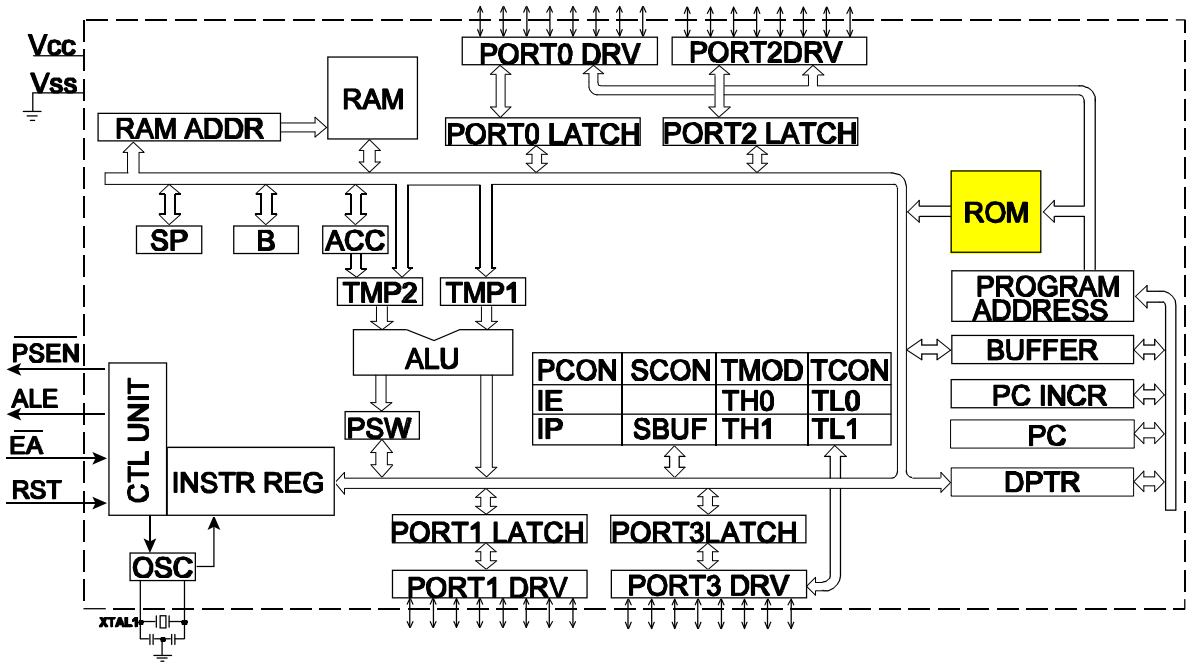
ÉLECTRONIQUE

8051 : mémoire programme

8051 : mémoire programme interne



3

Nous allons nous intéresser ici à la mémoire programme du 8051.

C'est une de manière tout à fait classique une mémoire :

- ***non volatile*** : son contenu n'est pas perdu lorsque l'on cesse d'alimenter le circuit
- ***ROM*** : (Read Only Memory) ou "***mémoire morte***" c'est à dire uniquement accessible en lecture pendant l'exécution du programme

4

8051 : mémoire programme

- plusieurs qualités
  - 83Cxx : ROM interne
  - 87Cxx : EPROM interne
  - 89Cxx : FLASH interne

  } **4 .. 64 Kbyte**
  - 80Cxx : pas de mémoire interne
- motivations pour mémoire morte externe
  - pas (assez) de mémoire interne
  - mélanger plusieurs types de mémoire
  - ajouter un type de mémoire non disponible en interne
    - RAM + pile
    - E²PROM
    - FLASH

Suivant le type de 8051, la mémoire programme peut être fabriquée en différentes technologies, indiquées par le deuxième chiffre du marquage :

