The 8051

29/09/2020

S SUBRAHMANYAM

BEC,BAPATLA

8051

- Today over fifty companies produce variations of the 8051.
- Several of these companies have over fifty versions of the 8051.
- 8051 cores are available for implementations in FPGA's or ASIC's.
- Over 100 million 8051's are sold each year.
- The 8051 has been extremely successful, and has directly influenced many of the more recent microcontroller architectures.

8051 software

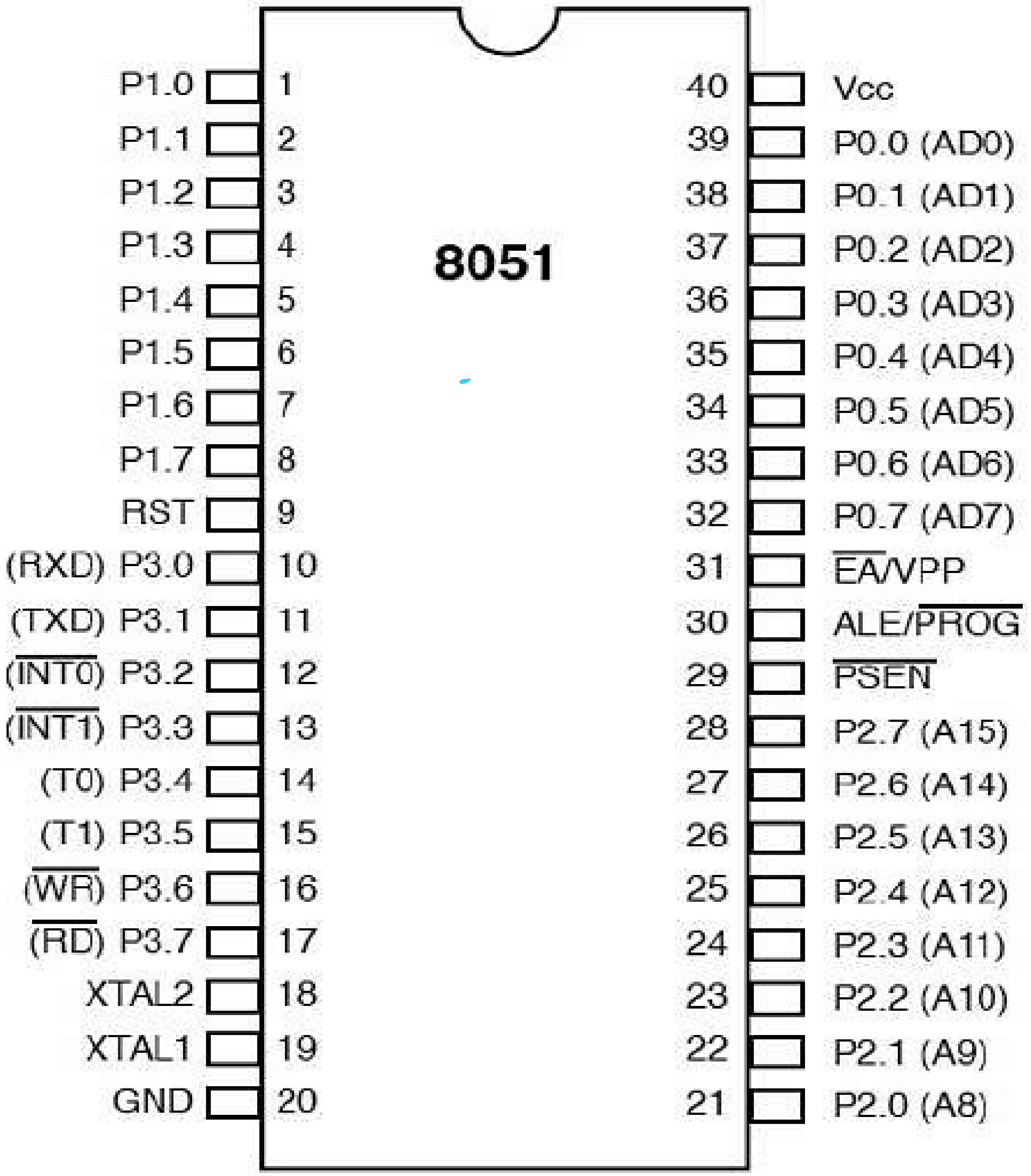
- Download the Silicon Labs software form the class web page. Don’t use the new version on the Silicon Labs web page.
- Download my tutorial on using the Silicon Labs University Daughter Card and go through both the C and assembly language tutorials.
- See handout on 8051 instruction set.
- The MCS 51 Family User’s Manual is available on the class web page.
  - Look under Resources - Other

MCS-51

- MCS-51 is Intel's designation for its family of 8051 devices.
- The 8051 is the original member of the MCS-51 family, and is the core for all MCS-51 devices.
- The original 8051 was available in three versions.
  - 8051 – A fixed program in read only memory (ROM) version.
  - 8031 – No internal ROM program stored in external programmable read only memory (PROM) memory.
  - 8751 – Program stored in internal erasable PROM (EPROM). Erased by exposing the chip to high intensity ultraviolet light for several minutes. Eventually EPROM was replaced by EEPROM.

