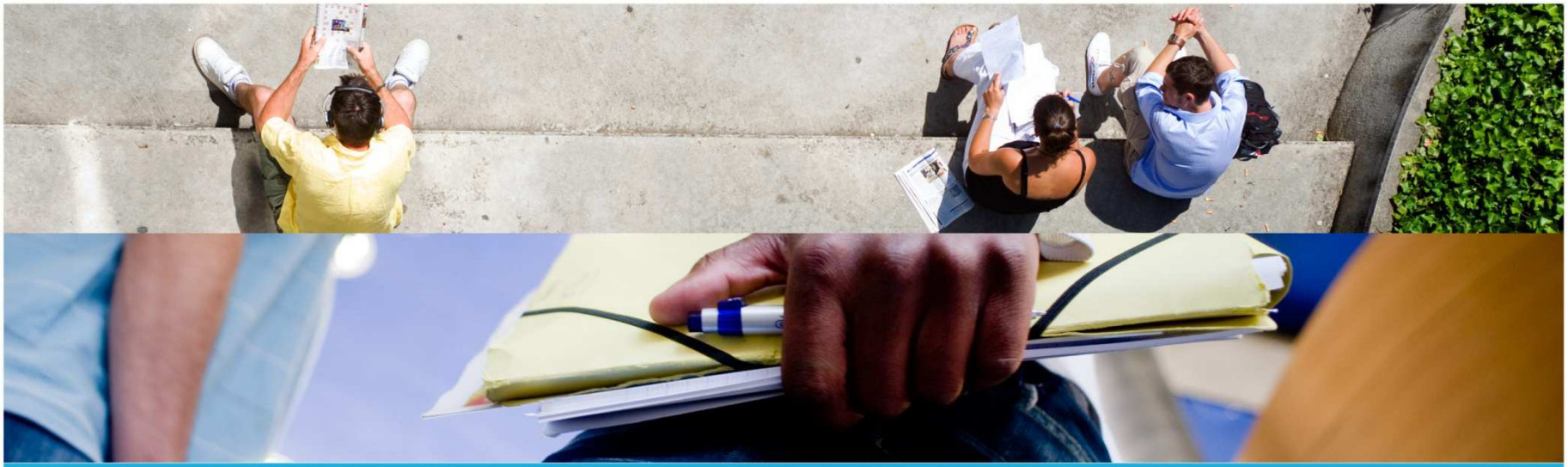


Architecture et programmation du 8051et 8051F020

Séance 6

N.ABOUCHI

membre de UNIVERSITÉ DE LYON

CPE LYON
ÉCOLE SUPÉRIEURE DE CHIMIE PHYSIQUE ÉLECTRONIQUE DE LYON

Objectifs du cours

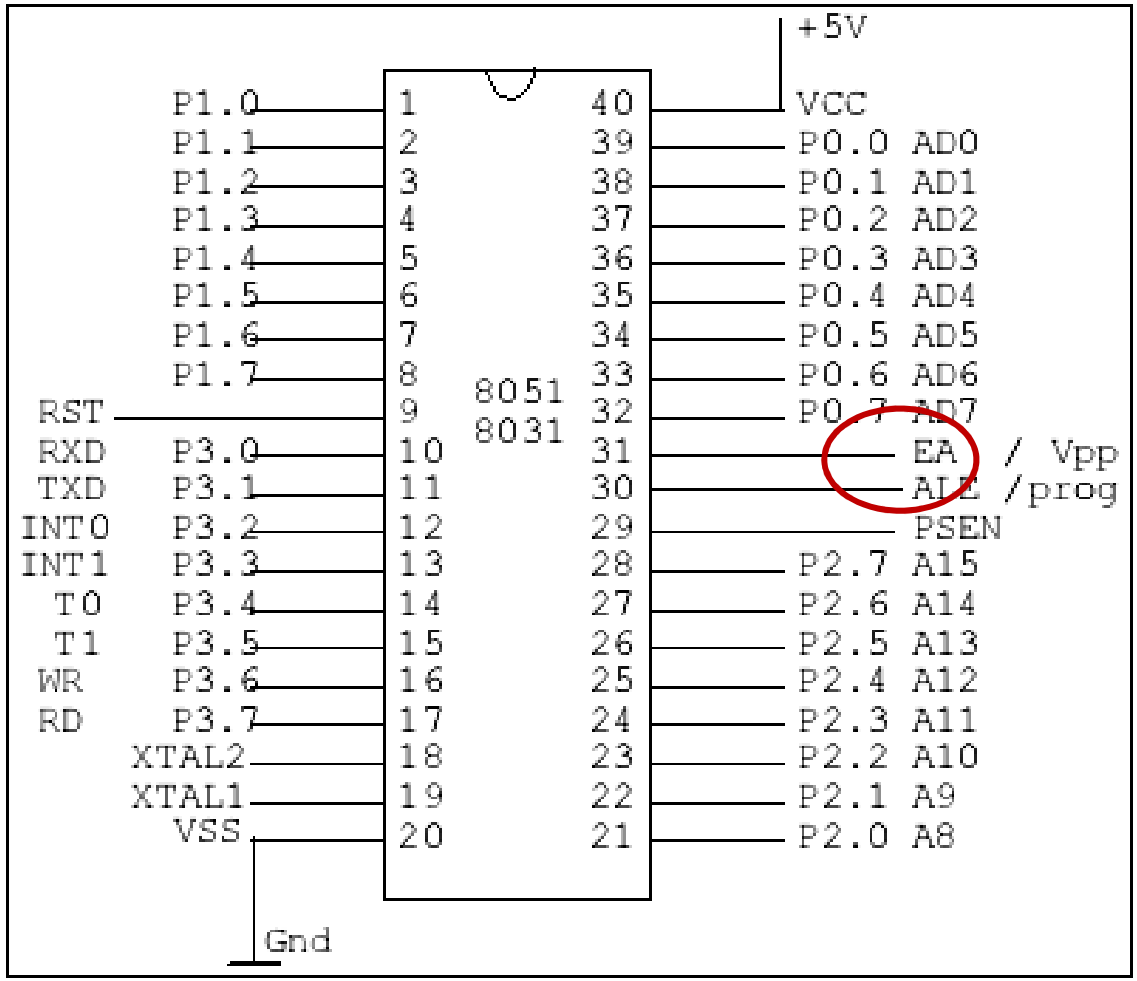
Architecture du 8051 / 8051F020

Organisation de la mémoire

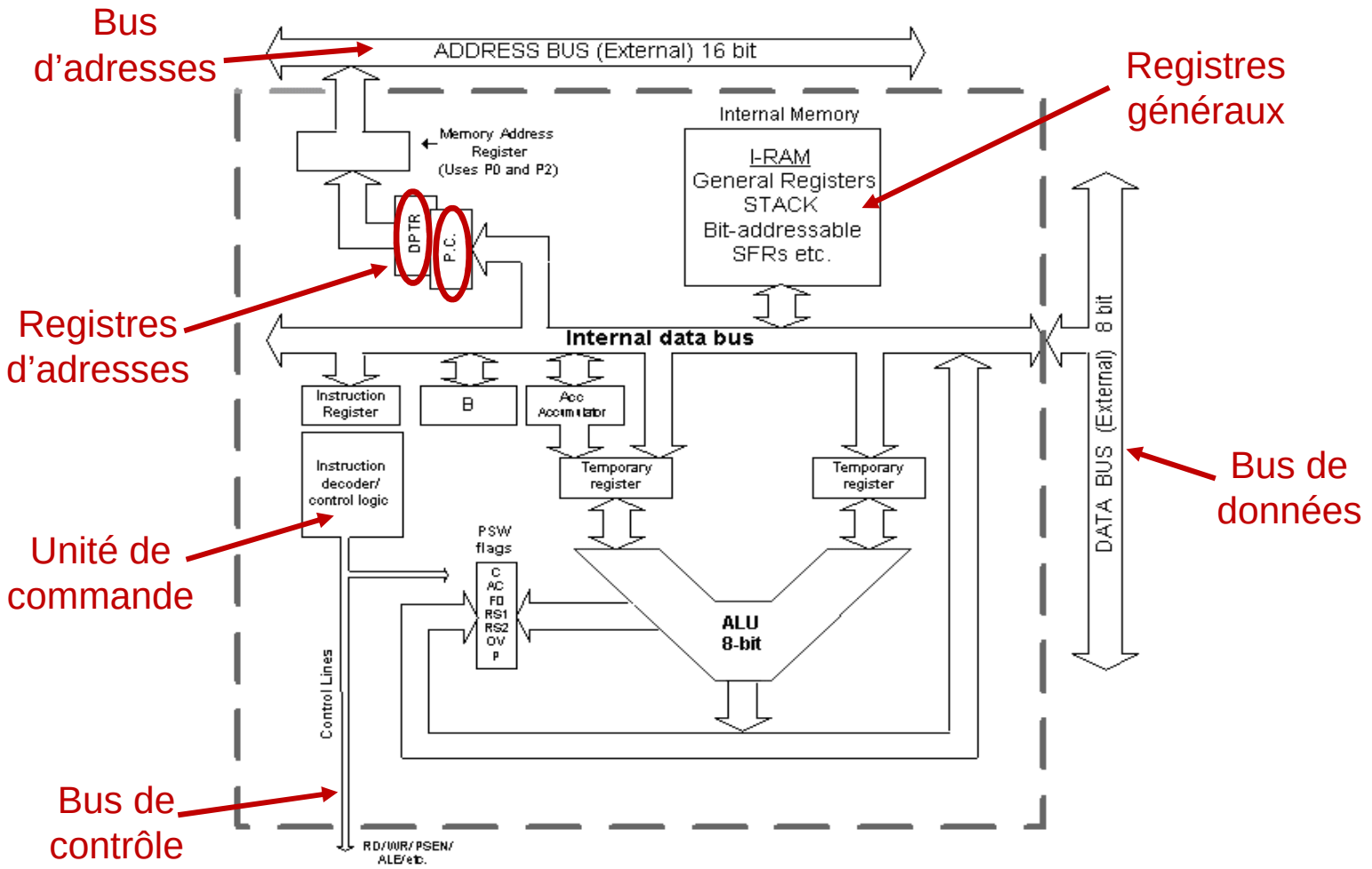
Modes d’adressages

Programmation assembleur