- 83Cxxx : "mask ROM" : mémoire où l'inscription du programme se fait par la personnalisation du dernier masque de fabrication. Le programmeur doit donc envoyer un fichier de données au fabricant du micro-contrôleur et commander une série d'au moins 10000 pièces
- 87Cxxx : EPROM ou "Erasable Programmable Read-Only Memory"; peut être programmée électriquement par l'utilisateur en quelques dizaines de secondes et son contenu effacé par exposition aux ultra-violets pendant environ 15 minutes
- 89Cxxx : FLASH EPROM est une mémoire EPROM pouvant être effacée électriquement

On peut aussi acheter des 8051 sans mémoire programme interne. Il faut alors configurer ce micro-contrôleur dans un mode où il peut gérer une mémoire placée hors du boîtier (voir plus loin).

On peut aussi recourir à de la mémoire programme externe même s'il en existe en interne parce que
- la quantité de mémoire interne est insuffisante
- on veut profiter des qualités respectives des différents types de mémoire (par exemple une partie du programme immuable en ROM interne, et une partie du programme modifiable en FLASH externe)
- on désire utiliser un type de mémoire non-volatile qui n'existe pas en interne.

Citons
- les RAM statiques munies d'une pile au lithium d'une durée de vie 10 ans
- les EEPROM ou E²PROM (Electrically Erasable Programmable Read-Only Memories) : mémoires analogues aux FLASH, mais de technologie plus ancienne et permettant la réécriture individuelle mot par mot

Combinaisons de mémoires programme

- "BOOT LOADER" : partie fixe
  - reset, initialisations, communication externe
  - charge et exécute l'OS (si ∃) puis programme(s) depuis port I/O, ligne série, réseau ou disque
  - technologies
    - ROM (ou µROM interne)
    - FLASH avec "boot block" (protégé en écriture)
- OS et programmes (mise à jour possible)
  - FLASH (↗) ou EPROM (↘)
    - BIOS des µ-ordinateurs
    - programme des "embedded systems" courants
  - RAM
    - programme des µ-ordinateurs
- E²PROM ou RAM+piles
  - paramètres et boîte noire (plutôt DATA)

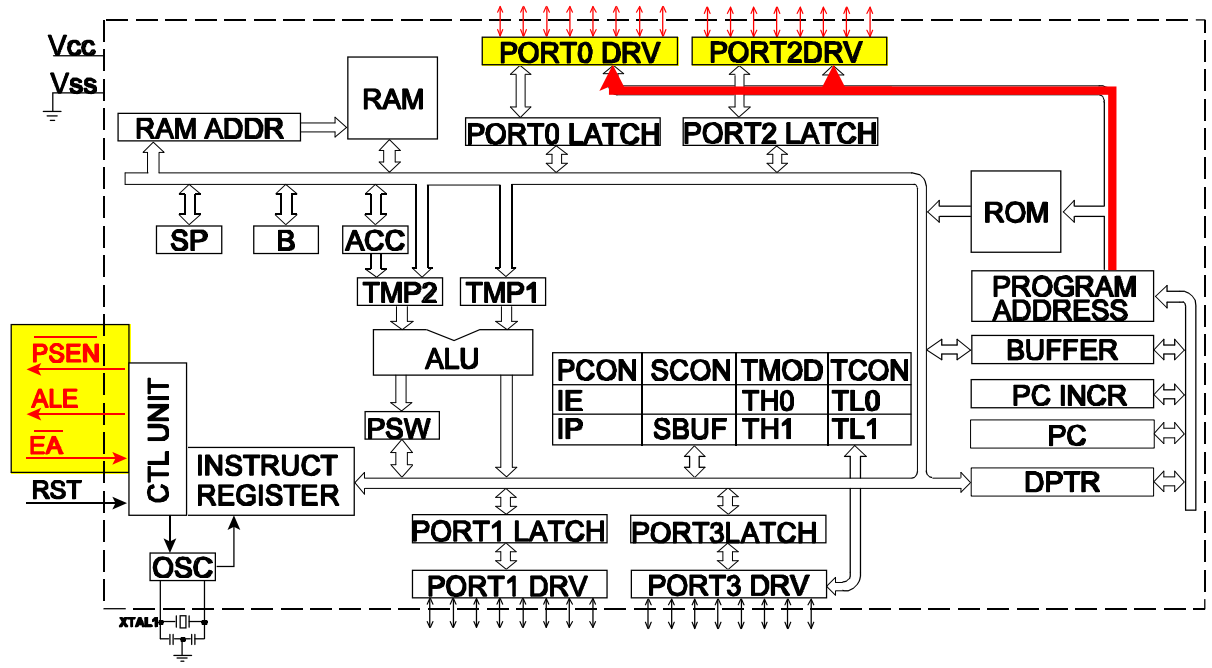
L'hybridation de différents types de mémoires, ou le partage de la mémoire en différents segments présente un intérêt certain.
Le recours à la logique programmée est synonyme de souplesse, ce qui est fort utile pour
- correction de "bugs"
- l'ajouter de nouvelles fonctionnalités en fonction de l'évolution du système ou des besoins