The basic 8051 Core

- 8-bit CPU optimized for control applications
- Capability for single bit Boolean operations.
- Supports up to 64K of program memory.
- Supports up to 64K of program memory.
- 4 K bytes of on-chip program memory.
    - Newer devices provide more.
- 128 or 256 bytes of on-chip data RAM
- Four 8 bit ports.
- Two 16-bit timer/counters
- UART
- Interrupts
- On-chip clock oscillator


8051
P1.0 1
P1.1 2
P1.2 3
P1.3 4
P1.4 5
P1.5 6
P1.6 7
P1.7 8
RST 9
(RXD) P3.0 10
(TXD) P3.1 11
($\overline{INT0}$) P3.2 12
($\overline{INT1}$) P3.3 13
(T0) P3.4 14
(T1) P3.5 15
($\overline{WR}$) P3.6 16
($\overline{RD}$) P3.7 17
XTAL2 18
XTAL1 19
GND 20
40 Vcc
39 P0.0 (AD0)
38 P0.1 (AD1)
37 P0.2 (AD2)
36 P0.3 (AD3)
35 P0.4 (AD4)
34 P0.5 (AD5)
33 P0.6 (AD6)
32 P0.7 (AD7)
31 $\overline{EA}$/VPP
30 ALE/$\overline{PROG}$
29 $\overline{PSEN}$
28 P2.7 (A15)
27 P2.6 (A14)
26 P2.5 (A13)
25 P2.4 (A12)
24 P2.3 (A11)
23 P2.2 (A10)
22 P2.1 (A9)
21 P2.0 (A8)

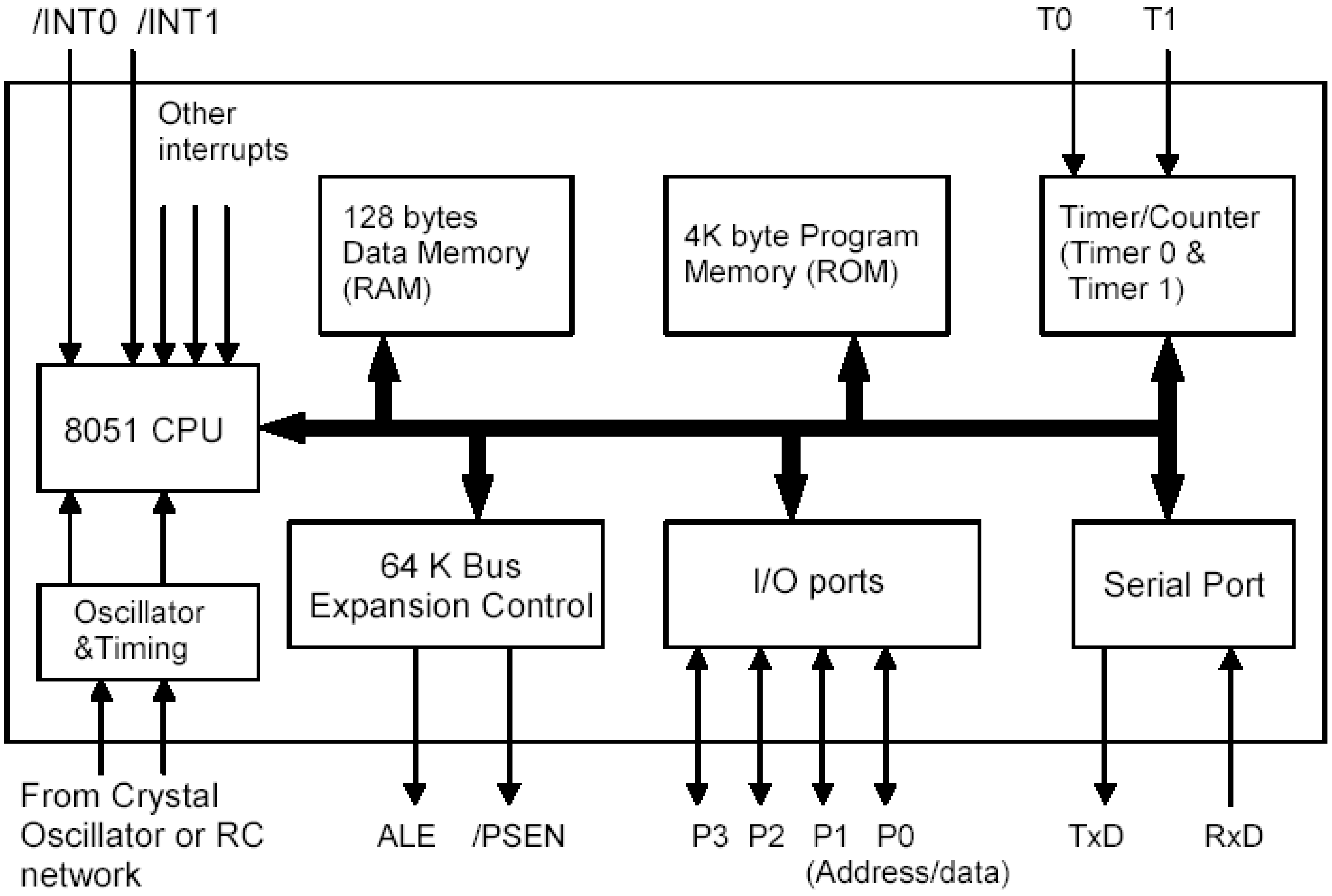

/INT0
/INT1
T0
T1
Other interrupts
128 bytes Data Memory (RAM)
4K byte Program Memory (ROM)
Timer/Counter (Timer 0 & Timer 1)
8051 CPU
64 K Bus Expansion Control
I/O ports
Serial Port
Oscillator &Timing
From Crystal Oscillator or RC network
ALE
/PSEN
P3
P2
P1
P0 (Address/data)
TxD
RxD

Figure 1.1 Block Diagram of the generic 8051 Microcontroller

MEMORY

CODE MEMORY Internal ROM selected by $\overline{EA} = 1$

CM = CM(0..0FFFFH) = CM(0..0FFFFH;7..0)^1

ON CHIP DATA MEMORY

DM = DM(0..7FH) = DM(0..7FH;7..0)

ON CHIP BIT ADDRESSIABLE MEMORY

BADM = BADM(0..0FF) = BADM(0..0FF;0) ;BIT ADDR DATA MEMORY

EXTERNAL RAM MEMORY

XM = XM(0..0FFFFH) = XM(0..0FFFFH;7..0)