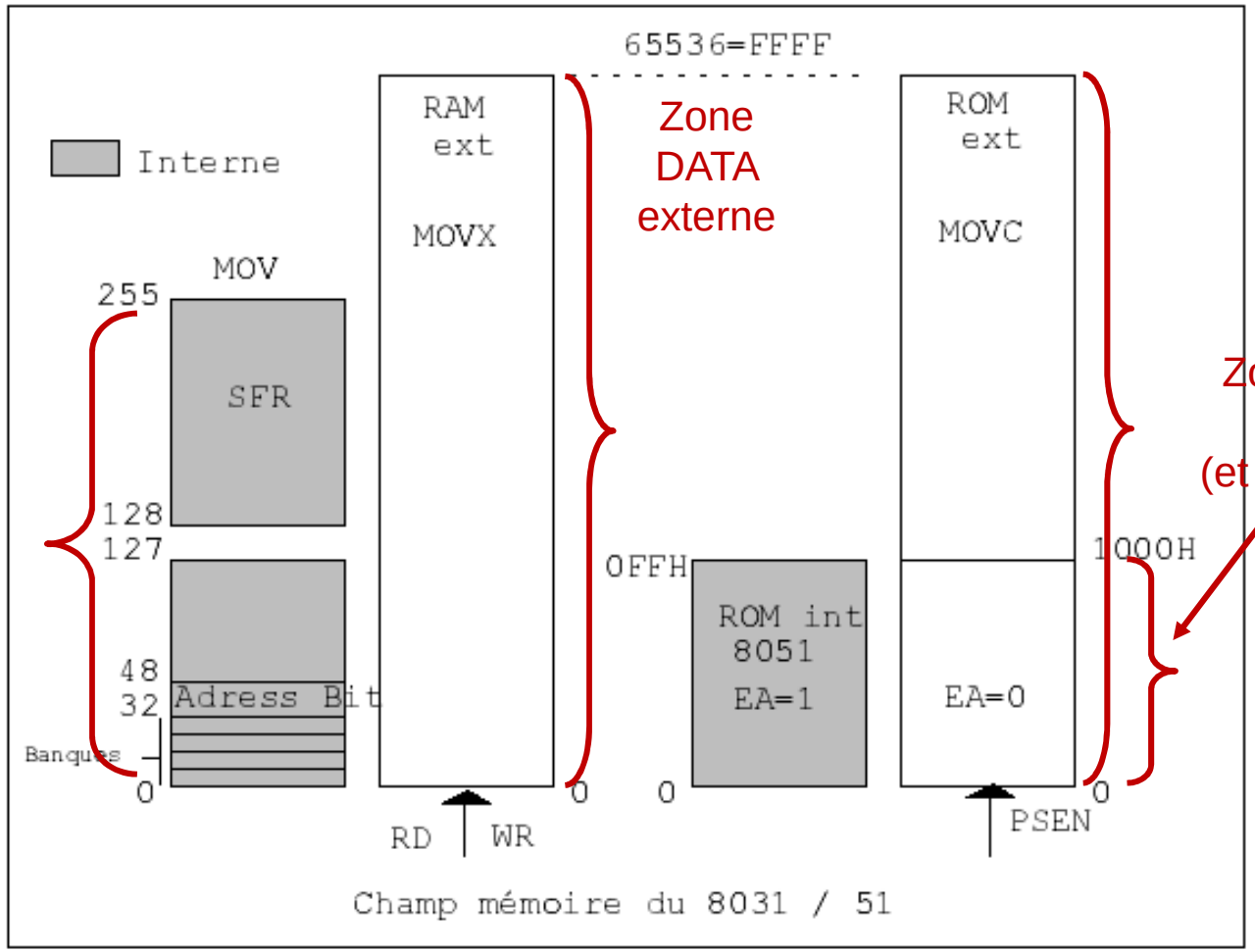
Le 8051 : architecture externe



Le 8051 : Architecture interne (architecture du type Harvard)



Organisation de la mémoire
(architecture du type Harvard : code et données séparées)



Zone DATA interne

Zone CODE externe (et /ou interne)

Zone CODE externe

D'une taille maximale de 64Ko, c'est l'espace affecté aux mémoires non volatiles telles que ROM, EPROM, FLASH EPROM.

