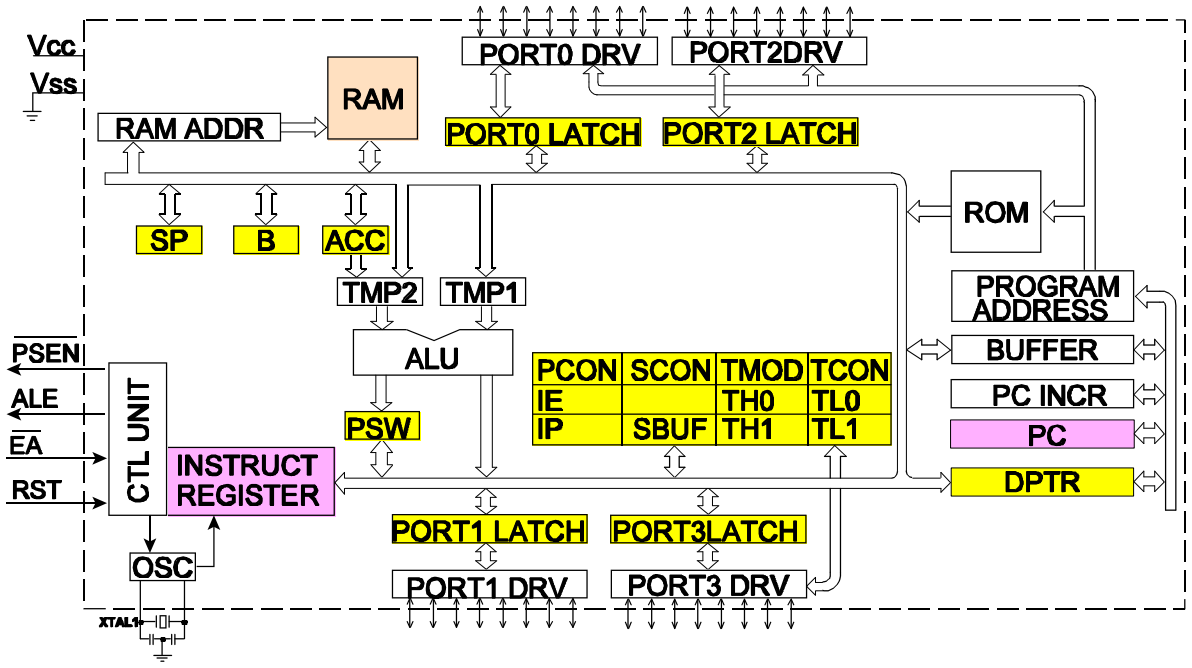
ÉLECTRONIQUE

8051 : mémoire vive et registres

8051 : mémoire vive/registres



Nous allons maintenant détailler la mémoire de données interne qui comprend essentiellement
- une RAM
- des registres banalisés (GPR General Purpose Register)
- des registres spéciaux (Special Function Register)

Distinction entre RAM et registres

- RAM : usage banalisé
  - stockage de variables statiques ordinaires
  - variables dynamiques (heap)
  - paramètres, adresses de retour, variables locales (stack)
  - interne, externe, ou les deux
- REGISTRES
  - accès plus rapide car ∃ mode d'adressage spécial
  - GPR (General Purpose Register)
    - variables statiques à usage fréquent
    - paramètres, variables locales
  - SFR (Special Function Register) : rôle spécial
    - PC, SP
    - ACC
    - registres d'adressage indirect
    - statut
    - configuration des périphériques
    - entrée-sortie

L'accès aux données en RAM interne est plus rapide qu'en RAM externe, essentiellement parce que les capacités parasites et parfois les tensions logiques sont plus faibles en interne et autorisent des fréquences plus élevées. Tous les microprocesseurs ne possèdent pas de RAM interne mais la plupart des micro-contrôleurs en sont équipés.
Il y a encore une différence de rapidité d'exécution suivant que les données sont contenues en RAM (même interne) ou dans les registres. La raison est ici liée au jeu d'instructions du processeur.
Lors de l'exécution d'une instruction, le processeur effectue en général
- 1 accès en mémoire programme pour y lire le code de l'instruction ("op code fetch")
- 1 ou plusieurs accès pour lire les opérandes

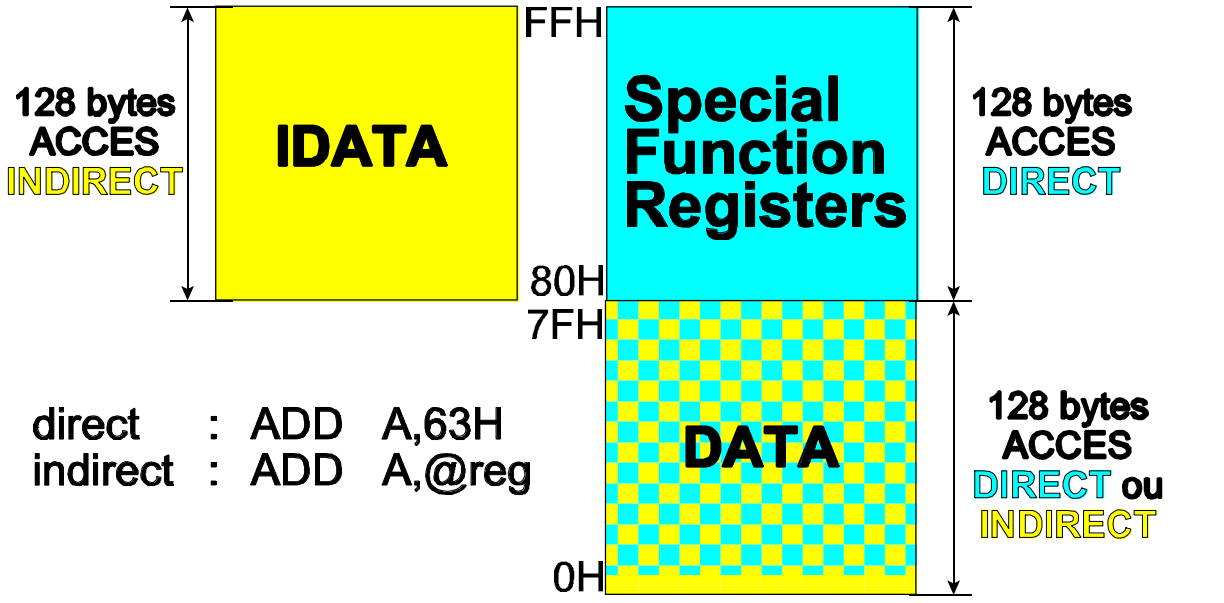
Dans le cas où l'opération porte sur le contenu des registres, il existe des instructions spéciales (ou plus exactement des modes d'adressage spéciaux) dans lesquels les adresses des registres contenant les opérandes sont incluses dans le code d'instruction. On économise ainsi plusieurs lectures en mémoire programme, ce qui accélère considérablement l'exécution.

