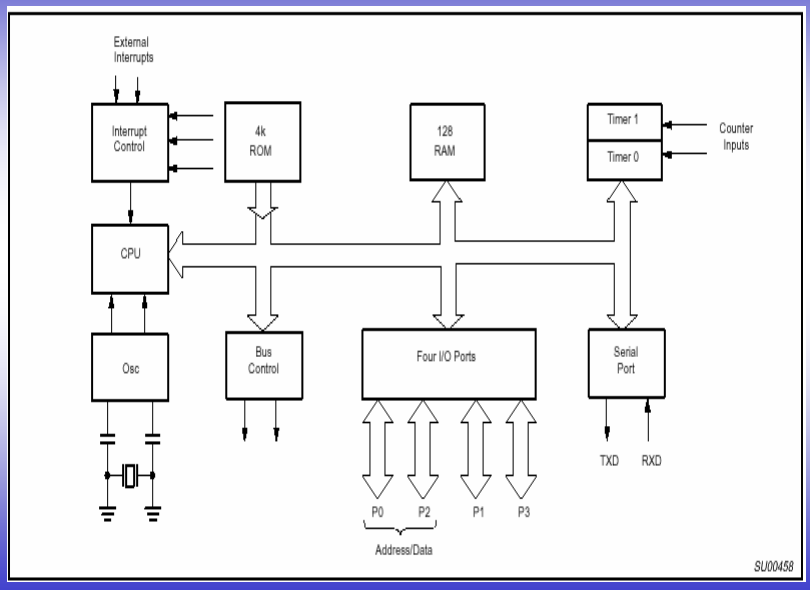
89C51: EEPROM

MCS-51 마이크로프로세서 소개

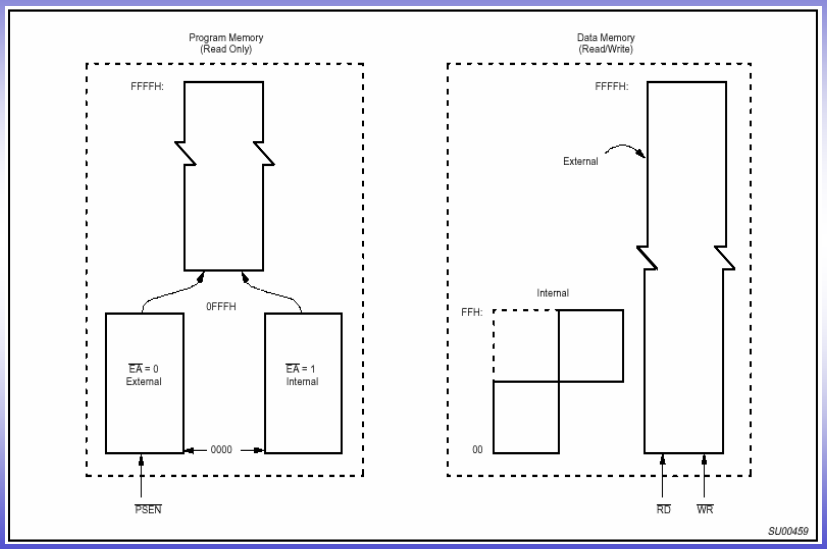
◆8051 마이크로 프로세서 특징

- ◆ 1981년 생산 시작
- ◆ 제어응용설계 전문의 8비트 CPU
- ◆ 강화된 부울대수의 연산능력
- ◆ 32개의 양방향 I/O
- ◆ 128 바이트의 저장 가능한 데이터 메모리(DATA RAM)
- ◆ 2개의 16비트 타이머 내장
- ◆ 완전한 이중 UART
- ◆ 5개의 인터럽트 가운데 이중(고/저)의 우선순위를 가진 인터럽트 보유
- ◆ 클럭(clock) 발진기 내장
- ◆ 4K(8K/8052) 바이트의 ROM 내장(내부 프로그램 메모리)
- ◆ 64K 바이트의 프로그램 영역(외부) → (8Mbyte 확장 가능함)
- ◆ 64K 바이트의 데이터 메모리 어드레스 영역 → (더 이상 확장가능)