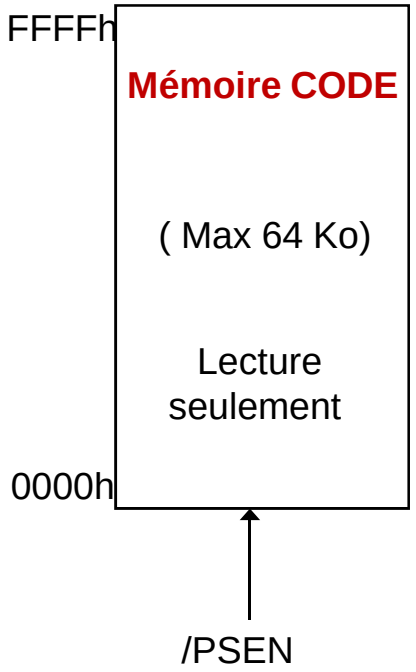
Cet espace sert à stocker le code exécutable et les constantes.

Il ne peut être lu, qu'avec l'instruction **MOVC** en adressage indexé.

```
MOVC A,   @A+DPTR
MOVC A,   @A+PC
```

Quand il est placé à l'extérieur du 8051, il est décodé avec le signal /PSEN.

La quasi-totalité des composants de la famille 8051 actuel intègrent de la mémoire CODE en interne.



MOVC
Adressage indexé

Zone DATA externe

D'une taille maximale de 64Ko, c'est l'espace affecté aux mémoires volatiles de type SRAM.

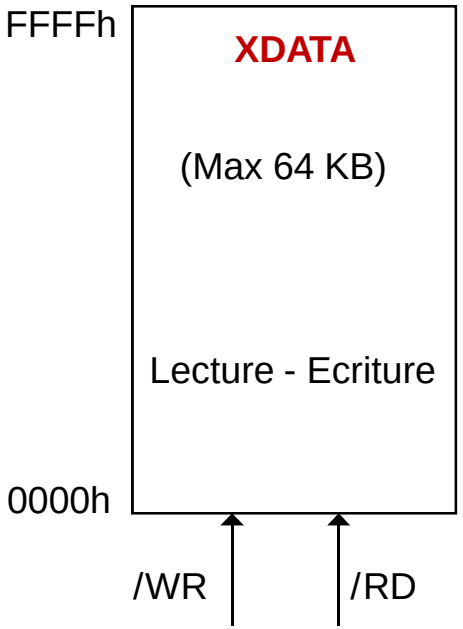
Cet espace sert à stocker les données.

Il peut être lu et écrit, avec l'instruction MOVX en adressage indirect.

```
MOVX A,   @DPTR
MOVX @DPTR, A
```

A cause de ce type d'adressage, le temps d'accès est bien supérieur au temps d'accès à la zone DATA interne.

Quand il est placé à l'extérieur du 8051, il est décodé avec les signaux /RD et /WR.



MOVX
Adressage indirect

Zone DATA interne

Cet espace mémoire, limité à 256 octets, est toujours interne au 8051.

Il peut être lu, avec l'instruction MOV.

MOV A,R0

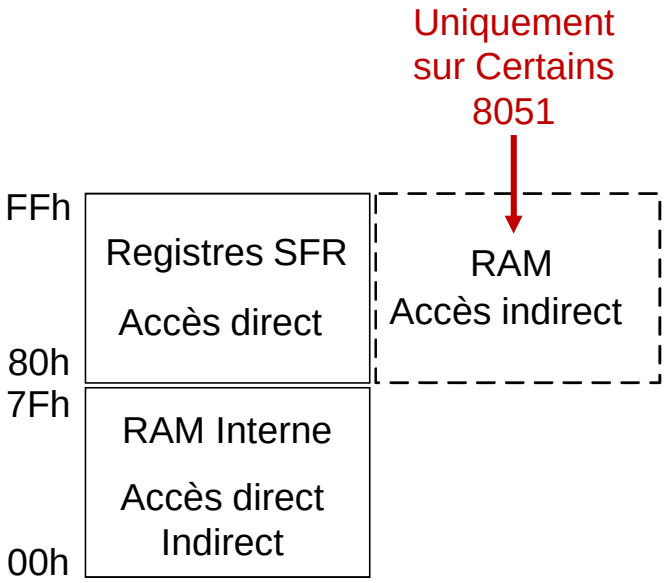
- 00h à 1Fh : 4 Bancs de 8 registres.

- 20h à 2Fh : RAM adressable bit à bit.

- 30h à 7Fh : RAM d'usage général.

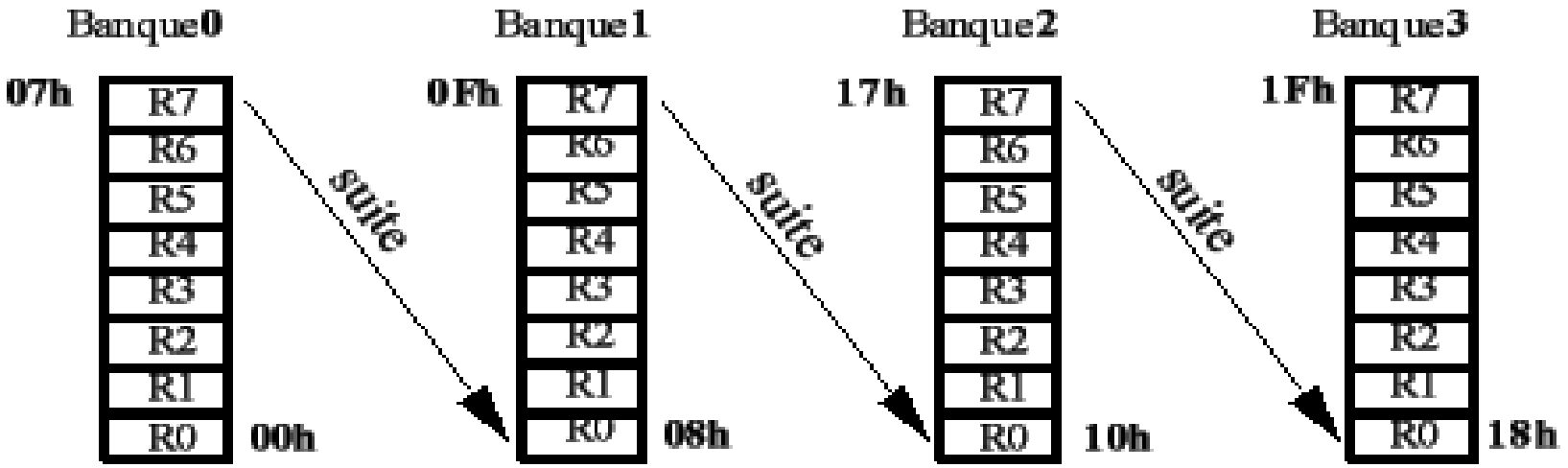
- 80H à FFh : zone SFR

Sur certain 8051, une RAM en adressage indirect est accessible de 80h à FFh.