INTERNAL REGISTERS AND PORTS

PC = PC(15..0)

SP = SP(7..0)

DPTR = DPTR(15..0)

PSW = CY|AC|F0|RS1|RS0|OV|P

TCON = TF1|TR1|TF0|TR0|IE1|IT1|IE0|IT0

Memory Organization

- The 8051 memory organization is rather complex.
- The 8051 has separate address spaces for Program Memory, Data Memory, and external RAM.
- This is refereed to as a Harvard architecture.
  - The early Mark I (1944) computer developed at Harvard was of this type of architecture.
  - Von Neumann at Princeton pointed out that it was not necessary to put instructions and data in separate memories.
  - Most machines have been Princeton architecture.
  - Recently Harvard architecture has been employed to help alleviate the memory bottleneck.
- Both program memory and external data memory are 8 bits wide and use 16 bits of address. The internal data memory is accessed using an 8-bit address.
- Since the same address can refer to different locations the specific location is determined by the type of instruction.

Program or Code Memory

- May consist of internal or external program memory. The amount of internal program memory varies depending on the device.
  - 4K bytes typical in older devices.
  - The Silicon Labs C8051F310 contains 16K of flash memory for programs.
  - The Silicon Labs C8051F020 which is on the University Daughter Card (UDC) contains 4K bytes of program memory.
- The MOVC instruction can be use to read code memory.
- To reference code memory I will use the notation:

CM = CM(0,…,FFFFH) = CM(0,…,FFFFH; 7,…,0)

- This notation can be used to specify particular bits and bytes of code memory.

For example CM(1234H) refers to the byte of code memory at address 1234H. CM(1234H;7) refers to the most significant bit in that address.

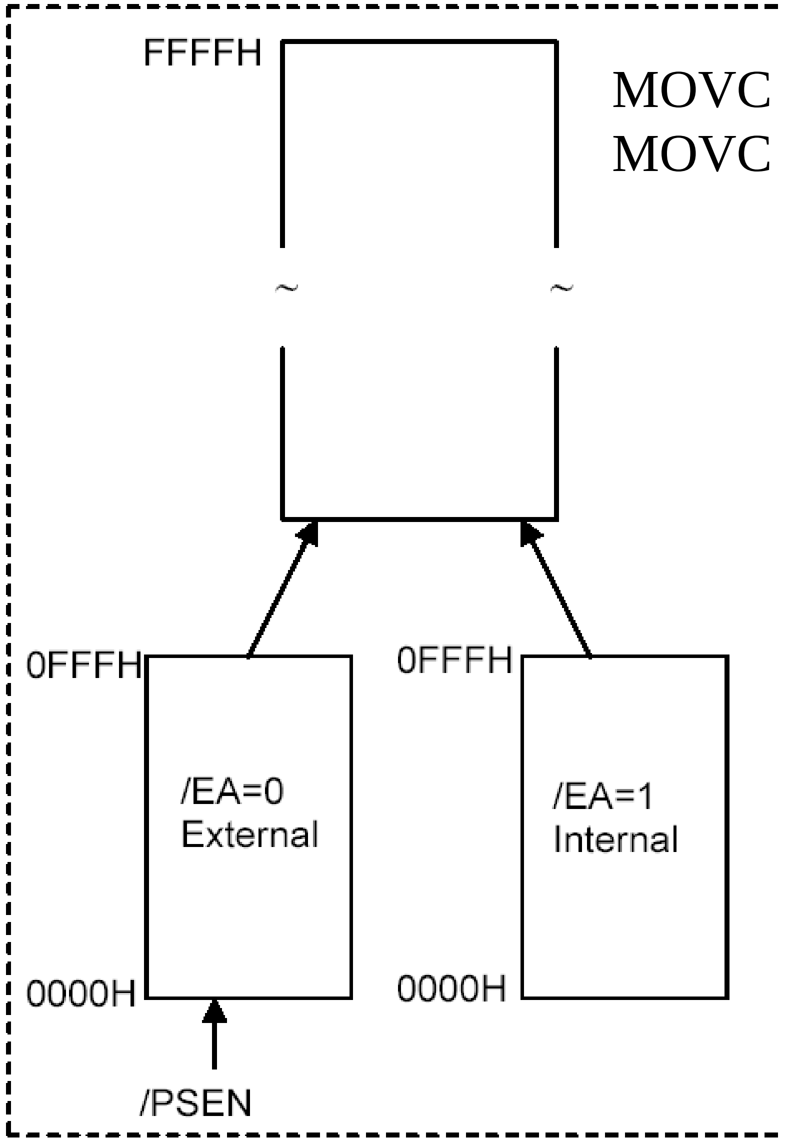


Figure 1.4 Program Memory Organization (Read Only)

MOVC A,@A + DPTR ;A ← CM(A+DPTR)
MOVC A,@A + PC ;A ← CM(A+PC)

CM

PC = PC(15..0)

DPTR = DPTR(15..0)