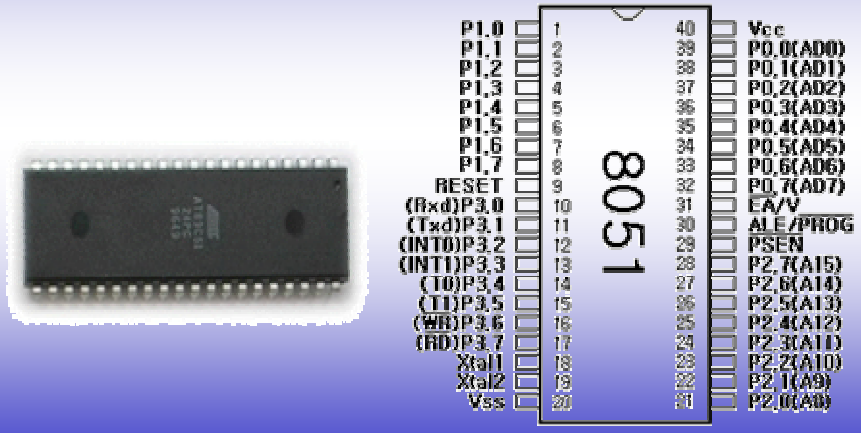
MCS-51 마이크로프로세서 소개



MCS-51 메모리 맵(memory map)



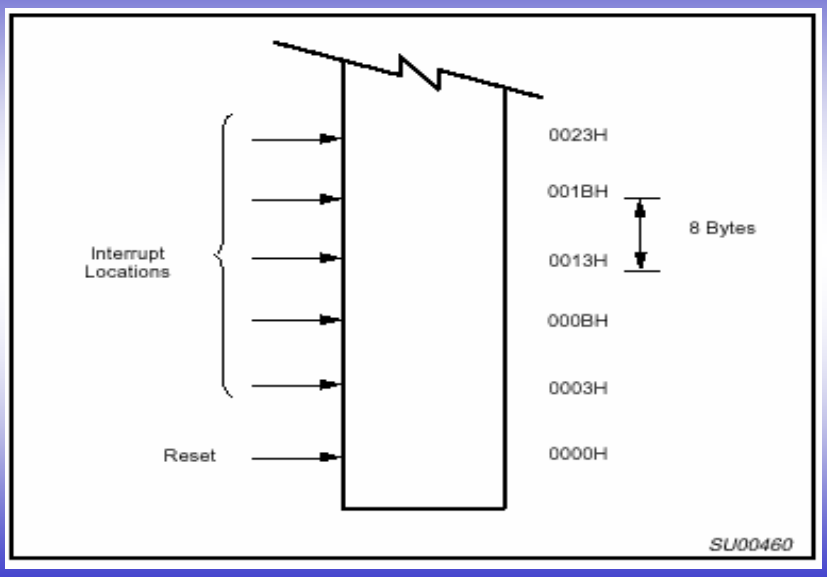
MCS-51 마이크로프로세서 외부핀(1)



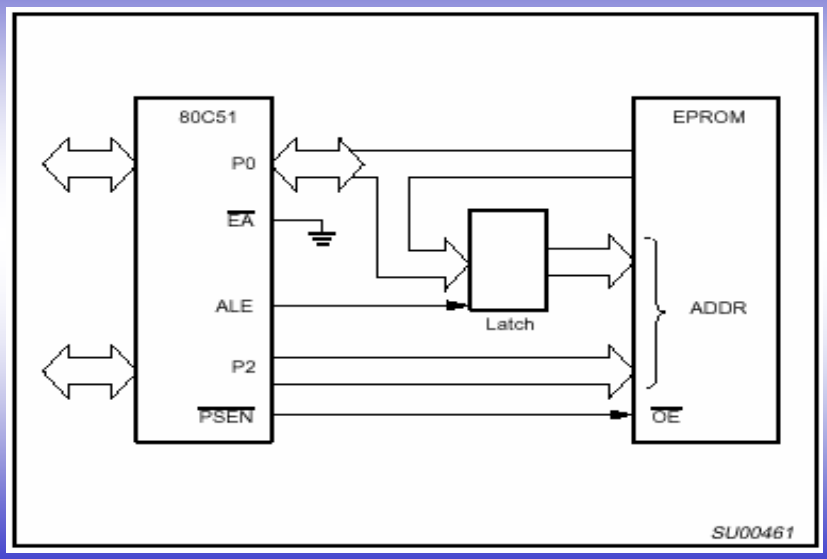
MCS-51 마이크로프로세서 외부핀(2)