MOV
Adressage direct
Adressage indirect

Zone data interne de 00h à 1Fh (4 Bancs de registres)



Le choix de la banque se fait par les valeurs des bits **RS0** et **RS1** du registre d'état : **PSW**

0 0 : banque 0
0 1 : banque 1
1 0 : banque 2
1 1 : banque 2

Zone data interne de 80h à FFh : SFR

ACC : registre accumulateur

PSW : registre d'état

F8	F9	FA	FB	FC	FD	FE	FF
F0 (B)							F7
E8							EF
E0(ACC)							E7
D8							DF
D0 (PSW)							D7
C8				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)	81 (SP)	DPL	DPH				87 (PCON)

Registre SFR adressable bit à bit

SP : adressage de la pile

DPTR : adressage des données

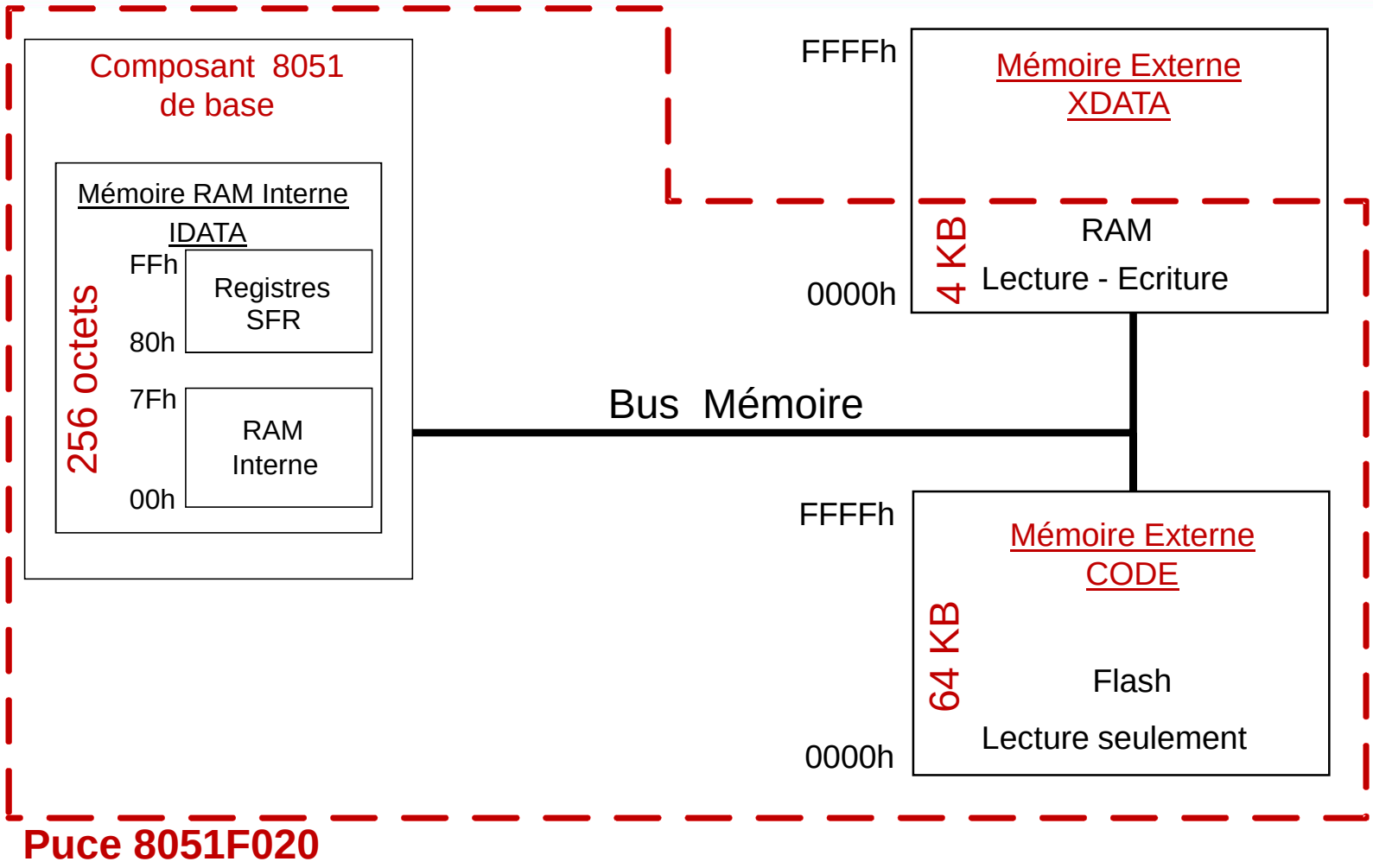
Zone data interne de 20h à 7Fh :
RAM adressable par bit à bit et RAM d'usage général

Adressable par octet

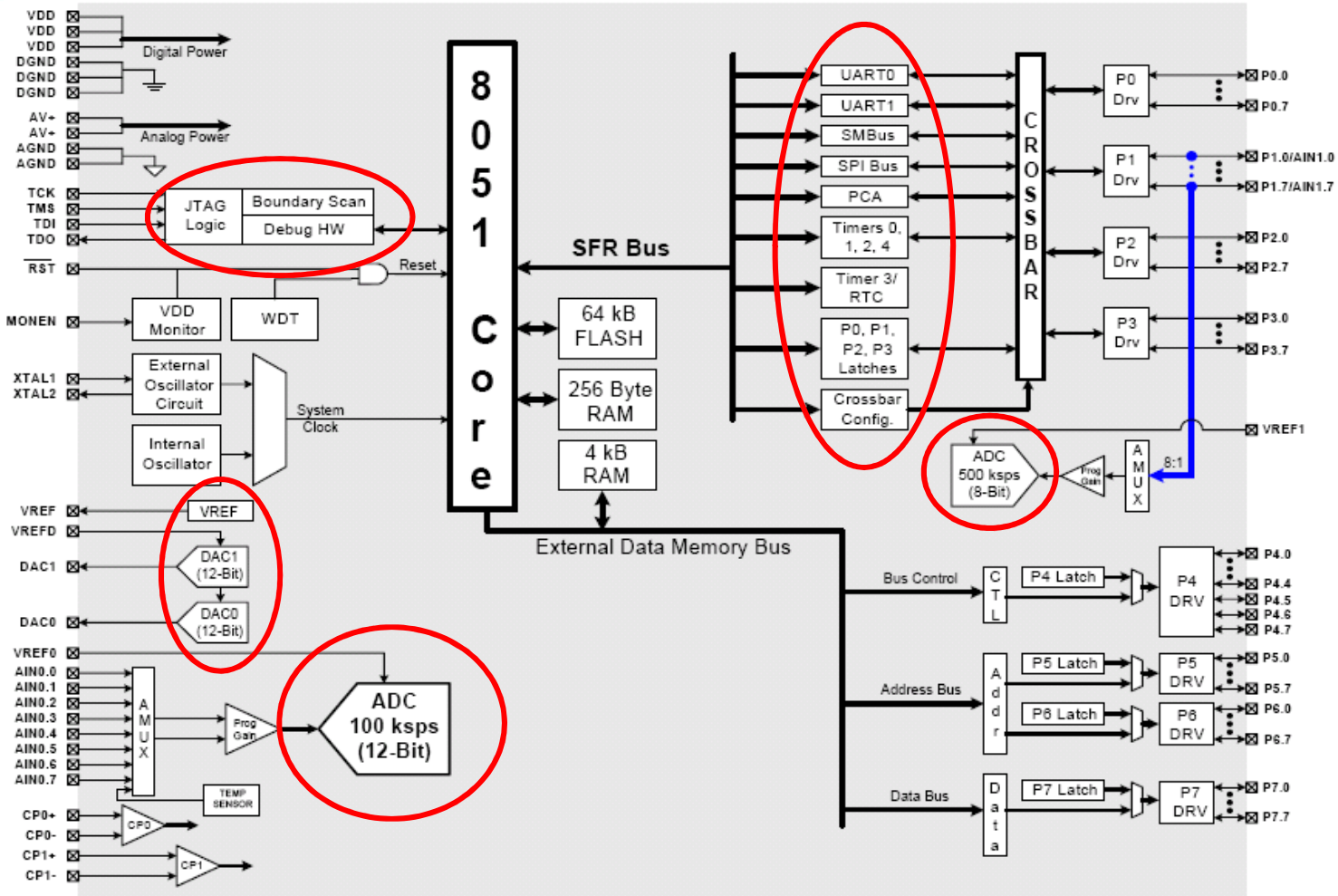
30h à 7Fh	RAM d'usage général	
2Fh	7F	78
2Eh	77	70
2Dh	6F	68
2Ch	67	60
2Bh	5F	58
2Ah	57	50
29h	4F	48
28h	47	40
27h	3F	38
26h	37	30
25h	2F	28
24h	27	20
23h	1F	18
22h	17	10
21h	0F	08
20h	07	00
	b7 b6 b5 b4 b3 b2 b1 b0	