La RAM, interne ou externe, sera utilisée pour :
- les variables statiques normales du programme
- les variables dynamiques (dans une partie de la RAM appelée "HEAP")
- la pile ou "STACK" réservée pour stocker les adresses de retour lors des appels de fonction et des interruptions, le passage des paramètres, les variables locales,...

Les registres se partagent encore en deux grandes catégories
- les GPR (General Purpose Registers) pour stocker des variables à accès rapide et fréquent (cf variables déclarées "register" en C)
- les SFR (Special Function Registers) qui ont chacun une fonction bien précise
  - accumulateur(s) comme données et résultats des opérations arithmétiques et logiques
  - compteur de programme, stack pointer
  - registres d'adressage
  - registres de statut
  - registres de configuration
  - registres d'entrée-sortie

REM : certains processeurs ne font aucune différence entre RAM interne et registres GPR; on désigne alors la RAM par le terme "REGISTER FILE" (fichier de registres)

8051: RAM / registres internes



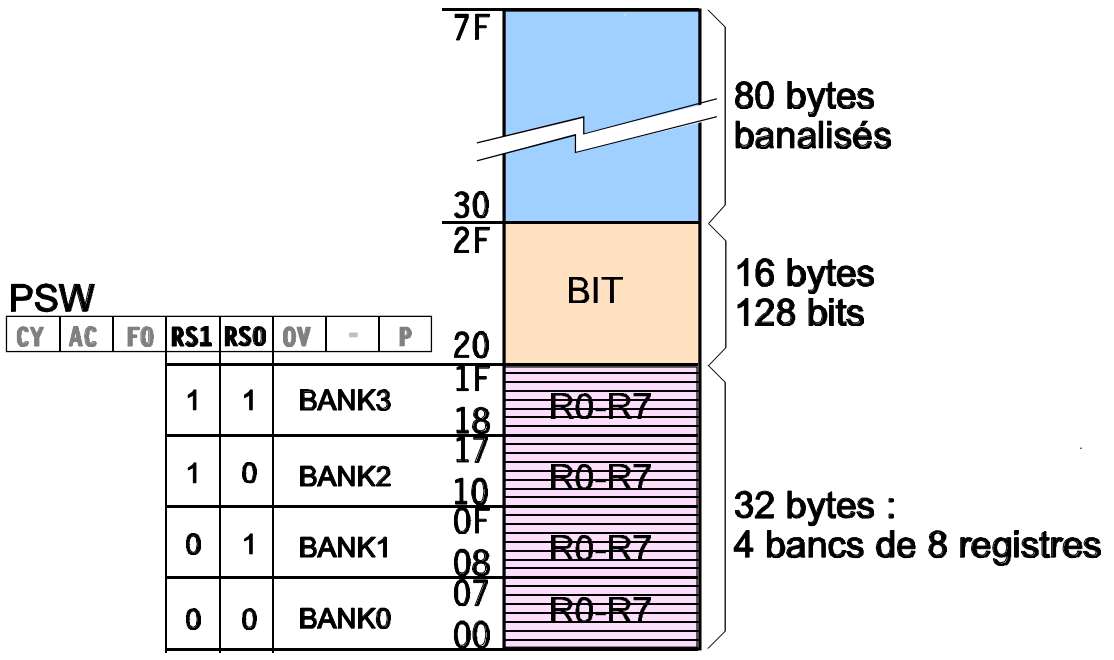
Cette figure montre la carte de mémoire ou “Memory mapping” interne du 8051, partagée en 3 blocs de 128 octets.

Le bloc qui commence à l'adresse 0 peut être adressé indifféremment de manière

- directe : l'instruction se réfère directement à l'adresse de l'opérande
  ADD A,63H signifie "ajouter le contenu de la case mémoire d'adresse 63H à l'accumulateur et placer le résultat dans l'accumulateur”
- indirecte : l'instruction se réfère à un registre qui contient l'adresse de l'opérande
  ADD A,@reg signifie "ajouter le contenu de la case mémoire dont l'adresse est contenue dans le registre reg à l'accumulateur et placer le résultat dans l'accumulateur”

Les deux blocs supérieurs partagent curieusement la même adresse ! La différence se fait sur le mode d'adressage :
- les SFR sont adressables uniquement de manière directe
- la zone IDATA (Indirect DATA) est uniquement accessible par adressage indirect, ce qui en fait un bon emplacement pour la pile (stack)

8051: zone de mémoire DATA



La zone DATA est constituée par les 128 octets inférieurs de la mémoire; ils sont tous accessibles, comme nous venons de le voir
- soit par adressage direct (via leur adresse de 00H à 7FH)
- soit indirectement via un registre de 8 bits

La zone DATA est encore partagée en 3 blocs, en liaison avec des modes d'adressages particuliers.

1°) les 32 premiers octets sont en plus accessibles en tant que registres par les noms R0 à R7
Attention ! il y a 4 bancs dont les registres portent le même nom et seul un banc est accessible à la fois.
La sélection du banc s'opère via les bits PSW.4 et PSW.3 du registre de statut PSW (Program Status Word)

2°) les 16 octets suivants peuvent être adressés soit comme 16 octets, soit comme 128 bits, à l'aide d'instructions spéciales de manipulation de bits

3°) les 80 octets restants n'ont que les deux modes classiques, direct et indirect

DATA : zone adressable par bit

manipulation d'un booléen de 1 seul bit

ADRESSE D'OCTET

```
MOV  A,27H
ANL  A,10111111B
MOV  27H,A
```

ADRESSE DE BIT

```
CLR  3EH
```

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

En zoomant sur la zone adressable par bit on voit que :
- les instructions portant sur un octet entier utilisent les adresses d'octet comprises entre 20H et 2FH (soit 16 adresses)
- les instructions portant sur les bits utilisent des adresses spéciales désignant individuellement chaque bit de la zone

Prenons comme exemple la remise à 0 du 7ème bit de l'octet 27H.