핀 이름	내 용
RxD	시리얼 입력 포트
TxD	시리얼 출력 포트
INT0	외부 인터럽트 0
INT1	외부 인터럽트 1
T0	타이머 0 외부 입력
T1	타이머 1 외부 입력
WR	외부 데이터 메모리 출력 스트로브 (Strobe)
RD	외부 데이터 메모리 입력 스트로브
RESET	시스템 리셋
ALE/PROG	Address Latch Enable : 외부 기억 소자를 처리하는 동안 하위 번지 래치
PSEN	Program Strobe Enable : 외부 프로그램 메모리에 대해 리드하기 위한 신호
EA/Vpp	External Access Enable : 외부 프로그램 메모리를 사용 가능
XTAL 1, 2	반전된 발진 증폭기에 대한 입력, 출력
P0 ~P3	입출력 포트
A0 ~A15	어드레스(주소 지정) 버스
D0 ~D7	데이터 버스

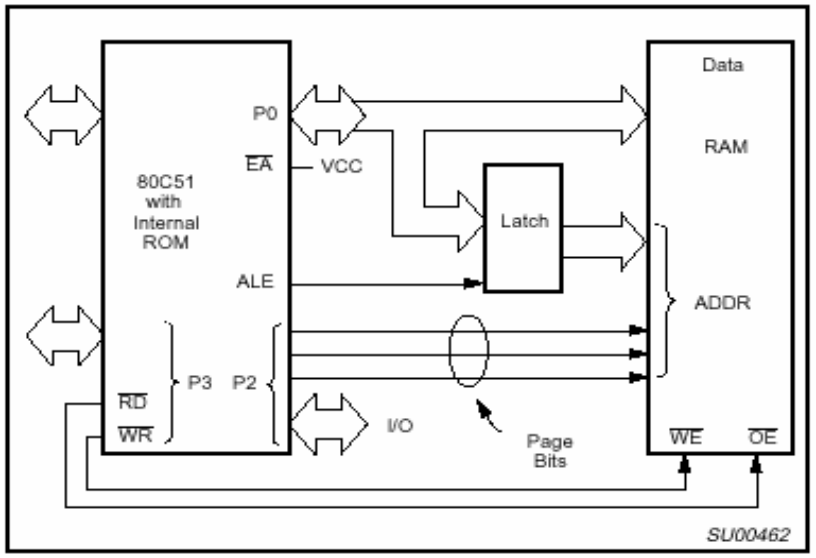
MCS-51 프로그램 메모리(program memory)



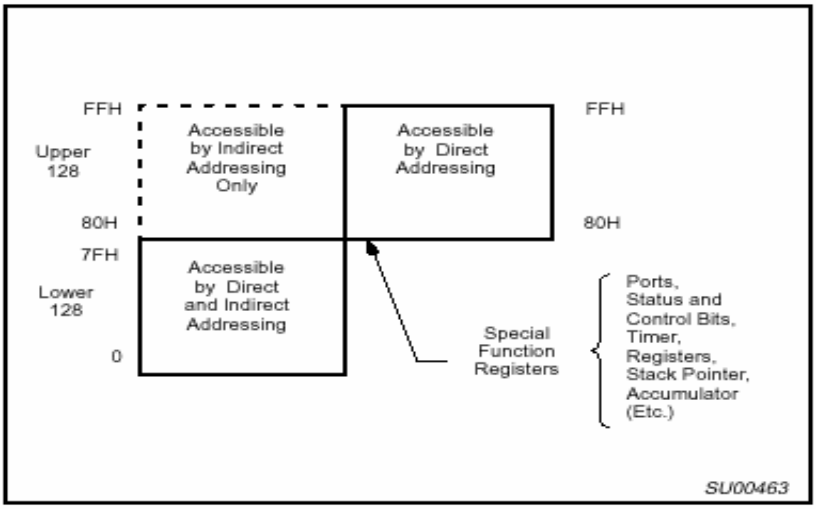
외부확장 메모리 인터페이스
- External Program Memory