Adressable Bit à bit

Le cas du 8051F020



Cas du 8051F020 : nouveaux périphériques



Cas du 8051F020 : nouvelle zone data interne de 80h à FFH : SFR

F8	SPI0CN	PCA0H	PCA0CPH0	PCA0CPH1	PCA0CPH2	PCA0CPH3	PCA0CPH4	WDTCN
F0	B	SCON1	SBUF1	SADDR1	TL4	TH4	EIP1	EIP2
E8	ADC0CN	PCA0L	PCA0CPL0	PCA0CPL1	PCA0CPL2	PCA0CPL3	PCA0CPL4	RSTSRC
E0	ACC	XBR0	XBR1	XBR2	RCAP4L	RCAP4H	EIE1	EIE2
D8	PCA0CN	PCA0MD	PCA0M0	PCA0CPM1	PCA0CPM2	PCA0CPM3	PCA0CPM4	
D0	PSW	REF0CN	DAC0L	DAC0H	DAC0CN	DAC1L	DAC1H	DAC1CN
C8	T2CON	T4CON	RCAP2L	RCAP2H	TL2	TH2		SMB0CR
C0	SMB0CN	SMB0STA	SMB0DAT	SMB0ADR	ADC0GTL	ADC0GTH	ADC0LTL	ADC0LTH
B8	IP	SADEN0	AMX0CF	AMX0SL	ADC0CF	P1MDIN	ADC0L	ADC0H
B0	P3	OSCXCN	OSCICN			P74OUT	FLSCL	FLACL
A8	IE	SADDR0	ADC1CN	ADC1CF	AMX1SL	P3IF	SADEN1	EMI0CN
A0	P2	EMI0TC		EMI0CF	P0MDOUT	P1MDOUT	P2MDOUT	P3MDOUT
98	SCON0	SBUF0	SPI0CFG	SPIODAT	ADC1	SPI0CKR	CPT0CN	CPT1CN
90	P1	TMR3CN	TMR3RLL	TMR3RLH	TMR3L	TMR3H	P7	
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PSCTL
80	P0	SP	DPL	DPH	P4	P5	P6	PCON

Les modes d'adressages mémoire

Adressage Immédiat

MOV A,#40H
MOV A,#11001101B
MOV P1,#80H

Toute case mémoire interne désignée par son adresse peut être chargée

MOV 40H,#5AH

Adressage direct : **Il n'existe pas d'adressage direct dans la RAM extérieure**.

MOV A, 45H
MOV R0, 30
MOV A, P1

Adressage Indirect (R0 ou R1)

MOV @R1, P1
MOVX @DPTR, A

Adressage relatif : réservé pour les instructions de rupture de séquence conditionnel.

Adressage étendu : permet d'effectué des ruptures de séquence sans condition afin d'atteindre une adresse non successive dans la mémoire programme.

Les modes d'adressages mémoire

Exemples

MOV R0, 40H

MOV R0, #40H

MOV @R0, 40H

MOV 00, 40H

Programmation assembleur

Le terme «assembleur », peut avoir deux significations :

- Désigne les instructions (langage) de de programmation d'un microprocesseur, exprimé sous forme de mnémoniques (MOV, ADD, JMP, etc.) et d'opérandes.
- Désigne le programme informatique qui transforme les mnémoniques en code binaires, compréhensibles par le processeur.

D'une façon générale, on appelle « assembleur » tout outil destiné à faciliter l'écriture de programmes en langage machine. Il permet la transcription de codes symboliques en codes exécutables par le processeur.

Code symbolique	→	Assemblage	→	Code exécutable
MOV DPTR,#01234h				90 12 34

l'opération d'assemblage est toujours complétée par l'édition de liens. Cette tâche, accomplie par l'éditeur de liens (Linker) va rassembler plusieurs programmes assemblés séparément, en un seul fichier.

Eléments d’un programme assembleur

Les principaux éléments d’un programme assembleur sont les instructions machines, exprimées sous la forme de mnémoniques et d’opérandes (MOV, ADD ,ORL, JMP, etc.).

Les opérandes sont les valeurs, les registres, les adresses mémoire manipulées par les instructions. Ces opérandes peuvent aussi être exprimés sous forme symbolique pour faciliter la tâche du programmeur.

Des directives et paramètres d’assemblage peuvent être introduite pour contrôler le fonctionnement de l’assembleur. Ils ne produisent pas de codes exécutables sauf quand on fait de l’assemblage conditionnel (au même titre que les commentaires), mais permettent de donner des informations à l’assembleur pour produire un code optimisé.

Structure d'un programme assembleur

Directives d'assemblage : informent l'assembleur comment traiter certaines instructions. En outre, elles permettent la définition des constantes, et la réservation d'espace mémoire pour les variables.

Paramètres d'assemblage : contrôlent les opérations d'assemblage, notamment sur la forme des fichiers LST (fichier listing) et objet.