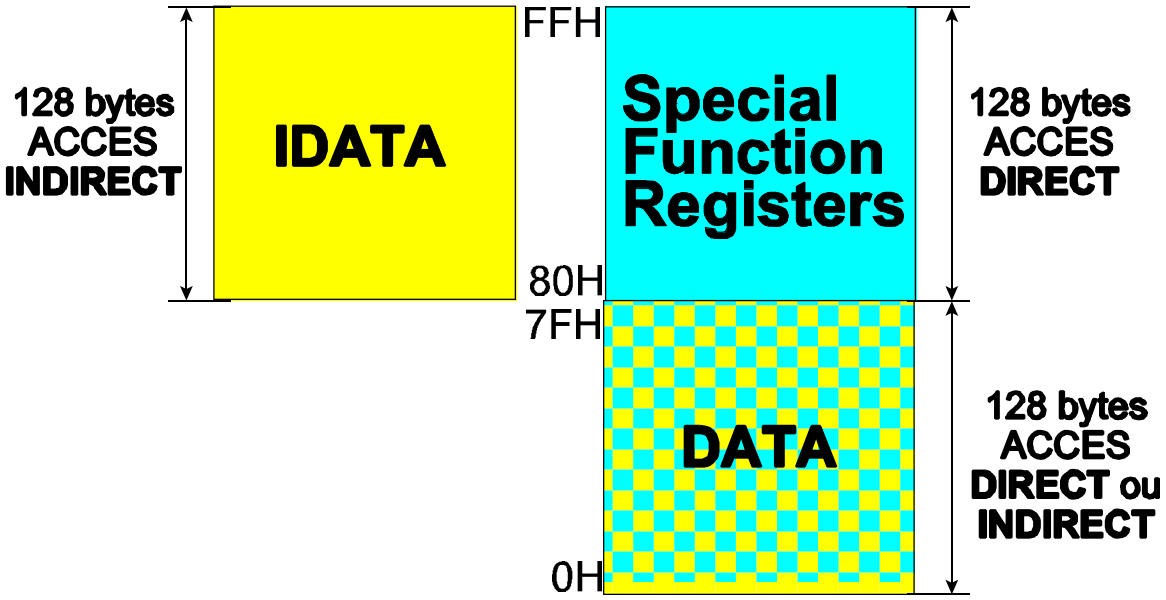
Si l'on ne dispose pas d'instructions portant directement sur les bits, on doit
- effectuer une lecture de l'octet pour l'amener dans l'accumulateur
- faire une opération de AND logique bit-à-bit avec le masque 7FH (comprenant un seul 0 au bit 7)
- réécrire l'octet modifié à sa place

Le 8051 connaît l'instruction CLR qui effectue la même opération en se référant à l'adresse du bit (ici 3EH).

En réalité, l'instruction CLR effectue la même suite d'opération lecture-modification-réécriture, c'est ce que l'on appelle une instruction de "Read-Modify-Write".

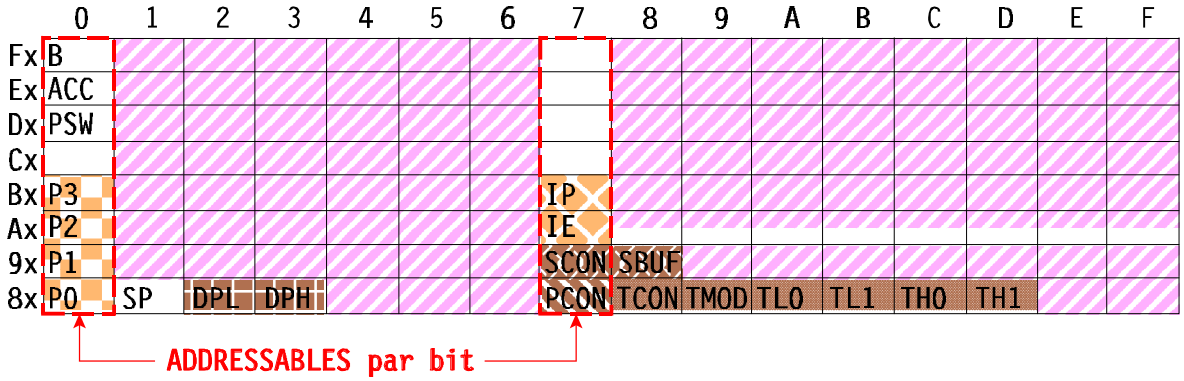
Le gain en temps d'exécution résulte donc du fait qu'il n'y a qu'un seul code opératoire à extraire de la mémoire programme.

8051: Special Function Register



Nous allons maintenant détailler les registres spéciaux ou SFR (Special Function Register).

Special Function Registers



accès à une RAM externe
gestion de l'économie d'énergie
gestion du port série
gestion des timers
gestion des interruptions
ports d'I/O parallèle
registres inexistants = réservés pour extensions ; ne pas utiliser

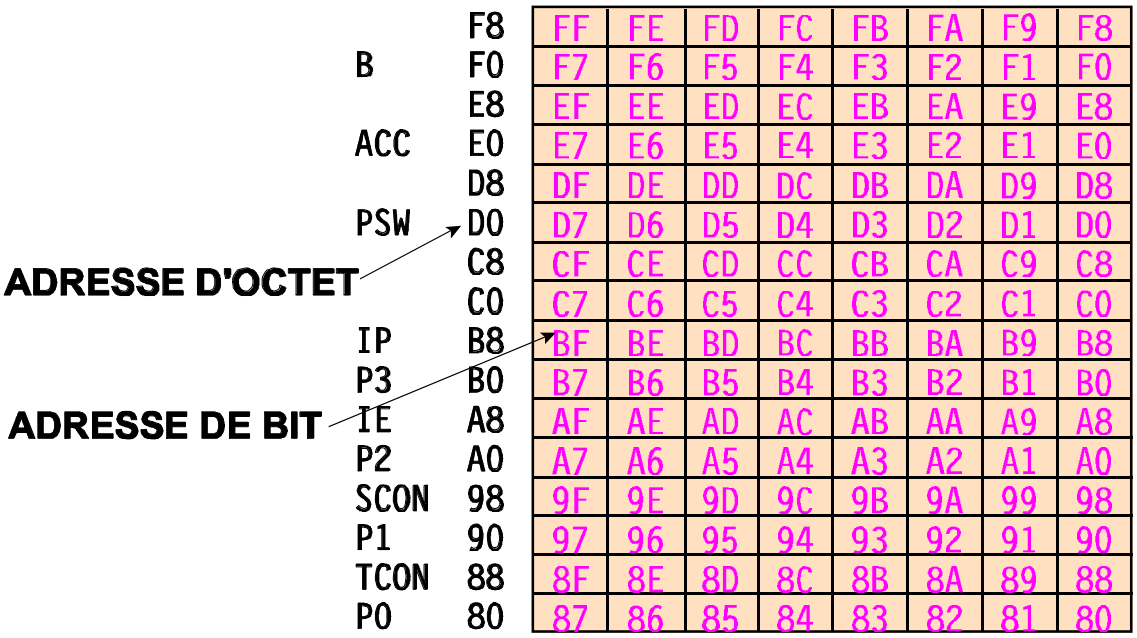
On voit ici la carte des SFR du 8051 "de base". Chaque registre possède une fonction spécifique :

- P0, P1, P2 et P3 sont les registres d'entrée-sortie parallèle; tout nombre lu (ou écrit) dans ces registres est le reflet des (ou se reflète sur les) bornes correspondantes du boîtier; nous reviendrons sur la manière de programmer le sens de fonctionnement de chaque bit.
- PSW est le registre de statut
- ACC et B sont les accumulateurs, c'est à dire les entrées de l'unité arithmétique et logique
- SP est le (Stack Pointer) qui pointe sur la première adresse libre de la pile (ou sur la dernière adresse utilisée suivant le type de processeur)
- DPH et DPL forment un registre de 16 bits utilisé pour accéder à une éventuelle RAM (ou à des périphériques) externes
- IP et IE sont des registres de configuration et de masquage des interruptions
- PCON gère l'économie d'énergie
- SCON configure le port série et SBUF est le registre d'échange de données via le port série
- tous les registres commençant par T ont trait à la configuration des Timers/Counters

On remarquera que tous les registres pour lesquels cela présente un intérêt sont adressables par bits. Pour les autres, ce serait superflu.