Dans ce cadre, la notion de "***boot loader***" est importante en particulier dans les ordinateurs, où le programme s'exécute depuis la mémoire vive (RAM). En effet, la RAM ne contient rien lorsque l'on met le système sous tension. Il faut alors disposer d'une "boot ROM" pour :

- démarrer le système c'est-à-dire et programmer les différents périphériques nécessaires pour accéder à la mémoire de masse (dont le contrôleur de disque évidemment)
- charger (d'où le nom de "boot loader") le système d'exploitation (OS) du disque vers la RAM. C'est l'OS qui chargera ensuite les programmes en mémoire au moment où l'on doit les exécuter.

Souvent, dans les "embedded systems", il n'y a pas de mémoire de masse. Le programme s'exécute directement depuis la ROM. La notion de "boot loader" n'a alors de sens que pour une mémoire programme reprogrammable électriquement "in situ". Elle devient occasionnelle, car elle n'intervient que pour modifier le programme. Lors de chaque mise sous tension, le programme de "boot" vérifie sur un port de communication (port série, connexion réseau, ...) la présence d'un signal indiquant une demande de mise à jour du programme. Le bloc de ROM contenant ce programme est effacé, puis chargé avec la nouvelle version et le système est redémarré. Il va de soi que la partie du programme exécutée au début du "boot" doit être protégée contre l'effacement, faute de quoi un ennui (comme une coupure de l'alimentation) lors de la mise à jour pourrait rendre le système inopérant.