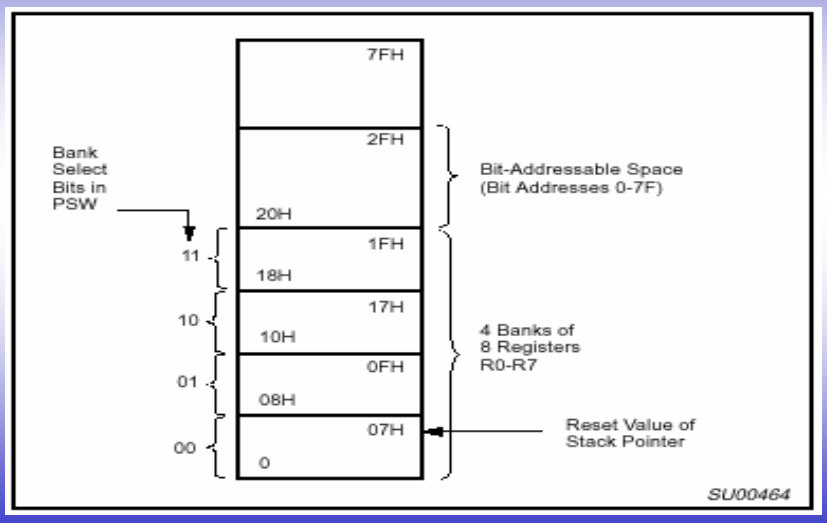
외부확장 메모리 인터페이스 - External Data Memory



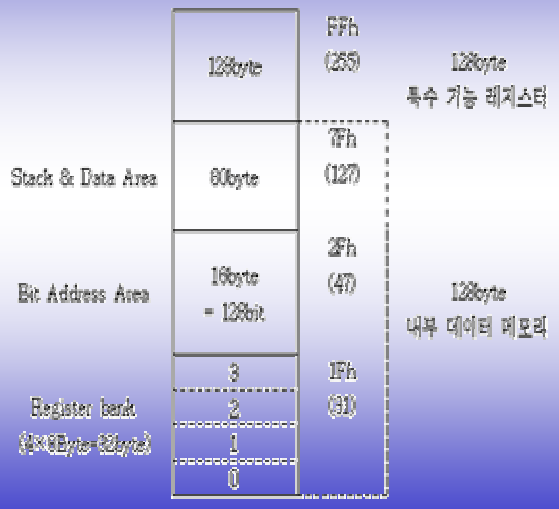
내부 메모리 구성(1)