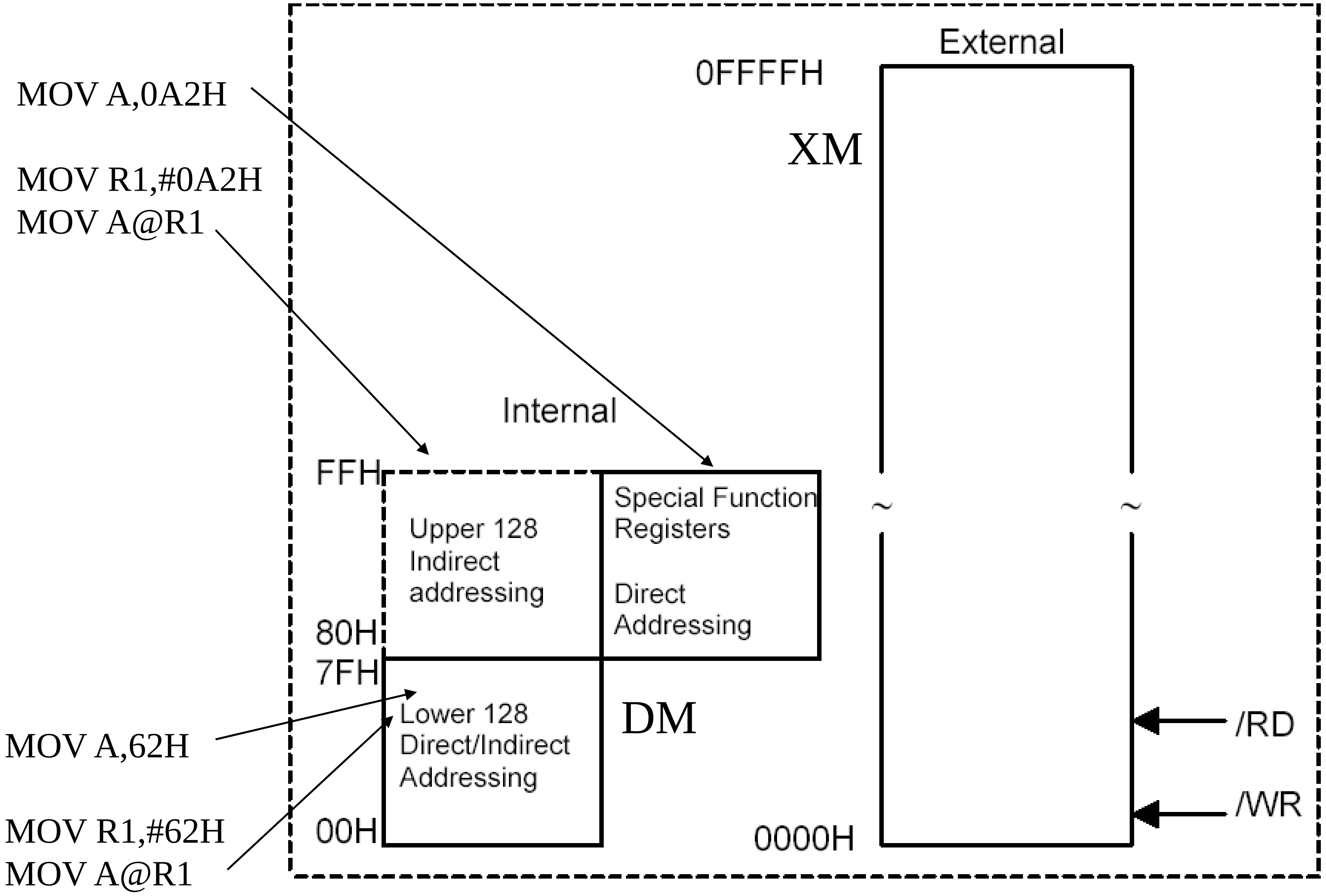
External Memory

- Supports up to 64K bytes external memory.
  - XM(0000,…,FFFF)
    = XM(0000,…,FFFF; 7,…,0)
  - Accessed by using the MOVX instruction.
- On the original using external memory reduces number of available I/O ports.
- On some new devices this is not the case.
  - For example in C8051F020 64K bytes of external memory has been included in the chip.
  - The 4 standard 8051 ports are available and three additional ports have been added.

```
MOVX  A,@DPTR        ;A ← XM(DPTR)
MOVX  A,@Rn          ;A ← XM(P2|Rn)
MOVX  @DPTR,A        ;XM(DPTR) ← A
MOVX  @Rn,A          ;XM(P2|Rn) ← A
```

Data Memory

- The original 8051 had 128 bytes of on-chip data RAM.
  - This memory includes 4 banks of general purpose registers at DM(00..1F)
  - Only one bank can be active at a time.
  - If all four banks are used, DM(20..7F) is available for program data.
  - DM(20..2F) is bit addressable as BADM(00..7F).
- DM(80,…,FF) contains the special function registers such as I/O ports, timers, UART, etc.
  - Some of these are bit addressable using BADM(80..FF)
- On newer versions of the 8051, DM(80,…,FF) is also use as data memory. Thus, the special functions registers and data memory occupy the same address space. Which is accessed is determined by the instruction being used.


MOV A,0A2H
MOV R1,#0A2H
MOV A@R1
MOV A,62H
MOV R1,#62H
MOV A@R1
External
0FFFFH
XM
Internal
FFH
Upper 128
Indirect
addressing
Special Function
Registers
Direct
Addressing
80H
7FH
Lower 128
Direct/Indirect
Addressing
DM
00H
0000H
/RD
/WR

Data memory

	Byte Address	Bit Address							
	7F	General Purpose RAM							
	30								
B	2F	7F	7E	7D	7C	7B	7A	79	78
i	2E	77	76	75	74	73	72	71	70
t	2D	6F	6E	6D	6C	6B	6A	69	68
	2C	67	66	65	64	63	62	61	60
A	2B	5F	5E	5D	5C	5B	5A	59	58
d	2A	57	56	55	54	53	52	51	50
d	29	4F	4E	4D	4C	4B	4A	49	48
r	28	47	46	45	44	43	42	41	40
e	27	3F	3E	3D	3C	3B	3A	39	38
s	26	37	36	35	34	33	32	31	30
s	25	2F	2E	2D	2C	2B	2A	29	28
a	24	27	26	25	24	23	22	21	20
b	23	1F	1E	1D	1C	1B	1A	19	18
l	22	17	16	15	14	13	12	11	10
e	21	0F	0E	0D	0C	0B	0A	09	08
	20	07	06	05	04	03	02	01	00
	1F	Bank 3							
	18								
	17	Bank 2							
	10								
	0F	Bank 1							
	08								
	07	Default Register Bank for R0 – R7							
	00								