La zone dévolue aux SFR comporte de larges trous inutilisables, car réservés aux extensions futures. Tous les dérivés modernes du 8051 possèdent des périphériques internes supplémentaires, dont les registres de configuration et de données font partie des SFR et sont placés aux endroits libres de cette carte.

SFR : zone adressable par bit



	F8	FF	FE	FD	FC	FB	FA	F9	F8
B	F0	F7	F6	F5	F4	F3	F2	F1	F0
	E8	EF	EE	ED	EC	EB	EA	E9	E8
ACC	E0	E7	E6	E5	E4	E3	E2	E1	E0
	D8	DF	DE	DD	DC	DB	DA	D9	D8
PSW	D0	D7	D6	D5	D4	D3	D2	D1	D0
	C8	CF	CE	CD	CC	CB	CA	C9	C8
	C0	C7	C6	C5	C4	C3	C2	C1	C0
IP	B8	BF	BE	BD	BC	BB	BA	B9	B8
P3	B0	B7	B6	B5	B4	B3	B2	B1	B0
IE	A8	AF	AE	AD	AC	AB	AA	A9	A8
P2	A0	A7	A6	A5	A4	A3	A2	A1	A0
SCON	98	9F	9E	9D	9C	9B	9A	99	98
P1	90	97	96	95	94	93	92	91	90
TCON	88	8F	8E	8D	8C	8B	8A	89	88
P0	80	87	86	85	84	83	82	81	80

La table ci-dessus donne la cartographie de la zone des SFR avec l'adressage par octet et par bit, en notation hexadécimale.

SFR : manipulation par bit

- intérêt
  - lecture et test d'un bit sans masque
  - modification d'un bit en une instruction

```
MOV    P0,#37H     initialisation par byte en une instruction
SETB   P0.7        } création d'une
CLR      P0.7      }   impulsion sur bit7 du PORT0
```


La possibilité de manipuler bit-à-bit les SFR, et en particulier les ports d'entrée-sortie, témoigne de la vocation de contrôle-commande des micro-contrôleurs.

Supposons que l'on veuille initialiser tous les bits du port P0 (par exemple juste après le RESET). On peut le faire en une seule l'instruction

MOV P0,#37H (le signe # désigne une constante et le suffixe H la notation hexadécimale);

P0 prend donc la valeur binaire 00110111

Maintenant, on peut créer une impulsion courte (coup d'horloge par exemple) par les instructions de manipulation de bit

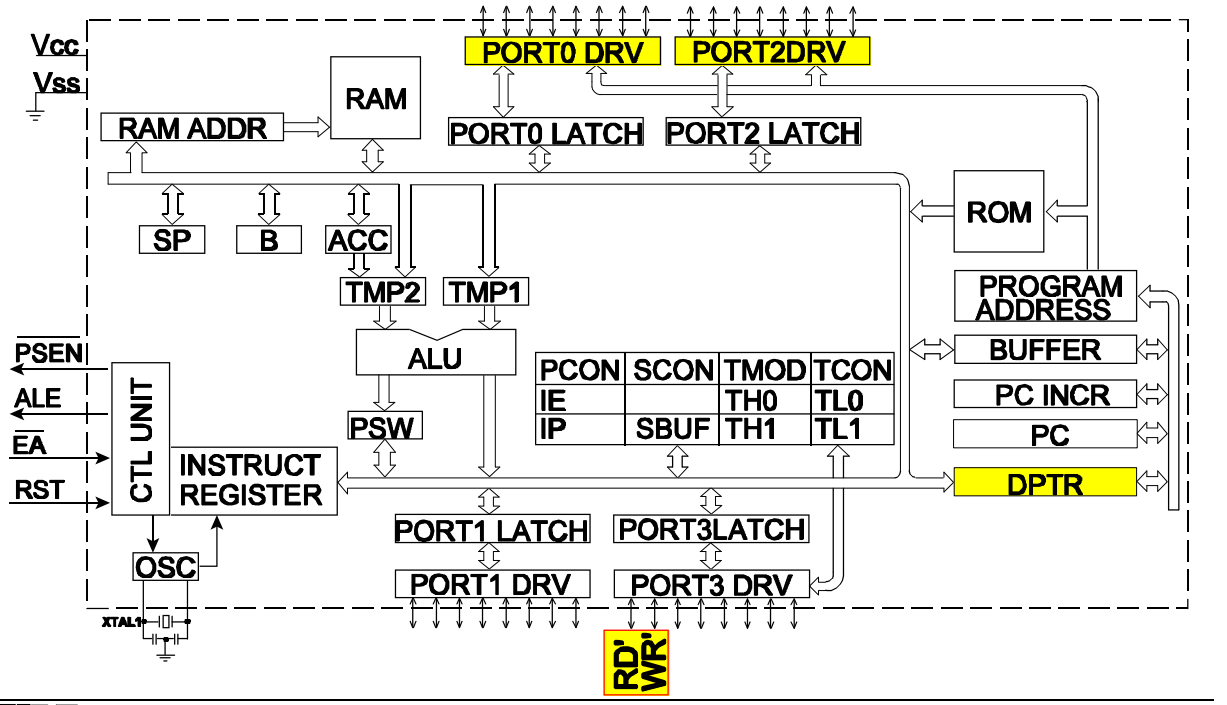
SETB P0.7 active en HI le bit de poids le plus fort du port P0

CLR P0.7 remet ce bit à l'état LO

Ces instructions sont très économes en mémoire et en temps d'exécution, ce qui compense la faiblesse des ressources internes de tels micro-contrôleurs.

Remarquons que le programmeur ne doit pas employer les adresses hexadécimales des registres ou des bits. Aussi bien l'assembleur que les compilateurs acceptent des noms symboliques tels que P0 et P0.x, beaucoup plus lisibles.

8051 : RAM externe



La RAM interne peut se révéler insuffisante, surtout si l'on manipule des tableaux ou si l'on a besoin de variables dynamiques.
Dans la famille 8051, l'accès à une RAM externe se fait comme pour la mémoire programme externe, via les ports P0 et P2 reconvertis en bus d'adresses/données multiplexé.
Les signaux de contrôle sont différents : au lieu de PSEN' (signal de lecture en mémoire programme) on utilise

- RD' signal de lecture en RAM
- WR' signal d'écriture en RAM

Ces signaux doivent malheureusement être "empruntés" au port P3, où l'on perd donc 2 bits d'entrée-sortie.