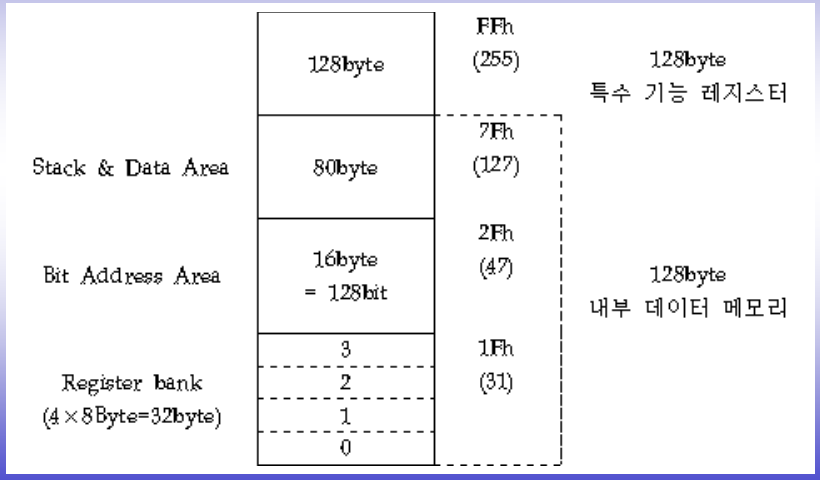
내부 메모리 구성(2)
- 하위 128 바이트



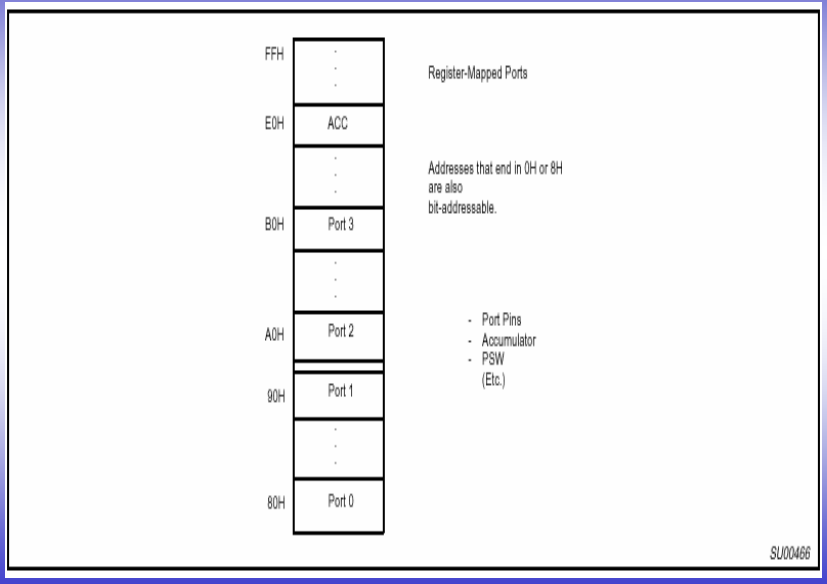
MCS-51 데이터 메모리(data memory)



내부 메모리 구성



Acc, B, PSW, Port address



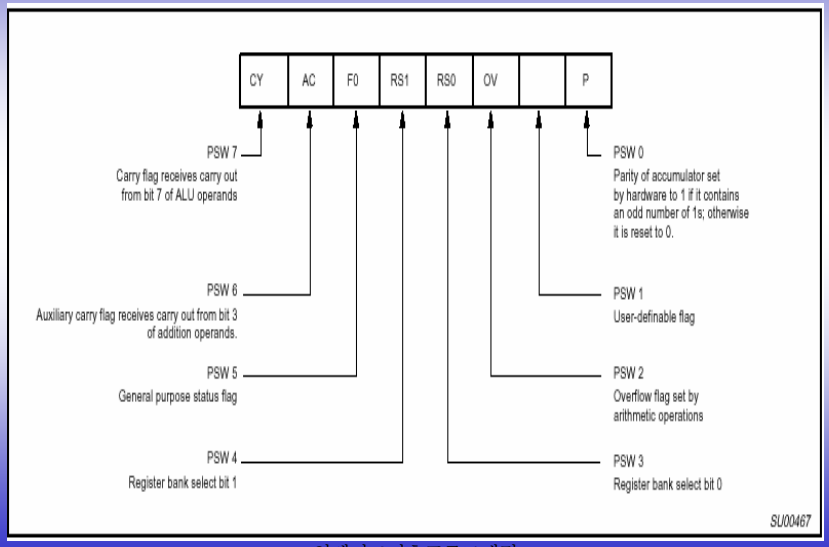
SFR 레지스터(1)

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8									CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

SFR 레지스터(2)

기 호	명 칭	주 소
ACC	Accmulator	0E0h
B	B Register	0F0h
PSW	Program Status Word	0D0h
SP	Stack Pointer	81h
DPTR	Data Pointer Register	
DPL	Low Byte	82h
DPH	High Byte	83h
P0	Port 0	80h
P1	Port 1	90h
P2	Port 2	0A0h
P3	Port 3	0B0h
IP	Interrupt Priority Control	0B8h
IE	Interrupt Enable Control	0A8h
TMOD	Timer/Counter Mode Control	89h
TCON	Timer/Counter Control	88h
TH0	Timer/Counter 0 High Byte	8Ch
TL0	Timer/Counter 0 Low Byte	8Ah
TH1	Timer/Counter 1 High Byte	8Dh
TL1	Timer/Counter 1 Low Byte	8Bh
SCON	Serial Control	98h
SBUF	Serial Data Buffer	99h
PCON	Power Control	87h