Byte Address	Bit Address								
FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	-	D0	PSW
B8	-	-	-	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	-	-	AC	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	Not bit-addressable								SBUF
98	9F	96	95	94	93	92	91	90	SCON
90	97	96	95	94	93	92	91	90	P1
8D	Not bit-addressable								TH1
8C	Not bit-addressable								TH0
8B	Not bit-addressable								TL1
8A	Not bit-addressable								TL0
89	Not bit-addressable								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	Not bit-addressable								PCON
83	Not bit-addressable								DPH
82	Not bit-addressable								DPL
81	Not bit-addressable								SP
80	87	86	85	84	83	82	81	80	P0

Data Memory (DM)

Table 1

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
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
*+T2CON	Timer/Counter 2 Control	0C8H
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
+TH2	Timer/Counter 2 High Byte	0CDH
+TL2	Timer/Counter 2 Low Byte	0CCH
+RCAP2H	T/C 2 Capture Reg. High Byte	0CBH
+RCAP2L	T/C 2 Capture Reg. Low Byte	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

* = Bit addressable
+ = 8052 only

SFR MEMORY MAP

8 Bytes

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			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

↑
Bit
Addressable

Figure 5

PSW: PROGRAM STATUS WORD. BIT ADDRESSABLE.

CY	AC	F0	RS1	RS0	OV	—	P

CY	PSW.7	Carry Flag.
AC	PSW.6	Auxiliary Carry Flag.
F0	PSW.5	Flag 0 available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1 (SEE NOTE 1).
RS0	PSW.3	Register Bank selector bit 0 (SEE NOTE 1).
OV	PSW.2	Overflow Flag.
—	PSW.1	User definable flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of '1' bits in the accumulator.

NOTE:

1. The value presented by RS0 and RS1 selects the corresponding register bank.

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

INTERNAL REGISTERS AND PORTS

PC = PC(15..0)

SP = SP(7..0)

DPTR = DPTR(15..0)

PSW = CY|AC|F0|RS1|RS0|OV|P

TCON = TF1|TR1|TF0|TR0|IE1|IT1|IE0|IT0

BADM(87H..80H) = **P0** = P0(7..0) = BADM(87H..80H)

BADM(8FH..88H) = **TCON** =BADM(8FH..88H)

BADM(97H..90H) = **P1** = P1(7..0) =BADM(97H..90H)

BADM(9FH..98H) = **SCON** =BADM(9FH..98H)

BADM(A7H..A0H) = **P2** = P2(7..0) =BADM(A7H..A0H)

BADM(AFH..A8H) = **IE** =BADM(AFH..A8H)

BADM(B7H..B0H) = **P3** = P3(7..0) =BADM(B7H..B0H)

BADM(BFH..B8H) = **IP** =BADM(BFH..B8H)

BADM(C7H..C0H) = BADM(C7H..C0H)

BADM(CFH..C8H) = BADM(CFH..C8H)

BADM(D7H..D0H) = **PSW** =BADM(D7H..D0H)

BADM(DFH..D8H) = BADM(DFH..D8H)

BADM(E7H..E0H) = **ACC** = **A** = A(7..0) = BADM(E7H..E0H)

BADM(EFH..E8H) = BADM(EFH..E8H)

BADM(F7H..F0H) = **B** =BADM(F7H..F0H)

BADM(FFH..F8H) = BADM(FFH..F8H)

RESET

PC ← 0, A ← 0, B ← 0, PSW ← 0, SP ← 7H, SPTR ← 0, P0-P3 ← 0FFH,
IP ← XXX00000B, IE ← 0XX00000B, TMOD ← 0, TCON ← 0, TH0 ← 0, TL0 ← 0,
TH1 ← 0, TL1 ← 0, SCON ← 0, PCON ← 0XXXXXXXB, DPTR ← 0000H

INTERNAL DATA MEMORY

00H-07H	RB0	Register Bank 0
08H-0FH	RB1	Register Bank 1
10H-17H	RB2	Register Bank 2
18H-1FH	RB3	Register Bank 3
20H-27H	BAM(00H)-BAM(3FH)	Bit addressable memory
28H-2FH	BAM(40H)-BAM(7FH)	Bit addressable memory
30H-37H		Available to user.
38H-3FH		Available to user.
40H-47H		Available to user.
48H-4FH		Available to user.
50H-57H		Available to user.
58H-5FH		Available to user.
60H-67H		Available to user.
68H-6FH		Available to user.
70H-77H		Available to user.
78H-7FH		Available to user.
80H	P0	

78H-7FH		Available to user.
80H	P0	
81H	SP	
82H	DPL	DPTR
83H	DPH	
84H		
85H		
86H		
87H	PCON	
88H		
89H	TMOD	Timer/Counter Mode Control
8AH	TL0	
8BH	TL1	
8CH	TH0	
8DH	TH1	
98H	SCON	Serial Port control
99H	SBUF	