Instructions : sous la forme suivante :

[label :] mnémonique [opérande] [,opérande] [,opérande] [;commentaire]

Exemple de code :

```
$TITLE (code_example)          ; Paramètre
    CSEG at 0000h              ; Directive
    JMP 0200h                  ; Instruction
END                            ; Directive
```

8051 : Instruction arithmétique

Mnémonic		Opération	Mode adressage				Exécution Temps(uS)
			Dir	Ind	Reg	Imm	
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>	X	X	X	X	1
INC	A	A = A + 1	Accumulateur				1
INC	<byte>	<byte> = <byte> + 1	X	X	X		1
INC	DPTR	DPTR = DPTR + 1	Data Pointer				2
DEC	A	A = A - 1	Accumulateur				1
DEC	<byte>	<byte> = <byte> - 1	X	X	X		1
MUL	AB	B et A = B*A	Acc et B				4
DIV	AB	A = Ent(A/B) B = Res(A/B)	Acc et B				4
DA	A	Decimal Adjust	Accumulateur				1

8051 : Instruction logique

Mnémonic		Opération	Mode adressage				Exécution
			Dir	Ind	Reg	Imm	Temps(uS)
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	Accumulateur				1
CPL	A	A = NOT.A	Accumulateur				1
RL	A	Rotate Acc Left 1 bit	Accumulateur				1
RLC	A	Rotate Left through Carry	Accumulateur				1
RR	A	Rotate Acc Right 1 bit	Accumulateur				1
RRC	A	Rotate Right through Carry	Accumulateur				1
SWAP	A	A(3..0) échange A(7..4)	Accumulateur				1

8051 : Instruction de transfert

RAM interne

Mnémonic		Opération		Mode Adressage				Exécution Temps(uS)
				Dir	Ind	Reg	Imm	
MOV	A,<src>	A = <src>		X	X	X	X	1
MOV	<dest>,A	<dest> = A		X	X	X		1
MOV	<dest>,<src>	<dest> = <src>	<dest>	<src>				
			direct	X	X	X	X	2
			registre	X			X	2
			indexé	X			X	2
MOV	DPTR,#data16	DPTR = constant Imm(16)					X	2
PUSH	<src>	INC SP : MOV <@SP>,<src>		X				2
POP	<dest>	MOV <dest>,<@SP> : DEC SP		X				2
XCH	A,<byte>	Acc et <byte> échange data		X	X	X		1
XCHD	A,@Ri	A(3..0) échange @Ri(3..0)		X				1

RAM externe

Adresses 8/16	Mnémonic	Opération	Temps(uS)
8 bits	MOVX A,@Ri	Lecture Ram	2
8 bits	MOVX @Ri,A	Ecriture Ram	2
16 bits	MOVX A,@DPTR	Lecture Ram	2
16 bits	MOVX @DPTR,A	Ecriture Ram	2

8051 : Instructions travaillant sur un bit

Mnémonic		Opération	Temps(uS)
ANL	C,bit	C = C.AND.bit	2
ANL	C,/bit	C = C.AND./bit	2
ORL	C,bit	C = C.OR.bit	2
ORL	C,/bit	C = C.OR./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 = /C	1
CPL	bit	bit = /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 et clr bit	2

8051 : Instructions de lecture en mémoire programme

Mnémonic	Opération	Temps(uS)
MOVC A,@A+DPTR	Lecture PGM à @(A+DPTR)	2
MOVC A,@A+PC	Lecture PGM à @(A+PC)	2

8051 : instructions de branchements

Branchement inconditionnels

Mnémonic		Opération	Temps(uS)
JMP	Adr	Jump to addr	2
JMP	@A + DPTR	Jump to A + DPTR	2
CALL	Adr	Saut subroutine	2
RET		Retour subroutine	2
RETI		Retour subroutine INT	2
NOP		Pas d'opération	1

Branchement conditionnels

Mnémonic		Opération	Mode Adressage				Exécution Temps(uS)
			Dir	Ind	Reg	Imm	
JZ	rel	Jump if A = 0	Accumulateur				2
JNZ	rel	Jump if A ≠ 0	Accumulateur				2
DJNZ	<byte>,rel	Dec and jump if not 0	X		X		2
CJNE	A,<byte>,rel	Jump if A =/ <byte>	X			X	2
CJNE	<byte>,#data,rel	Jmp if <byte> ≠ #data		X	X		2

Exemple d'un code 8051

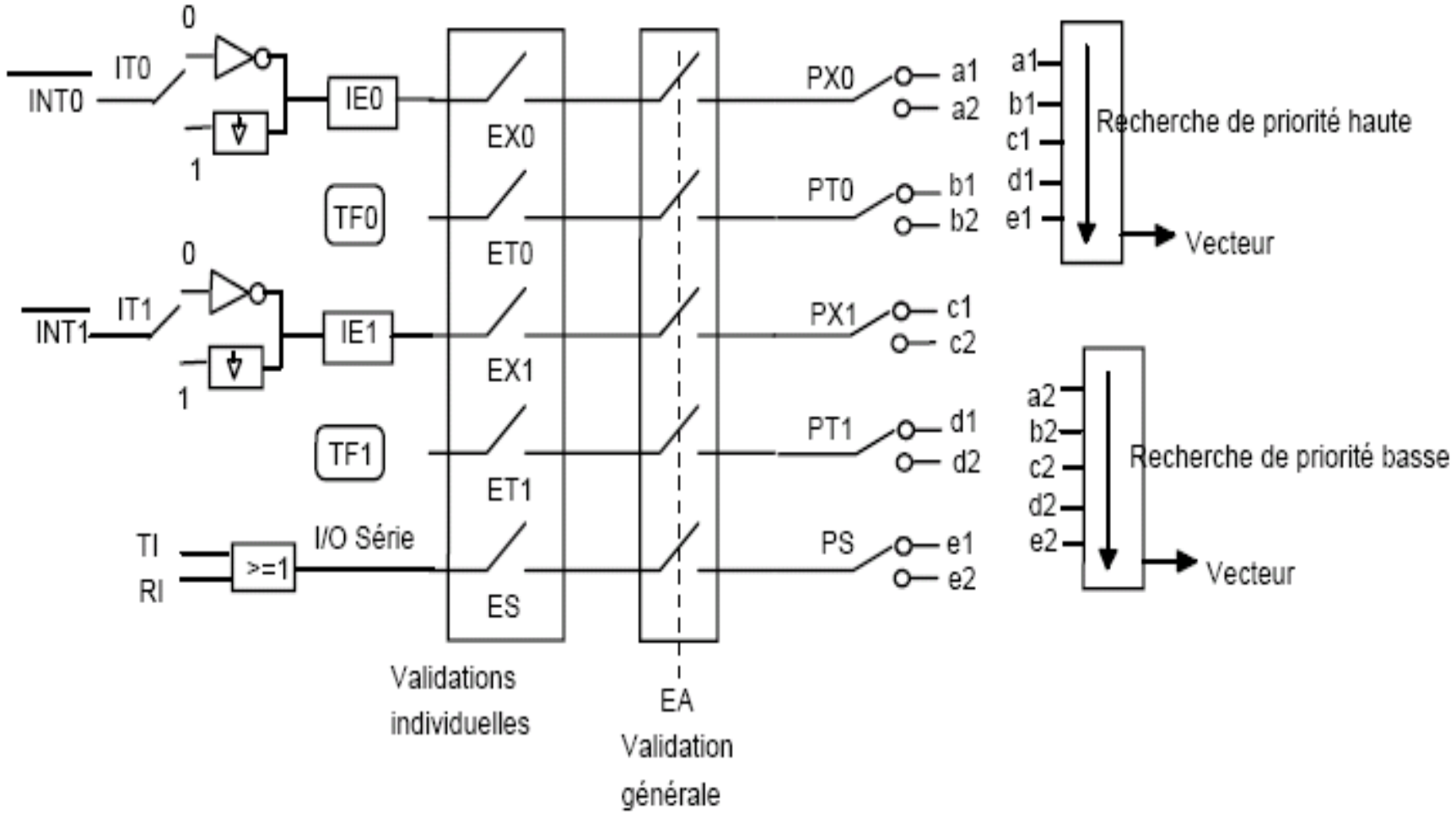
```
ORG     0000h    ; Prochaine instruction à l'adresse 0000h
LJMP    DEBUT
ORG     0003h    ; Prochaine instruction à l'adresse 0003h
RETI
ORG     000Bh    ; Prochaine instruction à l'adresse 000Bh
RETI
ORG     0013h    ; Prochaine instruction à l'adresse 0013h
LJMP    SPI1     ; saut à l'adresse du sous programme d'interruption
                  (demande sur INT1)
RETI
ORG     001Bh    ; Prochaine instruction à l'adresse 001Bh
RETI
ORG     0023h    ; Prochaine instruction à l'adresse 0023h
RETI
```