PSW 레지스터 (Program Status Word)



1.4 Addressing Mode(1)

◆ 직접 어드레싱(direct addressing)
- 내부 데이터 메모리와 SFR은 직접 어드레싱 가능함.
- 예) ADD A, 7FH
  ANL A, 7FH

◆ 간접 어드레싱(indirect addressing)
- 내부 데이터 메모리와 외부 데이터 메모리 모두 간접 어드레싱 가능함.
- 예) ADD A, @R0
  ORL A, @R1

◆ 레지스터 어드레싱(register addressing)
- 레지스터 뱅크 내의 R0~R7에 대한 직접적인 어드레싱
- 예) ADD A, R2
  XRL A, R3

1.4 Addressing Mode(2)

◆ 레지스터 지정 어드레싱(register indirect addressing)

- 레지스터 뱅크 내의 R0~R7의 내장값에 대한 어드레싱
- 예) ADD A, @R4 - -> Acc + the value of address of R4
  = Acc
  XRL A, @R5 - -> Acc ^ the value of address of R5
  = Acc

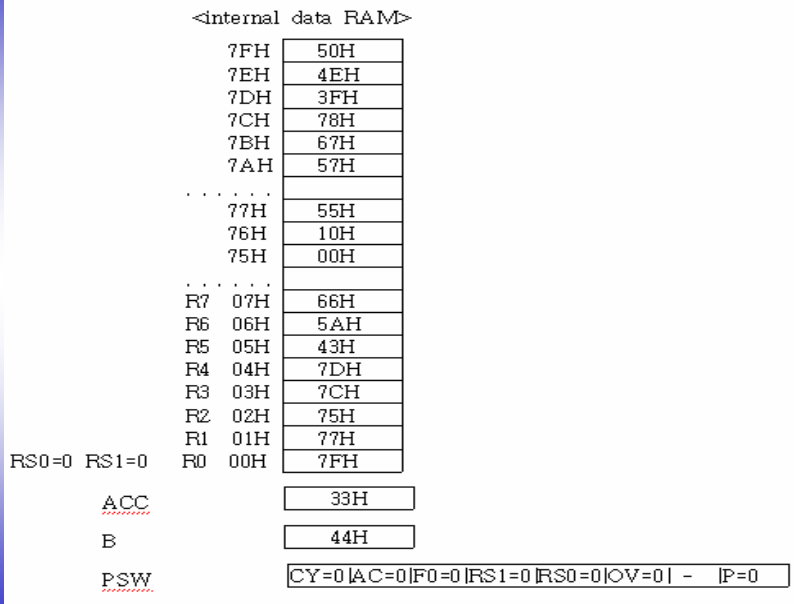
◆ 상수 어드레싱(immediate addressing)

- 임의의 상수값을 오퍼랜드로 하는 어드레싱
- 예) ADD A, #127
  XRL A, #FFH

◆ 색인 어드레싱(index addressing)

- 프로그램 메모리만 가능, 테이블 지정시 유리

1.4 Addressing Mode -예제(1)



1.4 Addressing Mode - 예제(2)

[예제1] 각 상태의 수행 결과는? (개별 결과 즉, 연속 실행 결과가 아님)

- direct addressing(DIR): ADD A, 7FH ⇒ CY=☐ ACC ☐
- indirect addressing(IND): ADD A, @R0 ⇒ CY=☐ ACC ☐
- register addressing(REG): ADD A, R2 ⇒ CY=☐ ACC ☐
- immediate addressing(IMM): ADD A, #127 ⇒ CY=☐ ACC ☐

[예제2] 각 상태의 수행 결과는? (개별 결과 즉, 연속 실행 결과가 아님)

- direct addressing(DIR): ANL A, 7EH ⇒ CY=☐ ACC ☐
- indirect addressing(IND): ORL A, @R1 ⇒ CY=☐ ACC ☐
- register addressing(REG): XRL A, R3 ⇒ CY=☐ ACC ☐
- immediate addressing(IMM): XRL A, #0FFH ⇒ CY=☐ ACC ☐