8051 : mémoire programme externe

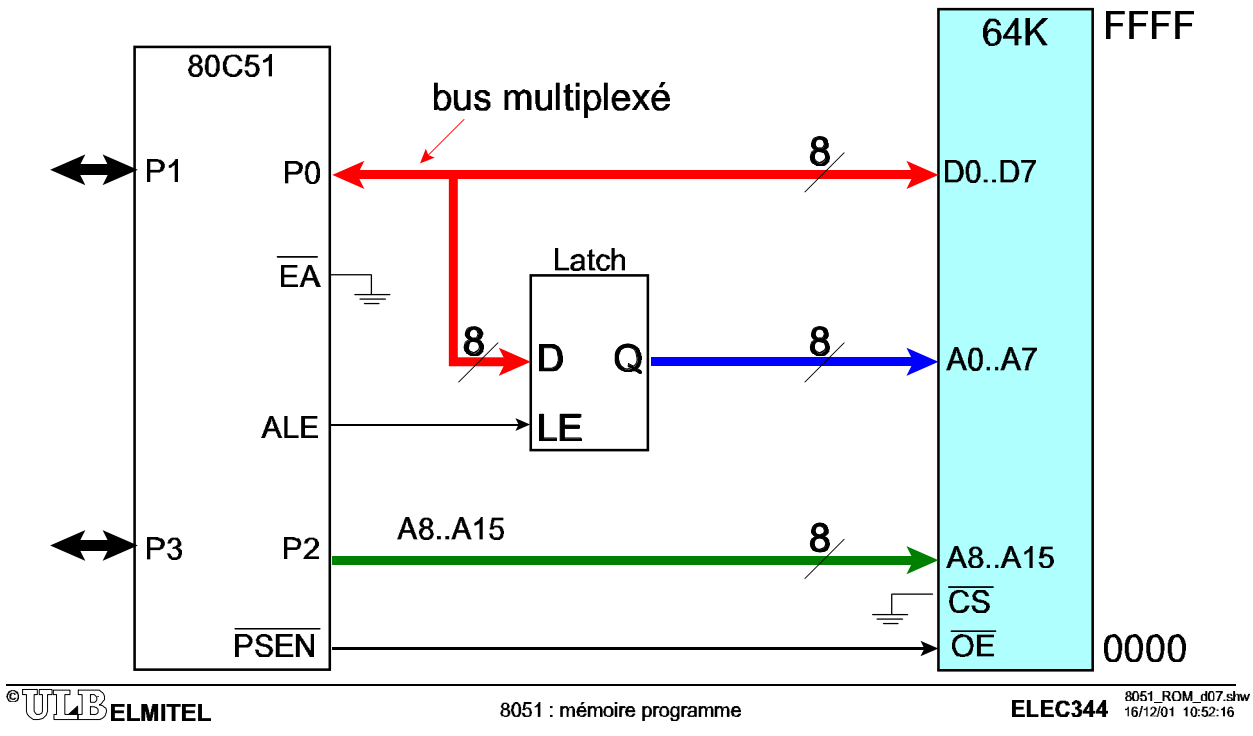


Le recours à des boîtiers de mémoire hors du micro-contrôleur implique la création d'un bus de données et d'un bus d'adresses externes. Dans la famille 8051, on doit sacrifier pour ce faire les deux ports d'entrée-sortie P0 et P2. L'unité de commande (CTL UNIT) est avertie de la présence d'une mémoire programme externe par l'activation du signal EA' (External Address). Elle va alors, à chaque cycle, piloter des signaux supplémentaires constituant le "bus de contrôle":

- PSEN' : "Program Space ENable" signal de lecture en mémoire programme
- ALE : "Address Latch Enable" : signal de démultiplexage des bus (voir dia suivante)

Si le micro-contrôleur ne contient pas de mémoire interne, il boute directement depuis la mémoire externe, qui commence à l'adresse 0. S'il contient de la mémoire interne, alors la mémoire externe vient en complément et le signal PSEN' est activé dès que l'on se réfère à une adresse supérieure à la mémoire interne.
Lorsqu'elle active PSEN, l'unité de commande dirige la sortie du registre "PROGRAM ADDRESS" vers les buffers de sortie (PORT0 DRV et PORT2 DRV). Les registres PORT0 LATCH et PORT2 LATCH sont placés en haute impédance.
Il n'est pas totalement impossible de se servir des ports P0 et P2, mais l'opération est assez alambiquée et en pratique, il faut considérer P0 et P2 comme réservés pour l'accès à la mémoire externe et inutilisables pour des entrées-sorties logiques.

Interface vers mémoire programme externe (boîtier unique)



Le 8051 possède un compteur de programme en 16 bits et un bus de données interne en 8 bits. On a donc accès à une mémoire programme de 64K(=2^{16}) mots de 8 bits.