Exemple d'un code 8051

```
ORG     0033h           ; début du programme principal (à l'adresse 0033)
DEBUT : CLR PX1         ; priorité de l'interruption INT1 à 0
        CLR IE1         ; interruption sur niveau
        SETB EX1        ; autorise la demande sur l'interruption INT1
        SETB EA         ; autorise toutes les demandes d'interruptions
SUITE:  SJMP SUITE      ; boucle infinie pour simuler la suite du programme
ici commence le sous programme d'interruption 1
SPI1 :  PUSH PSW        ; sauvegarde de PSW
        PUSH ACC        ; sauvegarde de ACC
        NOP             ; le sous programme d'interruption ne fait rien
        POP ACC         ; restitution de ACC
        POP PSW         ; restitution de PSW
        RETI            ; retour au programme principal
```

Mise en œuvre des interruptions

Cas du 8051 de base
(5 sources d'interruptions vectorisées)



Les différents registres nécessaires à la mise en œuvre des interruptions

F8	F9	FA	FB	FC	FD	FE	FF
F0 (B)							F7
E8							EF
E0(ACC)							E7
D8							DF
D0 (PSW)							D7
C8				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)	81 (SP)	DPL	DPH				87 (PCON)

Registres à initialiser

Contrôles des interruptions

Registre de validation : TCON

Adresse bit	8F	8E	8D	8C	8B	8A	89	88
	TF1	TR1	TF0	TR0	**IE1**	**IT1**	**IE0**	**IT0**

TF0/1 : Débordement timer 0/1
TR0/1 : Contrôle interne du timer 0/1
IE0/1 : Détection d'interruption externe 0/1
IT0/1 : Contrôle de déclenchement de IT0/1

Registre de validation : IE

EA	-	ET2	ES	ET1	EX1	ET0	EX0

Registre de validation : IP

-	-	PT2	PS	PT1	PX1	PT0	PX0

En cas de conflit : du plus haut au plus bas : IE0, TF0, IE1, TF1, RI ou TI

Table des vecteurs d'interruptions (cas du 8051)

RESET :	**Saut à l'adresse**	**0000h de la mémoire**
INT0:	**Saut à l'adresse**	**0003h de la mémoire**
TIMER0 :	**saut à l'adresse**	**000Bh de la mémoire**
INT1:	**Saut à l'adresse**	**0013h de la mémoire**
TIMER1 :	**Saut à l'adresse**	**001Bh de la mémoire**
TI ou RI :	**Saut à l'adresse**	**0023h de la mémoire**

Table des vecteurs d'interruption (cas du 8051*F020*)

Table 12.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	Always Enabled	Always Highest
External Interrupt 0 (/INT0)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1 (/INT1)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3)	PT1 (IP.3)
UART0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y		ES0 (IE.4)	PS0 (IP.4)

Table des vecteurs d'interruption cas du 8051F020 : 23 sources d'interruptions

Table 12.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	Always Enabled	Always Highest
External Interrupt 0 (/INT0)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1 (/INT1)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3)	PT1 (IP.3)
UART0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y		ES0 (IE.4)	PS0 (IP.4)
Timer 2 Overflow (or EXF2)	0x002B	5	TF2 (T2CON.7)	Y		ET2 (IE.5)	PT2 (IP.5)
Serial Peripheral Interface	0x0033	6	SPIF (SPI0CN.7)	Y		ESPI0 (EIE1.0)	PSPI0 (EIP1.0)
SMBus Interface	0x003B	7	SI (SMB0CN.3)	Y		ESMB0 (EIE1.1)	PSMB0 (EIP1.1)
ADC0 Window Comparator	0x0043	8	AD0WINT (ADC0CN.2)	Y		EWADC0 (EIE1.2)	PWADC0 (EIP1.2)
Programmable Counter Array	0x004B	9	CF (PCA0CN.7) CCFn (PCA0CN.n)	Y		EPCA0 (EIE1.3)	PPCA0 (EIP1.3)
Comparator 0 Falling Edge	0x0053	10	CP0FIF (CPT0CN.4)			ECP0F (EIE1.4)	PCP0F (EIP1.4)
Comparator 0 Rising Edge	0x005B	11	CP0RIF (CPT0CN.5)			ECP0R (EIE1.5)	PCP0R (EIP1.5)
Comparator 1 Falling Edge	0x0063	12	CP1FIF (CPT1CN.4)			ECP1F (EIE1.6)	PCP1F (EIP1.6)
Comparator 1 Rising Edge	0x006B	13	CP1RIF (CPT1CN.5)			ECP1R (EIE1.7)	PCP1F (EIP1.7)
Timer 3 Overflow	0x0073	14	TF3 (TMR3CN.7)			ET3 (EIE2.0)	PT3 (EIP2.0)

Registres de validations :

- Priorité
  - **IP**
  - **EIP1**
  - **EIP2**
- Autorisation :
  - **IE**
  - **EIE1**
  - **EIE2**

Table des vecteurs d'interruption (cas du 8051F020)

ADC0 End of Conversion	0x007B	15	AD0INT (ADC0CN.5)	Y		EADC0 (EIE2.1)	PADC0 (EIP2.1)
Timer 4 Overflow	0x0083	16	TF4 (T4CON.7)			ET4 (EIE2.2)	PT4 (EIP2.2)
ADC1 End of Conversion	0x008B	17	AD1INT (ADC1CN.5)			EADC1 (EIE2.3)	PADC1 (EIP2.3)
External Interrupt 6	0x0093	18	IE6 (P3IF.5)			EX6 (EIE2.4)	PX6 (EIP2.4)
External Interrupt 7	0x009B	19	IE7 (P3IF.6)			EX7 (EIE2.5)	PX7 (EIP2.5)
UART1	0x00A3	20	RI1 (SCON1.0) TI1 (SCON1.1)			ES1	PS1
External Crystal OSC Ready	0x00AB	21	XTLVLD (OSCXCN.7)			EXVLD (EIE2.7)	PXVLD (EIP2.7)