1.5 Arithmetic 명령어

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (μs)
ADD A,<byte>	A = A + <byte>	X	X	X	X	1
ADDC A,<byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A,<byte>	A = A – <byte> – C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A – 1	Accumulator only				1
DEC <byte>	<byte> = <byte> – 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B only				4
DIV AB	A = Int[A/B] B = Mod[A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

```
ADD     a, 7FH (direct addressing)
ADD     A, @R0 (indirect addressing)
ADD     a, R7 (register addressing)
ADD     A, #127 (immediate constant)
```

1.6 Logical Instructions

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (μs)
ANL A,<byte>	A = A.AND. <byte>	X	X	X	X	1
ANL <byte>,A	<byte> = <byte> .AND.A	X				1
ANL <byte>,#data	<byte> = <byte> .AND.#data	X				2
ORL A,<byte>	A = A.OR.<byte>	X	X	X	X	1
ORL <byte>,A	<byte> = <byte> .OR.A	X				1
ORL <byte>,#data	<byte> = <byte> .OR.#data	X				2
XRL A,<byte>	A = A.XOR. <byte>	X	X	X	X	1
XRL <byte>,A	<byte> = <byte> .XOR.A	X				1
XRL <byte>,#data	<byte> = <byte> .XOR.#data	X				2
CRL A	A = 00H	Accumulator only				1
CPL A	A = .NOT.A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

```
ANL   A,7FH (direct addressing)
ANL   A,@R1 (indirect addressing)
ANL   A,R6 (register addressing)
ANL   A,#53H (immediate constant)
```


1.7 Data Transfer Instructions(1)

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (μs)
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>,A	<dest> = A	X	X	X		1
MOV <dest>,<src>	<dest> = <src>	X	X	X	X	2
MOV DPTR,#data16	DPTR = 16-bit immediate constant				X	2
PUSH <src>	INC SP:MOV"@SP",<src>	X				2
POP <dest>	MOV <dest>,"@SP":DEC SP	X				2
XCH A,<byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A,@Ri	ACC and @Ri exchange low nibbles		X			1

1.7 Data Transfer Instructions(2)

		2A	2B	2C	2D	2E	ACC
MOV	A,2EH	00	12	34	56	78	78
MOV	2EH,2DH	00	12	34	56	56	78
MOV	2DH,2CH	00	12	34	34	56	78
MOV	2CH,2BH	00	12	12	34	56	78
MOV	2BH,#0	00	00	12	34	56	78

A. Using direct MOVs: 14 bytes, 9 μs

		2A	2B	2C	2D	2E	ACC
CLR	A	00	12	34	56	78	00
XCH	A,2BH	00	00	34	56	78	12
XCH	A,2CH	00	00	12	56	78	34
XCH	A,2DH	00	00	12	34	78	56
XCH	A2EH	00	00	12	34	56	78

B. Using XCHs: 9 bytes, 5 μs

SU00468

1.7 Data Transfer Instructions(3)
- 외부 데이터 메모리 억세스

ADDRESS WIDTH	MNEMONIC	OPERATION	EXECUTION TIME (μs)
8 bits	MOVX A,@Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri,A	Write external RAM @ Ri	2
16 bits	MOVX A,@DPTR	Read external RAM @ DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM @ DPTR	2

1.7 Data Transfer Instructions(4)
- 프로그램 메모리 억세스

MNEMONIC	OPERATION	EXECUTION TIME (μs)
MOVC A,@A+DPTR	Read program memory at (A + DPTR)	2
MOVC A,@A+PC	Read program memory at (A + PC)	2

1.8 Boolean Instructions

MNEMONIC		OPERATION	EXECUTION TIME (μs)
ANL	C,bit	C = C.AND.bit	2
ANL	C,/bit	C = C.AND..NOT.bit	2
ORL	C,bit	C = C.OR.bit	2
ORL	C,/bit	C = C.OR..NOT.bit	2
MOV	C,bit	C = bit	1
MOV	bit,C	bit = C	2
CLR	C	C = 0	1
CLR	bit	bit = 0	1
SETB	C	C = 1	1
SETB	bit	bit = 1	1
CPL	C	C = .NOT.C	1
CPL	bit	bit = .NOT.bit	1
JC	rel	Jump if C = 1	2
JNC	rel	Jump if C = 0	2
JB	bit,rel	Jump if bit = 1	2
JNB	bit,rel	Jump if bit = 0	2
JBC	bit,rel	Jump if bit = 1; CLR bit	2