NOTATION DEFINITIONS

n $\in$ {0,1}, i $\in$ {0,...,7}, bit $\in$ {0,1}, byte $\in$ {0,...,255}, dbyte $\in$ {0,...,255}

short $\in$ {0,...,03FFH}, addr $\in$ {0,...,0FFFFH}

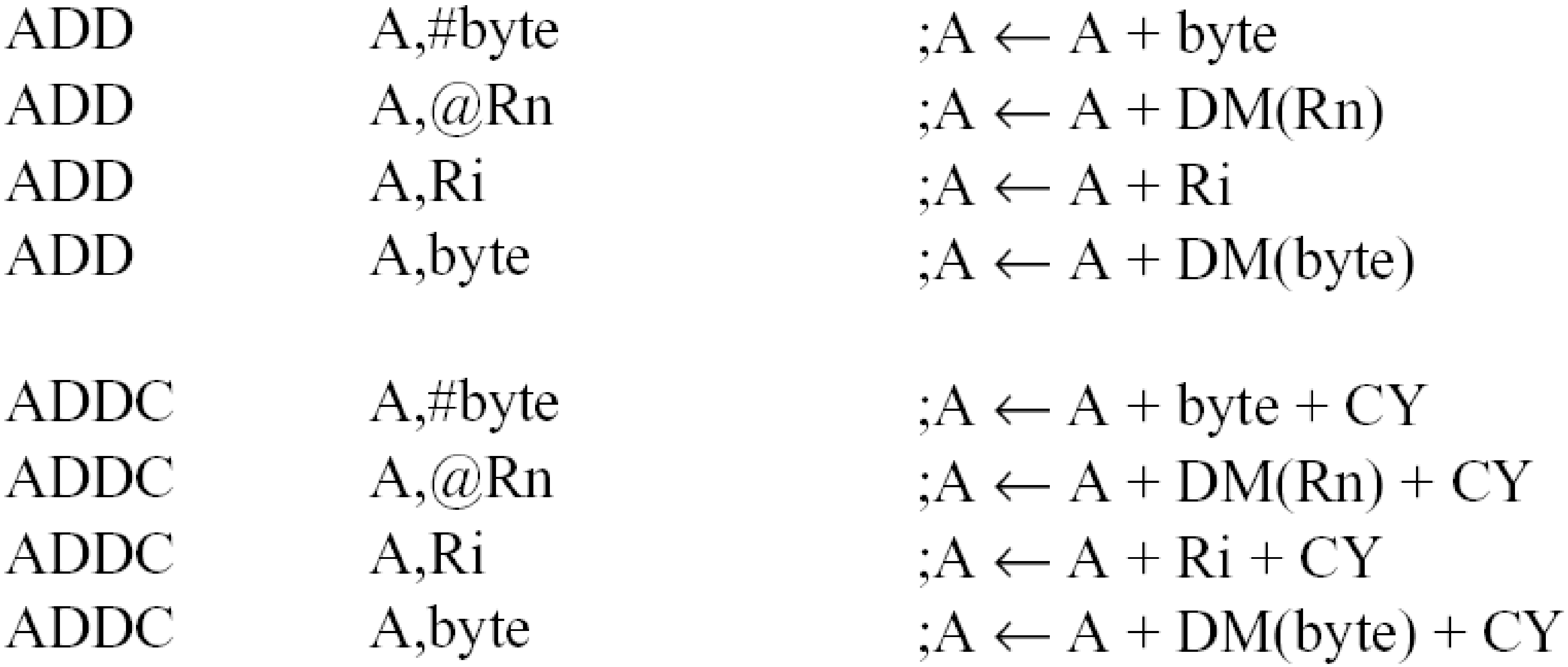
Rn $\in$ {R0, R1}, Ri $\in$ {R0, R1, ..., R7}

Ri = DM(i+8*RBANK)

RBANK = RS1*2+RS0

	CY	AC	FO	RS1	RS0	OV		P	
D0H	D7	D6	D5	D4	D3	D2	D1	D0	**PSW**

ARITHMETIC INSTRUCTIONS



```
ADD     A,#byte     ;A ← A + byte
ADD     A,@Rn       ;A ← A + DM(Rn)
ADD     A,Ri        ;A ← A + Ri
ADD     A,byte      ;A ← A + DM(byte)

ADDC    A,#byte     ;A ← A + byte + CY
ADDC    A,@Rn       ;A ← A + DM(Rn) + CY
ADDC    A,Ri        ;A ← A + Ri + CY
ADDC    A,byte      ;A ← A + DM(byte) + CY
```

Examples:

```
ADD   A,#7FH      ;A ← 7FH

ADD   A,7FH       ;A ← DM(7FH)
```

LOGICAL INSTRUCTIONS

ANL	A,#byte	;$A \leftarrow A \wedge$ byte
ANL	A,@Rn	;$A \leftarrow A \wedge$ DM(Rn)
ANL	A,Ri	;$A \leftarrow A \wedge$ Ri
ANL	A,byte	;$A \leftarrow A \wedge$ DM(byte)
ANL	dbyte,#byte	;DM(dbyte) $\leftarrow$ DM(dbyte) $\wedge$ byte
ANL	byte,A	;DM(byte) $\leftarrow A \wedge$ DM(byte)

CLR	A	;$A \leftarrow 0$
CPL	A	;$A \leftarrow \overline{A}$
RL	A	;$A \leftarrow A(6..0)\|A(7)$
RLC	A	;$CY\|A \leftarrow A\|CY$
RR	A	;$A \leftarrow A(0)\|A(7..1)$
RRC	A	;$A\|CY \leftarrow CY\|A$
SWAP	A	;$A \leftarrow A(3..0)\|A(7..4)$