Registres de validations :

- Priorité
  - **IP**
  - **EIP1**
  - **EIP2**
- Autorisation :
  - **IE**
  - **EIE1**
  - **EIE2**

Rappel : la zone mémoire SFR (cas du 8051F020)

F8	SPI0CN	PCA0H	PCA0CPH0	PCA0CPH1	PCA0CPH2	PCA0CPH3	PCA0CPH4	WDTCN
F0	B	SCON1	SBUF1	SADDR1	TL4	TH4	EIP1	EIP2
E8	ADC0CN	PCA0L	PCA0CPL0	PCA0CPL1	PCA0CPL2	PCA0CPL3	PCA0CPL4	RSTSRC
E0	ACC	XBR0	XBR1	XBR2	RCAP4L	RCAP4H	EIE1	EIE2
D8	PCA0CN	PCA0MD	PCA0M0	PCA0CPM1	PCA0CPM2	PCA0CPM3	PCA0CPM4	
D0	PSW	REF0CN	DAC0L	DAC0H	DAC0CN	DAC1L	DAC1H	DAC1CN
C8	T2CON	T4CON	RCAP2L	RCAP2H	TL2	TH2		SMB0CR
C0	SMB0CN	SMB0STA	SMB0DAT	SMB0ADR	ADC0GTL	ADC0GTH	ADC0LTL	ADC0LTH
B8	IP	SADEN0	AMX0CF	AMX0SL	ADC0CF	P1MDIN	ADC0L	ADC0H
B0	P3	OSCXCN	OSCICN			P74OUT	FLSCL	FLACL
A8	IE	SADDR0	ADC1CN	ADC1CF	AMX1SL	P3IF	SADEN1	EMI0CN
A0	P2	EMI0TC		EMI0CF	P0MDOUT	P1MDOUT	P2MDOUT	P3MDOUT
98	SCON0	SBUF0	SPI0CFG	SPIODAT	ADC1	SPI0CKR	CPT0CN	CPT1CN
90	P1	TMR3CN	TMR3RLL	TMR3RLH	TMR3L	TMR3H	P7	
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PSCTL
80	P0	SP	DPL	DPH	P4	P5	P6	PCON

Procédure de mise en place d'une interruption en assembleur

1. Ecrire la routine d'interruption ISR (*Interrupt Service Routine*), à la manière d'un sous-programme. Par contre, cette routine doit obligatoirement se terminer par une instruction « **RETI** » au lieu de «**RET** ».
2. Placer dans la table de vecteurs d'interruption, à l'adresse réservée pour ce vecteur d'interruption, une ligne de code permettant le saut inconditionnel (type JMP) vers la routine ISR de traitement de l'interruption**.**
3. *Configurer le périphérique pour qu'il soit en mesure de produire des demandes d'interruption au moment souhaité.*
4. Configurer le bit (*Enable Flag*) dans un des registres de validation d'interruption (IE, EIE1 et EIE2) pour autoriser cette interruption**.**
5. Choisir le niveau de priorité donné à cette interruption par configuration du bit adéquat (*Priority Control*) dans un des registres de gestion des priorités (IP, EIP1 et EIP2)**.**
6. Pour terminer, autoriser la prise en charge globale des interruptions en validant le bit EA du registre IE.

Questions de révision

Questions de révisions

Le 8051est un microprocesseur :

- ❑ 8 bits de type Von Neumann car il dispose d'une espace mémoire pour le code et d'un espace mémoire pour les données.
- ❑ 16 bits de type Von Neumann car l'espace mémoire code est mélangé avec l'espace mémoire données.
- ❑ 16 bits de type Harvard car il dispose d'un registre d'adresses pour le code (PC) et d'un registre d'adresses pour les données (DPTR).
- ❑ Type Harvard car l'espace mémoire code est mélangée avec l'espace mémoire données.
- ❑ Type Harvard car l'espace mémoire code est séparé de l'espace mémoire données.

Le registre d'état du 8051 se trouve :

- ❑ Dans la zone mémoire code externe.
- ❑ Dans la zone mémoire données externe.
- ❑ Dans la zone mémoire RAM externe.
- ❑ Dans la zone mémoire RAM interne.
- ❑ Dans l'accumulateur.

Questions de révisions

Les registres internes du 8051 :

- ❑ Sont au nombre de 8.
- ❑ Sont au nombre de 32.
- ❑ Sont localisées dans la RAM interne.
- ❑ Sont localisées dans l'UAL.
- ❑ Sont au nombre de 8 localisée dans la RAM externe.

Les zones mémoires du 8051 :

- ❑ La zone data interne correspond à la zone ou sont localisés les registres internes du 8051.
- ❑ La zone data interne correspond à la zone mémoire RAM interne disponible pour l'utilisateur.
- ❑ La zone data interne correspond à la zone mémoire RAM interne ou se trouve stocké le programme de démarrage.
- ❑ La zone data externe correspond à la zone mémoire externe accessible avec l'instruction assembleur MOVX.
- ❑ La zone code externe correspond à la zone mémoire externe dans laquelle se trouve stocké le code exécutable.

Questions de révisions

Mode d'adressage du 8051 :

- L'instruction MOV permet de déplacer un octet depuis une zone mémoire interne vers une autre zone mémoire externe.
- l'instruction MOVX permet de déplacer un octet depuis une zone mémoire interne vers une zone mémoire externe.
- l'instruction MOV A, #42H utilise l'adressage immédiat et place la valeur 42 dans le registre accumulateur.
- l'instruction MOV A, 42H utilise l'adressage direct et place la valeur 42 dans le registre accumulateur.
- l'instruction MOVX A, @DPTR utilise l'adressage indirect et déplace la valeur mémorisée dans la zone mémoire dont l'adresse est stockée dans le registre DPTR vers le registre accumulateur.

Questions de révisions

Dans le cas du 8051, l'instruction :

- PUSH permet d'incrémenter de 1 la valeur du pointeur de pile SP.
- POP permet de retirer un octet de la pile et incrémenter de 1 la valeur du pointeur de pile SP.
- CALL permet de mémoriser automatiquement la valeur du pointeur de code (PC) dans la pile et décrémenter de 1 la valeur du pointeur de pile SP.
- CALL permet de mémoriser automatiquement la valeur du pointeur de code (PC) dans la pile et de décrémenter de 2 la valeur du pointeur de pile SP.
- RET permet d'affecter automatiquement au pointeur de code (PC) les deux premiers octets de la pile.

Que fait le programme suivant et quelles sont ses incidences sur le registre d'état, la pile, le pointeur de pile et le pointeur de code ?

```
        MOV A, #55H
        ADD A, #0AAH
        CALL 1234
SUITE:  JMP SUITE
```