1.9 Relative Offset Instructions(1) - Unconditional Jump

MNEMONIC	OPERATION	EXECUTION TIME (μs)
JMP addr	Jump to addr	2
JMP @A+DPTR	Jump to A + DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

```
MOV   DPTR,#JUMP TABLE
MOV   A,INDEX_NUMBER
RL    A
JMP   @A+DPTR
```

```
AJMP   CASE 0
AJMP   CASE 1
AJMP   CASE 2
AJMP   CASE 3
AJMP   CASE 4
```


1.9 Relative Offset Instructions(2) - Conditional Jump

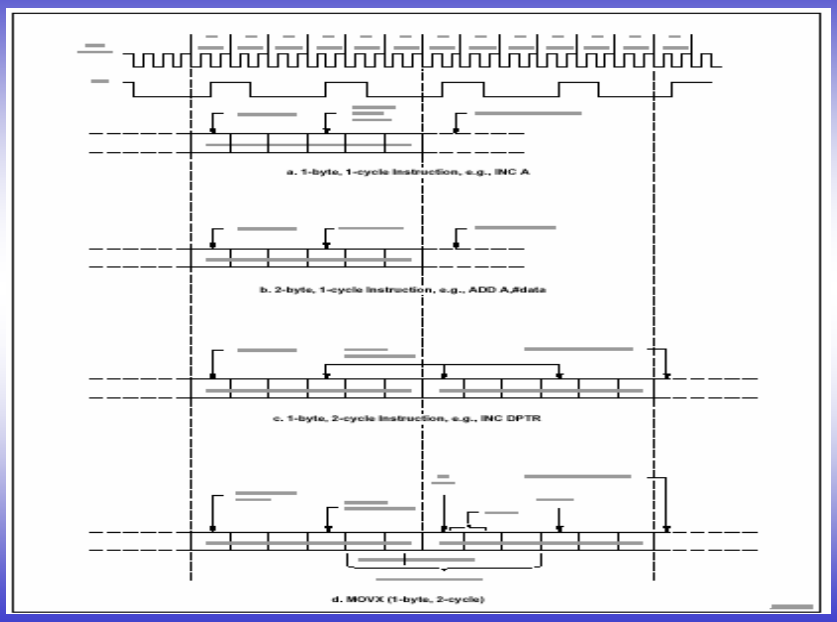
MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION TIME (μs)
		DIR	IND	REG	IMM	
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNE A,<byte>,rel	Jump if A ≠ <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

```
        MOV     COUNTER,#10
LOOP: (begin loop)
        •
        •
        •
        (end loop)
        DJNZ    COUNTER,LOOP
        (continue)
```

Branch 명령어

니모닉	동 작	어드레싱 모드(명령의 바이트 수)				실행시간 (머신 사이클)
		Direct	Indirect	Register	Immediate	
SJMP rel	Jump using relative address	2				2
AJMP addr11	Jump using 11-bit address	2				2
LJMP addr16	Jump using 16-bit address	3				2
JMP @A+DPTR	Jump to A+DPTR	1				2
JZ rel	Jump if A = 0	누산기 A에만 적용(2)				2
JNZ rel	Jump if A ≠ 0	누산기 A에만 적용(2)				2
JC rel	Jump if CY = 1	(2)				2
JNC rel	Jump if CY = 0	(2)				2
JB bit,rel	Jump if bit = 1	(3)				2
JNB bit,rel	Jump if bit = 0	(3)				2
JBC bit,rel	Jump if bit = 1, and clear bit	(3)				2
DJNZ <byte>,rel	Decrement and jump if not zero	3		2		2
CJNE A,<byte>,rel	Jump if A ≠ <byte>	3			3	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		3	3		2
ACALL addr11	Call subroutine using 11-bit address	2				2
LCALL addr16	Call subroutine using 16-bit address	3				2
RET	Return from subroutine	1				2
RETI	Return from interrupt	1				2
NOP	No operation	1				1

1.10 CPU Timing(1)



1.10 CPU Timing(2)