DATA MOVE INSTRUCTIONS

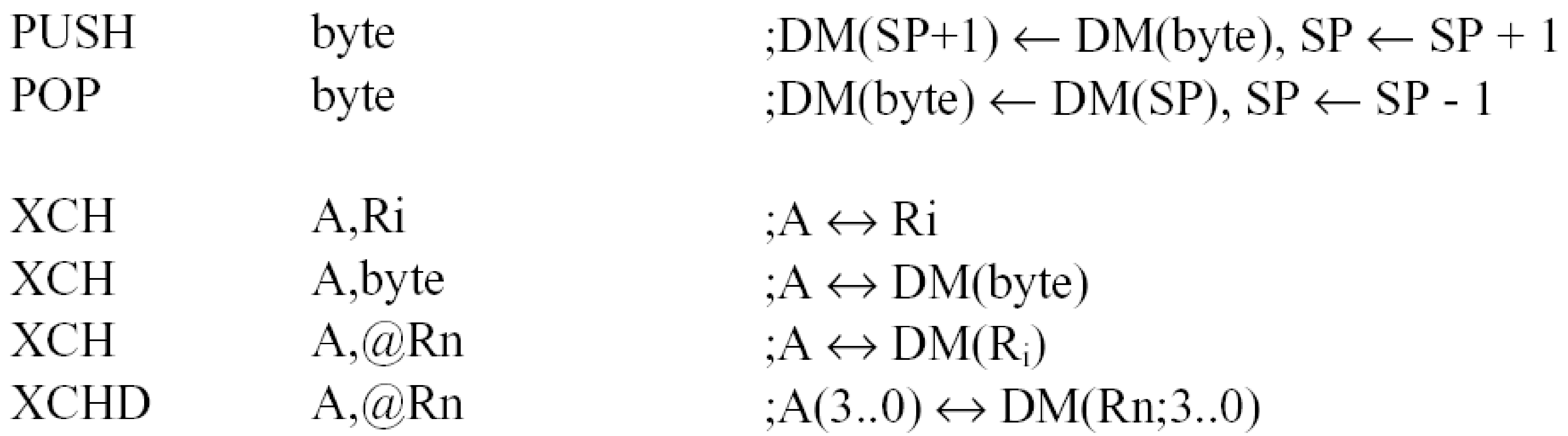
```
MOV        A,#byte          ;A ← byte
MOV        A,@Rn            ;A ← DM(Rn)
MOV        A,Ri             ;A ← Ri
MOV        A,byte           ;A ← DM(byte)
MOV        @Rn,A            ;DM(Rn) ← A
MOV        @Rn,#byte        ;DM(Rn) ← byte
MOV        @Rn,byte         ;DM(Rn) ← DM(byte)
MOV        Ri,A             ;Ri ← A
MOV        Ri,#byte         ;Ri ← byte
MOV        Ri,byte          ;Ri ← DM(byte)
MOV        byte,A           ;DM(byte) ← A
MOV     dbyte,#byte     ;DM(dbyte) ← byte
MOV     byte,@Rn        ;DM(byte) ← DM(Rn)
MOV     byte,Ri         ;DM(byte) ← Ri
MOV     dbyte,byte      ;DM(dbyte) ← DM(byte)
MOV     DPTR,#addr      ;DPTR ← addr

MOVC    A,@A + DPTR     ;A ← CM(A+DPTR)
MOVC    A,@A + PC       ;A ← CM(A+PC)

MOVX    A,@DPTR         ;A ← XM(DPTR)
MOVX    A,@Rn           ;A ← XM(P2|Rn)
MOVX    @DPTR,A         ;XM(DPTR)  ← A
MOVX    @Rn,A           ;XM(P2|Rn) ← A
```

```
MOVE INSTRUCTIONS
   MOV      A,#-1               ;LOAD A WITH 0FFH
   MOV      A,@R1           ;LOAD A WITH DATA MEM POINTED TO BY R1
   MOV      A,R1                ;LOAD A WITH CONTENTS OF R1
   MOV      A,2*30H          ;LOAD A WITH CONTENTS OF DATA MEM 60H
   MOV      A,P2                ;LOAD A WITH CONTENTS OF PORT 2
   MOV      @R0,A            ;STORE A IN DATA MEM POINTED TO BY R0
   MOV      @R1,#0FFH         ;PUT ALL 1'S IN LOC POINTED TO BY R1
   MOV      @R0,7FH             ;READ DATA MEMORY 7FH AND STORE IN
                                ;DATA MEMORY POINTED TO BY R0.
   MOV      R7,A                ;STORE A IN REGISTER R7
   MOV      R6,#01010101B       ;PUT 55H IN R6
   MOV      R0,#50H             ;LOAD R0 WITH 50H = 80
   MOV      R7,#01010001B       ;LOAD R7 WITH 51H = 81
   MOV      R5,55               ;PUT CONTENTS OF LOCATION 55 IN R5
   MOV      55,A                ;PUT A IN DATA MEM LOC 55
   MOV      55,#0AAH            ;PUT AAH IN DM LOC 55 DECIMAL
   MOV      P1,#0FFH            ;SET P1 TO ALL 1'S
   MOV      P2,@R1            ;SET P2 TO DATA MEM POINTED TO BY R1
   MOV      P0,P3               ;READ PORT 3 AND OUTPUT IT TO P0
   MOV      0,P0              ;READ PORT 0 AND SAVE IN DATA MEM 0
   MOV      R2,P2               ;READ P2 AND SAVE IT IN R2
   MOV      DPTR,#A_TABLE       ;POINT DPTR TO LOOK UP TABLE
   MOV      A,#0AH              ;PUT HEX B = 1011 = 11 IN ACC
   MOVC     A,@A+DPTR           ;CONVERT IT TO ASCII (A = 'B')
   CLR      A                   ;LOAD A WITH 0
   MOVC     A,@A+PC           ;PUT OPCODE OF PC+0 (NEXT INST) IN A
```

Push, Pop, and exchange



PUSH	byte	;DM(SP+1) $\leftarrow$ DM(byte), SP $\leftarrow$ SP + 1
POP	byte	;DM(byte) $\leftarrow$ DM(SP), SP $\leftarrow$ SP - 1
XCH	A,Ri	;A $\leftrightarrow$ Ri
XCH	A,byte	;A $\leftrightarrow$ DM(byte)
XCH	A,@Rn	;A $\leftrightarrow$ DM(R_i)
XCHD	A,@Rn	;A(3..0) $\leftrightarrow$ DM(Rn;3..0)