Le registre spécial DPTR (Data PoinTeR) permet l'accès à la RAM externe par un adressage indirect; il n'y a pas d'adressage direct possible.

8051 : RAM externe

- plusieurs types
  - SRAM (RAM statique)
  - SRAM + pile
  - E²PROM
  - DRAM : pas justifié (quantité trop faible)
- motivations pour mémoire RAM externe
  - pas assez de RAM interne
  - mélanger plusieurs types de mémoire
  - ajouter un type de mémoire non disponible en interne
    - RAM + pile
    - E²PROM
  - ajouter des périphériques "memory mapped"

Pour les micro-contrôleurs tels que le 8051, le bus d'adresse est de 16 bits et le bus de données de 8 bits, ce qui fournit un espace de 64Koctets pour les données externes. Une telle quantité de RAM est, de nos jours, suffisamment bon marché en RAM statique (SRAM) et ne justifie pas l'emploi de RAM dynamiques (DRAM), qui sont utilisées dans les micro-ordinateurs pour des quantités de mémoire 100 à 1000 fois supérieures.

La zone de 64Ko peut aussi être partagée entre plusieurs boîtiers différents,

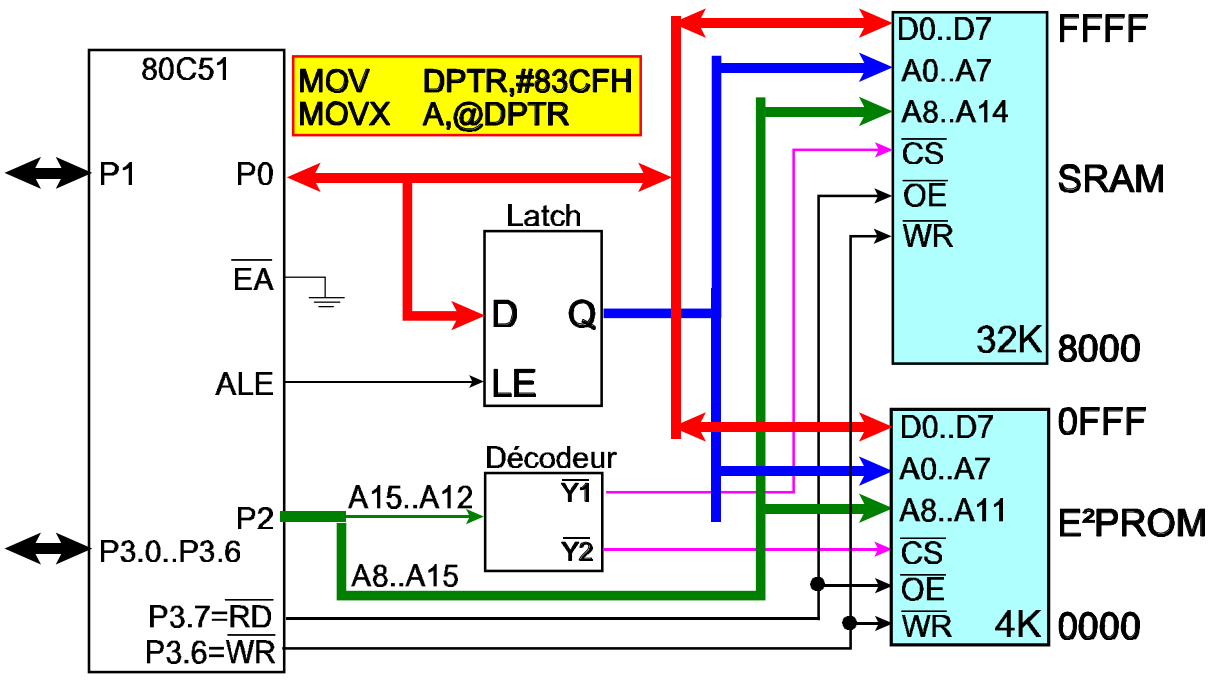
- pour bénéficier des avantages de différentes technologies
  - RAM pour les variables ordinaires, la pile et le heap
  - NOVRAM ou E²PROM pour les paramètres non volatiles
- pour ajouter des périphériques non-présents en interne; les registres de données et de configuration de ces périphériques seront vus comme des octets de mémoire ("memory mapped I/O").

Types de mémoire RAM

indications

type	convient mieux pour
SRAM	données normales volatiles
SRAM+pile	données non-volatiles - indications de fabrication (N° de série) - paramètres de configuration - résultats de mesure - calibration - boîte noire
E²PROM	idem mais - temps d'écriture + long (100µs) - qqK à qq 100K écritures

Interface vers RAM externe scindée



Ce schéma-bloc montre comment scinder l'espace de 64Ko dévolu à la RAM en 2 boîtiers, par exemple une E²PROM de 4Ko et une SRAM de 32Ko.
Comme nous l'avons vu pour la mémoire programme, c'est le décodeur d'adresses qui, au départ des adresses hautes (bits de poids le plus fort), va créer les deux signaux CS' (Chip Select) mutuellement exclusifs sélectionnant l'un ou l'autre des boîtiers. On fixe ainsi leur position respective dans la cartographie de la mémoire.

Le démultiplexage du bus se fait par le même LATCH que pour la mémoire programme.

Par rapport à la mémoire programme, les signaux de contrôle sont différents
- RD' (ReaD) est activé à l'état bas lorsque le processeur lit en mémoire, et est connecté à la borne OE' (Output Enable) des mémoires pour faire quitter l'état haute impédance aux buffers de sortie de la mémoire sélectionnée. L'autre (ou les autres) boîtier(s) restent en haute impédance parce que leur CS' n'est pas activé.
- WR' (WRite) est activé si le processeur veut écrire en mémoire; sa remontée indique à la mémoire que la donnée mise sur le bus D0-D7 par le processeur est valable et doit être verrouillée au sein de la mémoire immédiatement, car on va entamer le cycle suivant et les données vont changer.