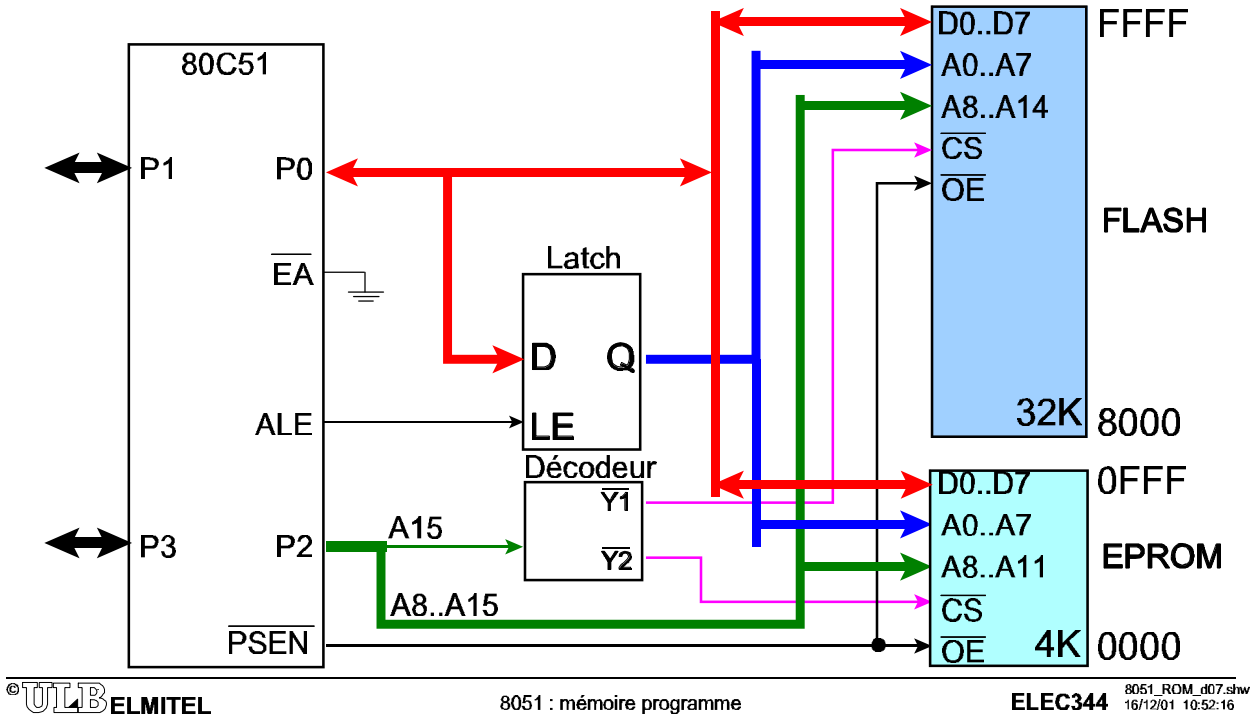
Pour créer des bus de données et d'adresse séparés, il faudrait donc sacrifier 24(=16+8) bornes du boîtier, ce qui devient inacceptable. On a alors recours au multiplexage temporel du bus de données et du bus d'adresse :

- le port P2 est relié directement aux entrées A8 à A15 de la mémoire programme.
- le port P0 est relié indirectement aux entrées A0 à A7 via un D-LATCH et directement au bus de données D0 à D7 de la mémoire

La séquence de lecture suivante est alors réalisée par l'unité de commande :

- placer sur P2 les 8 bits de poids fort de l'adresse et sur P0 les 8 bits de poids faible de l'adresse. PSEN n'étant pas activé, les buffers de sortie de la mémoire sont en état HiZ et il n'y a donc pas de conflit.
- activer brièvement le signal ALE, relié à l'activation du bistable; celui-ci mémorise donc la partie basse de l'adresse. La mémoire ROM peut commencer son décodage d'adresse et le port P0 devient libre pour les données.
- activer PSEN pour libérer les buffers de sortie de la ROM, via OE' (Output Enable)
- diriger l'octet lu soit vers le registre d'instruction (s'il s'agit d'un "opcode fetch") soit vers un autre registre s'il s'agit d'une opérande.