PROGRAM AND MACHINE CONTROL

CALL	**Note: Assembler translates CALL to ACALL or LCALL**	
ACALL	short	;DM(SP+2)|DM(SP+1) ← PC+2,
		;PC(10..0) ← short
LCALL	addr	;DM(SP+2)|DM(SP+1) ← PC+2,
		;PC(10..0) ← addr
RET		;PC ← DM(SP)|DM(SP-1), SP ← SP-2
RETI		;PC ← DM(SP)|DM(SP-1), SP ← SP-2
		;Reenable equal or lower priority INT
JMP	**Note: JMP is translated to AJMP, LJMP, or SJMP.**	
AJMP	short	;PC(10..0) ← short
LJMP	addr	;PC(15..0) ← addr
SJMP	byte	;PC ← PC + 2 + byte(7)..byte(7)|byte
JMP	@A+DPTR	;PC ← DPTR + A
JZ	byte	;IF A = 0 THEN
		;PC ← PC + 2 + byte(7)..byte(7)|byte
		;ELSE PC ← PC+2
JNZ	byte	;IF A ≠ 0 THEN
		;PC ← PC + 2 + byte(7)..byte(7)|byte
		;ELSE PC ← PC+2

```
CJNE        A,dbyte,byte          ;IF A ≠ DM(dbyte) THEN
                                  ;PC ← PC + 2 + byte(7)..byte(7)|byte
                                  ;IF A < DM(dbyte) THEN CY ← 1
                                  ;ELSE CY ← 0
CJNE        A,#dbyte,byte         ;IF A ≠ dbyte THEN
                                  ;PC ← PC + 2 + byte(7)..byte(7)|byte
                                  ;IF A < dbyte THEN CY ← 1
                                  ;ELSE CY ← 0
CJNE        Rn,#dbyte,byte        ;IF Rn ≠ dbyte THEN
                                  ;PC ← PC + 2 + byte(7)..byte(7)|byte
                                  ;IF A < DM(dbyte) THEN CY ← 1
                                  ;ELSE CY ← 0
CJNE        @Rn,#dbyte,byte       ;IF DM(Rn) ≠ dbyte THEN
                                  ;PC ← PC + 2 + byte(7)..byte(7)|byte
                                  ;IF A < dbyte THEN CY ← 1
                                  ;ELSE CY ← 0
```

```
DJNZ      Rn,byte            ;Rn ← Rn-1, IF (Rn-1) ≠ 0 THEN
                             ;PC ← PC + 2 + byte(7)..byte(7)|byte
                             ;ELSE PC ← PC + 2
DJNZ      dbyte,byte         ;DM(dbyte) ← DM(dbyte)-1,
                             ;IF (DM(dbyte)-1) ≠ 0 THEN
                             ;PC ← PC + 3 + byte(7)..byte(7)|byte
                             ;ELSE PC ← PC + 3
NOP                          ;PC ← PC + 1
```

BIT MANIPULATION INSTRUCTIONS

CLR	C	;CY $\leftarrow$ 0
CLR	byte	;BADM(byte) $\leftarrow$ 0
SETB	C	;CY $\leftarrow$ 1
SETB	byte	;BADM(byte) $\leftarrow$ 1
CPL	C	;CY $\leftarrow \overline{C}\overline{Y}$
CPL	byte	;BADM(byte) $\leftarrow \overline{\text{BADM(byte)}}$
ANL	C,byte	;CY $\leftarrow$ CY $\wedge$ BADM(byte)
ANL	C,/byte	;CY $\leftarrow$ CY $\wedge \overline{\text{BADM(byte)}}$
ANL	byte,bit	;BADM(byte) $\leftarrow$ BADM(byte) $\wedge$ bit
ORL	C,byte	;CY $\leftarrow$ CY $\vee$ BADM(byte)
ORL	C,/byte	;CY $\leftarrow$ CY $\vee \overline{\text{BADM(byte)}}$
ORL	byte,bit	;BADM(byte) $\leftarrow$ BADM(byte) $\vee$ bit
MOV	C,byte	;CY $\leftarrow$ BADM(byte)
MOV	byte,C	;BADM(byte) $\leftarrow$ CY

BIT JUMP INSTRUCTIONS

```
JB      dbyte,byte      ;IF BADM(dbyte) = 1 THEN
                        ;PC ← PC + 3 + byte(7)..byte(7)|byte
                        ;ELSE PC ← PC+3
JNB     dbyte,byte      ;IF BADM(dbyte) = 0 THEN
                        ;PC ← PC + 3 + byte(7)..byte(7)|byte
                        ;ELSE PC ← PC+3
JBC     dbyte,byte      ;IF BADM(dbyte) = 1 THEN
                        ;    BADM(dbyte) ← 0
                        ;    PC ← PC + 3 + byte(7)..byte(7)|byt
                        ;ELSE PC ← PC+3
JC      byte            ;IF CY = 1 THEN
                        ;PC ← PC + 3 + byte(7)..byte(7)|byte
                        ;ELSE PC ← PC+3
JNC     byte            ;IF CY = 0 THEN
                        ;PC ← PC + 3 + byte(7)..byte(7)|byte
                        ;ELSE PC ← PC+3
```

INSTRUCTIONS THAT AFFECT FLAGS (incomplete)

Instruction	CY	OV	AC	Instruction	CY	OV	AC
ADD	X	X	X	CLR C	0	-	-
ADDC	X	X	X	CPL C	X	-	-
SUBB	X	X	X	ANL C,bit	X	-	-
MUL	0	X	-	ANL C,/bit	X	-	-
DIV	0	X	-	ORL C,bit	X	-	-
DA	X	-	-	ORL C,bit	X	-	-
RRC	X	-	-	MOV C,bit	X	-	-
RLC	X	-	-	CJNE	X	-	-
SETB C	1	-	-				