Pour le programmeur, l'accès à un octet de mémoire se fait en préchargeant le registre DPTR (DataPoinTeR) avec une constante de 16 bits égale à l'adresse cible en mémoire (ici 83CFH)

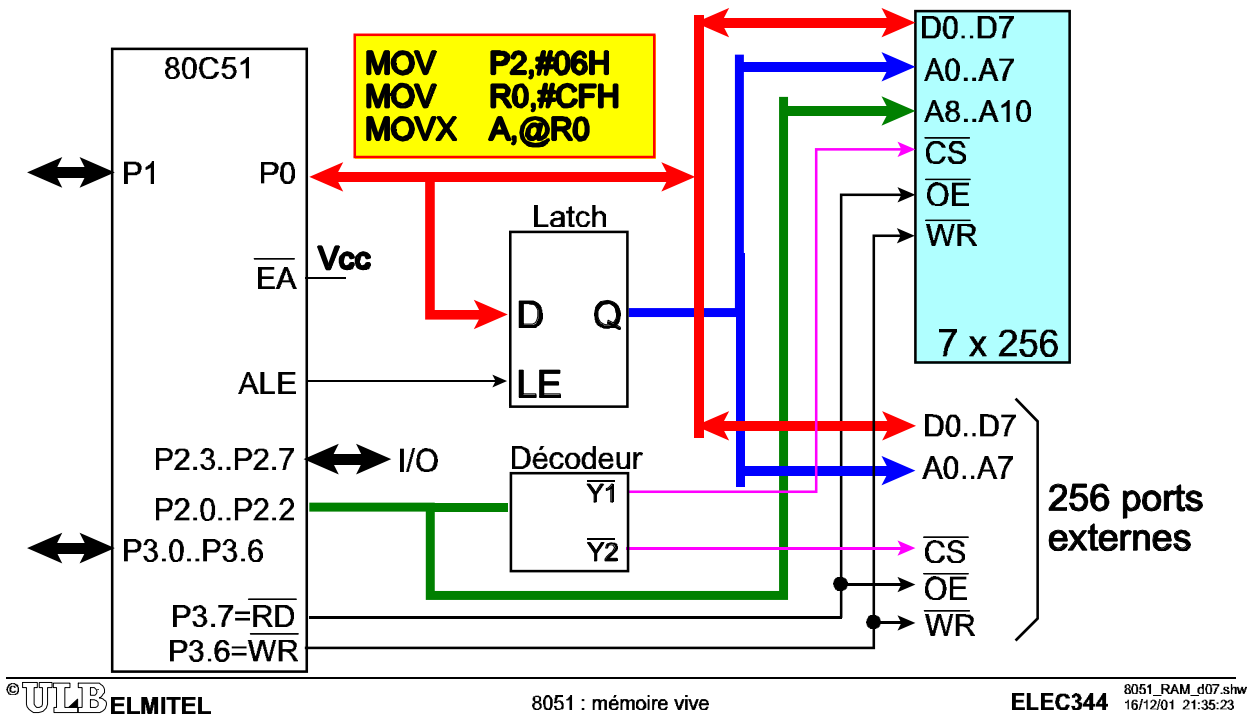
```
MOV  DPTR,#83CFH
```

puis en faisant une écriture (ou lecture) de type indirect (seul mode possible ici)

```
MOVX   A,@DPTR   transfère le contenu de l'adresse 83CFH dans l'accumulateur A
MOVX   @ DPTR, A  écrit le contenu de l'accumulateur à l'adresse 83CFH
```

Dans le mnémonique MOVX, le suffixe X signifie "eXternal" pour le différencier du MOV normal en RAM interne, le signe @ marque l'indirection.

ROM interne / RAM externe paginée



Si l'on veut utiliser de faibles quantités de RAM externe et que l'on n'a pas besoin de ROM externe, il est dommage de sacrifier totalement les deux ports P0 et P2 pour fabriquer les bus d'adresses et de données. On peut alors utiliser uniquement P0, ce qui donne accès à 256 octets externes.

Si c'est insuffisant, on a recours à la "pagination" de la mémoire externe en plusieurs blocs de 256 octets, en se servant par exemple de quelques bits de P2 pour commander la sélection d'une page à la fois.
Dans l'exemple ci-dessus, on utilise les trois bits de poids faible du port P2 (P2.2, P2.1 et P2.0) pour piloter un décodeur 3 vers 8, qui active 1 page de 256 octets parmi les 8 possibles. On peut par exemple utiliser une des sorties du décodeur pour définir une "page d'entrée-sortie" où l'on placera des périphériques externes.

Les 5 bits P2.3 à P2.7 restent disponibles pour des entrées-sorties banalisées.

Pour le programmeur, l'accès à un octet de mémoire se fait d'abord en chargeant le numéro de la page dans le port P2 (sans toucher aux 5 bits de poids forts P2.3 à P2.7 s'ils sont utilisés à d'autres fins !)

```
MOV     P2,#06
```

sélectionne la page 6 (en supposant que l'on puisse mettre à 0 les autres bits)

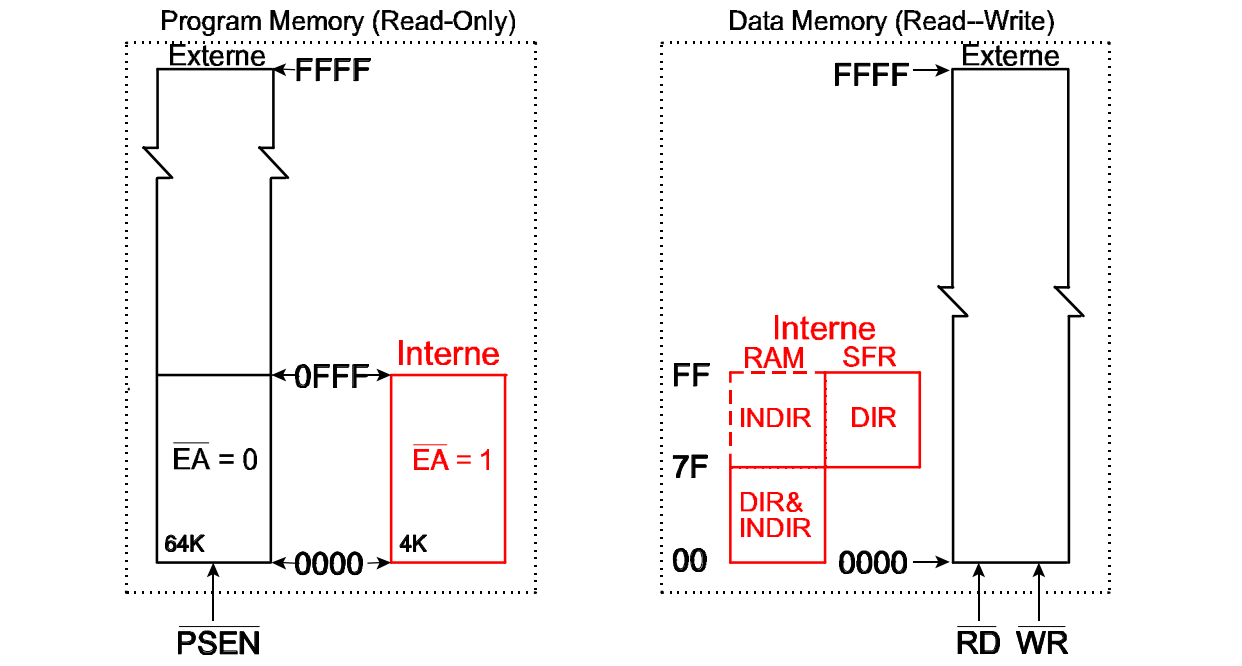
On utilise alors le registres R0 comme registre d'adressage indirect, en y chargeant le numéro (ou offset) de l'octet visé dans la page (ici CFH)