Interface vers mémoire programme externe plusieurs boîtiers(1)



Si l'on veut partager la mémoire programme en plusieurs boîtiers (notamment pour bénéficier des avantages de plusieurs technologies de mémoire), on va rajouter au système un décodeur d'adresse relié aux bits de poids le plus fort du bus d'adresse.
Suivant la combinaison de ces bits, une seule des sorties du décodeur est active et un seul des boîtiers de mémoire est concerné, le (ou les) autre(s) ignorant tous les signaux de contrôle et restant en haute impédance.

Le nombre de bit de poids fort qui doit être relié au décodeur dépend de la quantification désirée pour le positionnement en mémoire des boîtiers.

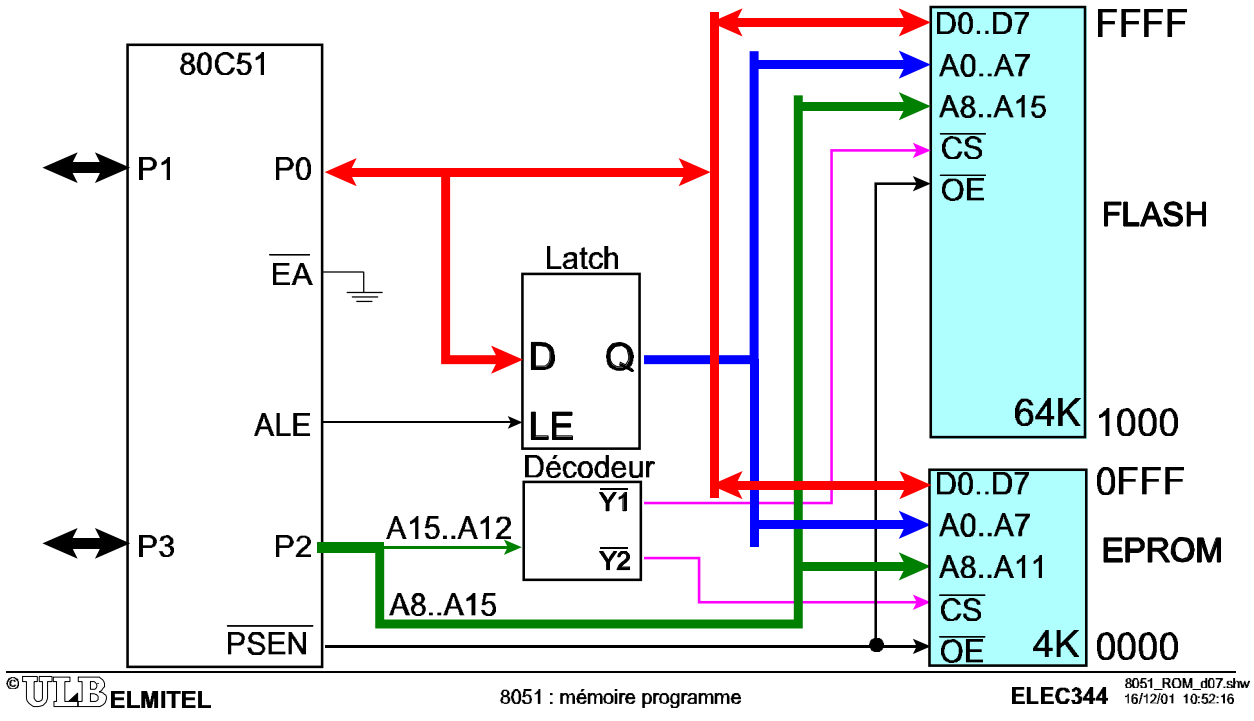
Dans l'exemple de cette figure, on désire placer une EPROM de 4Ko pour bouter le système. elle doit donc être "placée" à l'adresse 0, ce qui veut dire que le décodeur d'adresse doit activer sa sortie Y0', reliée à l'entrée CS'("Chip Select") de l'EPROM pour toute adresse comprise entre 0 et 0FFFH.