On termine par l'instruction de lecture indirecte se référant à R0

```
MOVX    A,@R0          @ marquant l'indirection
```

REM : l'entrée EA' (External Address) doit être désactivée, puisque l'on utilise exclusivement la mémoire programme interne

Mémoires du 8051 : résumé



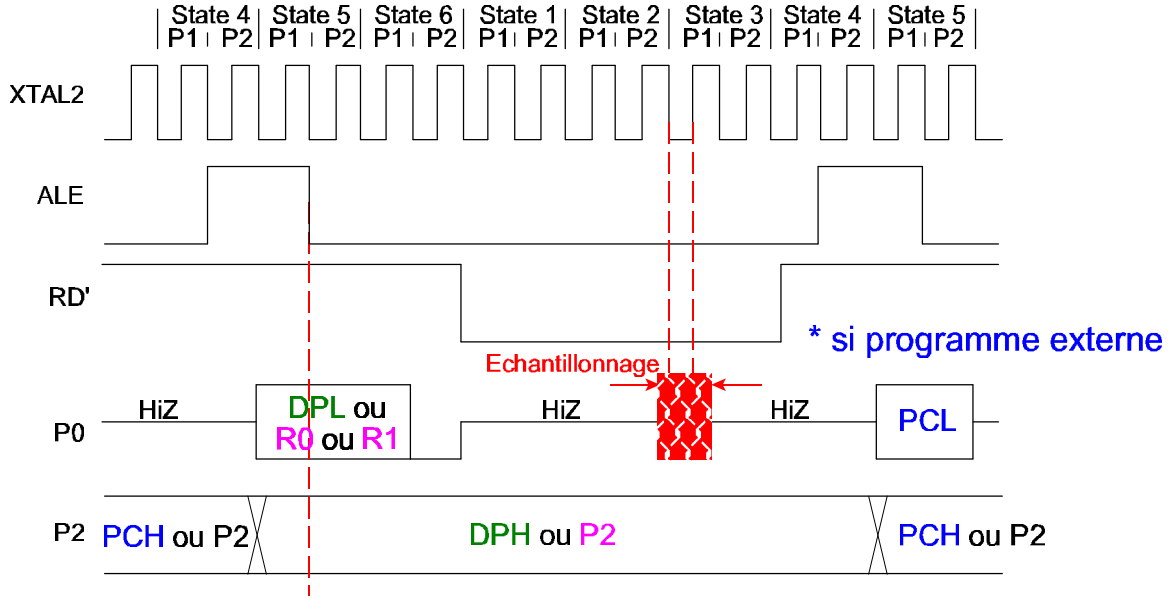
En résumé, le sacrifice des 2 ports d'entrée-sortie P0 et P2 donne au processeur un espace d'adressage externe de 64Ko.
En réalité cet espace est de 2 x 64Ko grâce au dédoublement des signaux de contrôle :

- 64Ko de mémoire programme dont la lecture est activée par PSEN'
- 64Ko de mémoire vive dont la lecture (l'écriture) est activée par RD' (WR')

Si de la mémoire programme interne est présente
- si EA' est activé, alors la mémoire interne est ignorée et PSEN' est actif sur les 64Ko de mémoire
- si EA' est désactivé, alors la mémoire interne est utilisée dans ses limites et PSEN' n'est activé que si l'on fait accès à une adresse au-dessus de cette limite.

Pour rappel on dispose comme mémoire vive interne de
- 128 octets de DATA (RAM+GPR) internes à accès direct ou indirect
- 128 octets de IDATA internes à accès indirect
- un espace de 128 octets partiellement peuplé de SFR, à accès direct

Chronogramme de lecture en XDATA



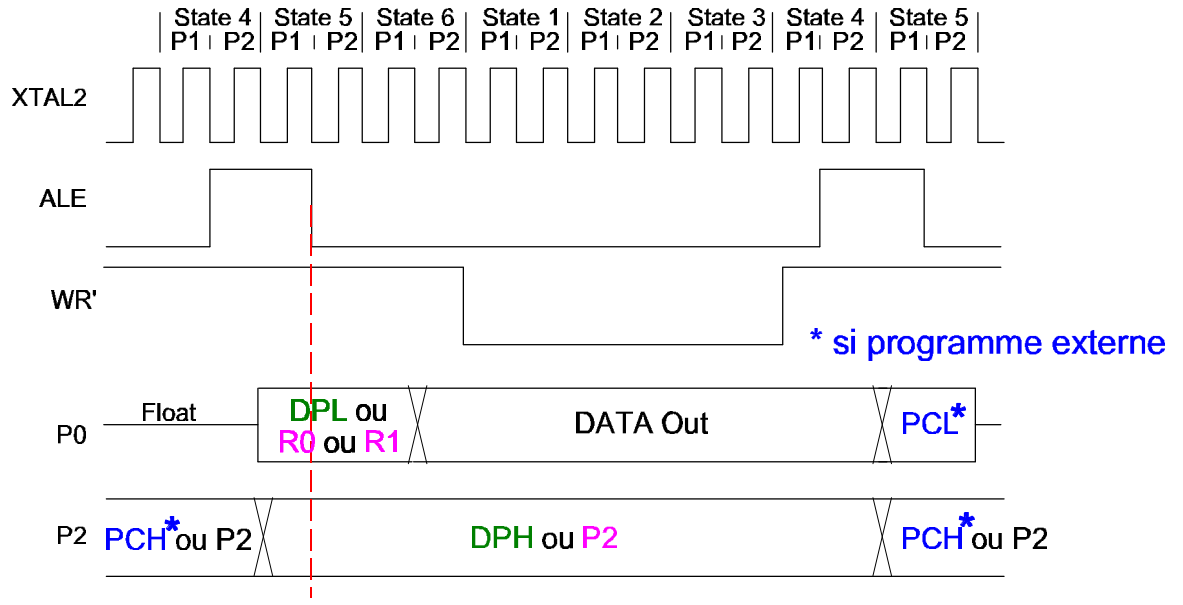
La figure montre le chronogramme d'une lecture en mémoire RAM externe.

- le processeur commence par rendre transparent le latch de démultiplexage en activant le signal ALE
- l'adresse basse est alors placée sur P0 qui quitte l'état de haute impédance (provenant de sa précédente utilisation comme bus de données). L'adresse basse provient
  - soit du contenu de R0 ou R1 si l'on exécute un MOVX @Ri (adressage indirect sur 8 bits)
  - soit de la partie basse de DPTR appelée DPL par MOVX @DPTR (adressage indirect sur 16 bits)

- s'il y a une mémoire programme externe, le contenu précédent de P2 était la partie haute du compteur de programme; sinon c'était la valeur issue du latch du port P2.
- si l'adressage se fait sur 8 bits (MOVX @Ri), P2 ne change pas; si l'adressage se fait sur 16 bits (MOVX@DPTR), P2 prend la partie haute de DPTR, appelée DPH.
- une fois l'adresse stable, ALE est désactivé et le latch de démultiplexage verrouille l'adresse basse
- le processeur active la ligne RD', met ses propres buffers de sortie de P0 en haute impédance et laisse la mémoire imposer le niveau logique sur le bus de données, puisque l'activation de RD' débloque les buffers de sortie de la mémoire
- les données placées par la mémoire sur le bus sont échantillonnées pendant la phase P1 du stade 3 du cycle
- le processeur désactive la ligne RD', pour indiquer la fin de la lecture et remettre le buffer de sortie de la mémoire en haute impédance

- si la mémoire programme est externe, le prochain cycle externe est une lecture en mémoire programme et c'est le compteur de programme qui apparaît sur le bus d'adresse.

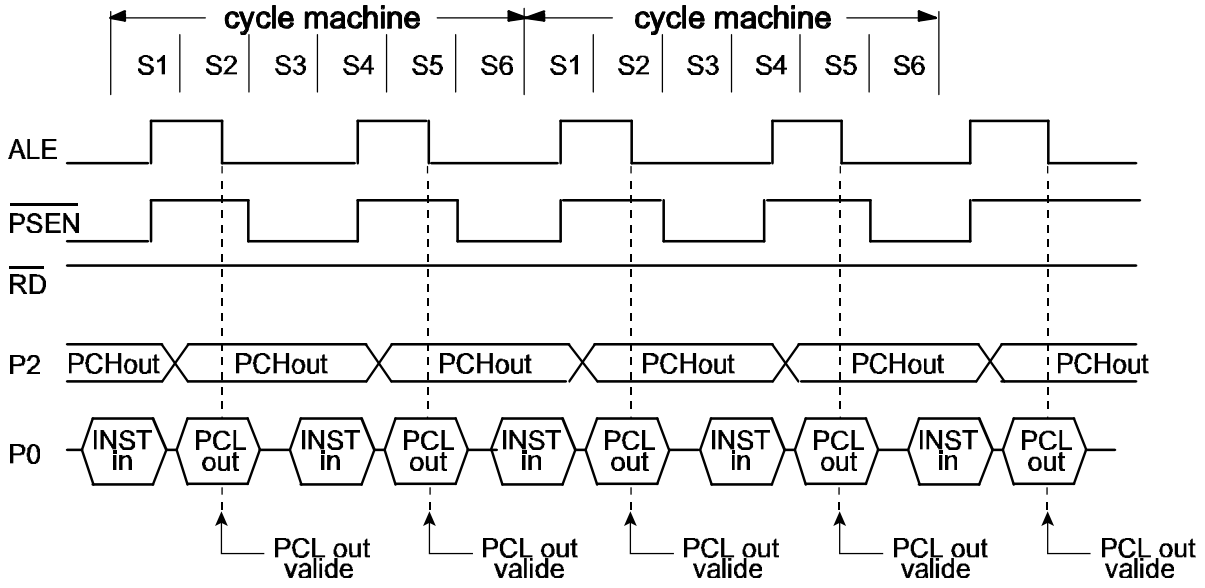
Chronogramme d'écriture en XDATA



Dans le cas d'un cycle d'écriture, le mécanisme d'adressage est le même.
La différence par rapport au cycle de lecture porte sur :

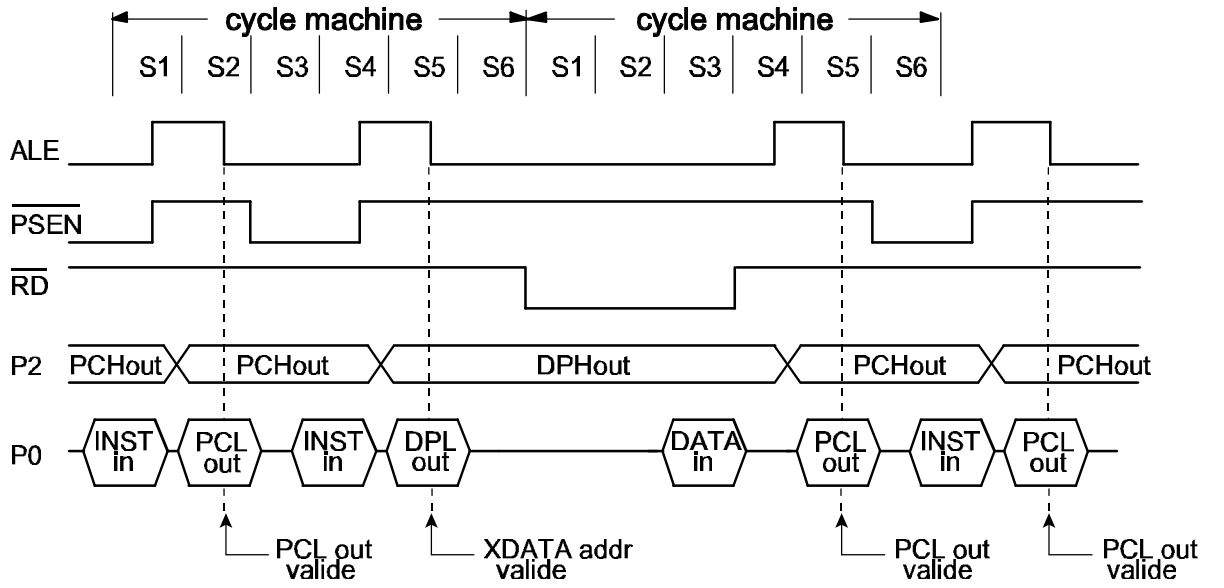
- l'activation de WR' au lieu de RD'
- le port P0 ne passe pas en haute impédance puisque c'est lui qui va mettre la donnée sur le bus
- le processeur indique la fin de cycle (c'est à dire l'ordre pour la mémoire de verrouiller la valeur du bus de données avant qu'elle ne change) par la désactivation de WR'

Accès au programme externe (sans accès aux données externes)



Dans un cycle d'accès normal en mémoire programme externe, 2 lectures successives (INST in) ont lieu. PSEN' est activé et RD' reste inactif, marquant l'accès à la mémoire programme.

Accès au programme externe (avec accès aux données externes)



Dans un cycle d'accès à la mémoire programme externe, suivi d'un accès à la RAM externe, on voit que, immédiatement après la lecture de l'instruction (milieu du Stade S4) :

- PSEN' est désactivé, ce qui met la mémoire programme en haute impédance
- ALE est activé pour déverrouiller le latch de démultiplexage
- le contenu de DPTR apparaît sur le bus d'adresse
- RD' est activé

La lecture en mémoire vive externe prend 1 cycle