Le deuxième boîtier de mémoire est une FLASH de 32 Ko pour la partie modifiable du programme.

Le décodage d'adresse le plus simple consiste à placer la FLASH dans la zone 8000H-FFFFH. Dans ce cas le quantum d'espace d'adresse est de 32Ko. Il suffit de partager les 64Ko en 2 zones égales de 32Ko par un décodeur "1 vers 2" commandé par le bit A15.

Remarquons que le décodage de l'EPROM n'est pas univoque, son entrée CS' sera en effet activée pour toutes les adresses entre 0 et 7FFFH. N'importe quel octet de l'EPROM pourra donc être accédé à 8 adresses différentes. Ce n'est absolument pas gênant; il suffit, lors de la création des programmes, de spécifier à l'éditeur de lien qu'il ne peut pas placer de code dans la zone de 1000H à 7FFFH, où il n'y a pas physiquement de mémoire; le bus d'adresse ne portera donc jamais ces valeurs.

Interface vers mémoire programme externe plusieurs boîtiers (2)



15

Supposons maintenant que l'on dispose d'une FLASH de 64Ko pour placer le programme et que l'on veuille mettre le "boot loader" en EPROM pour pouvoir (re)programmer la FLASH.

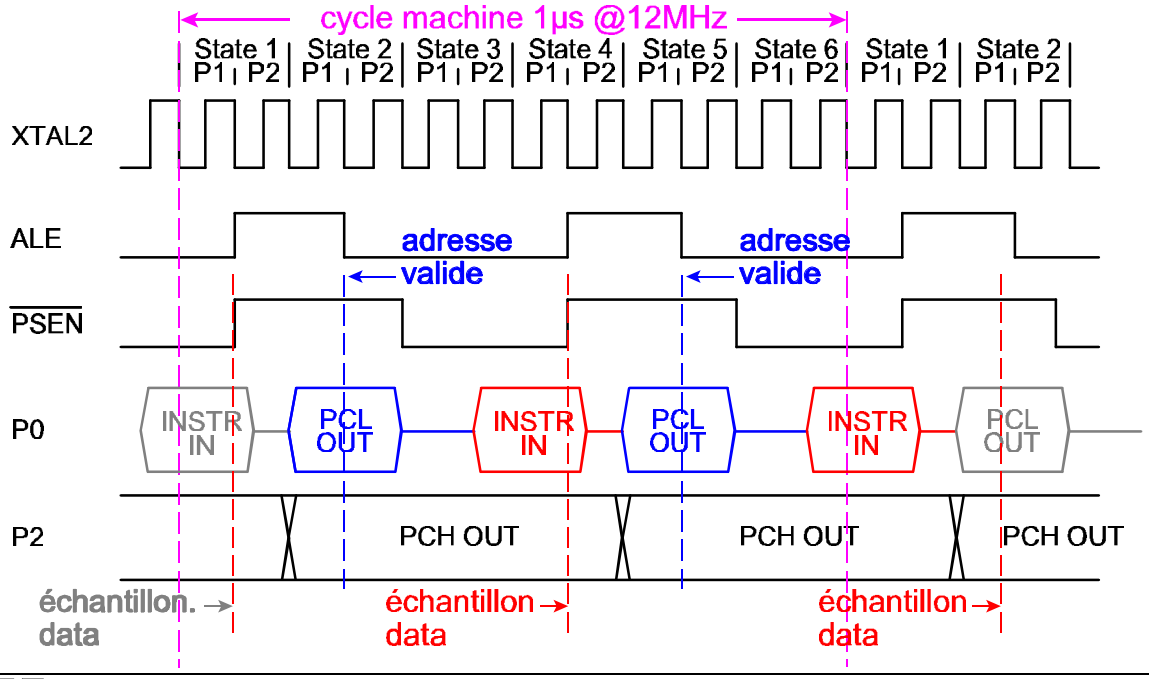
Il s'agit donc de "voler" les 4 premiers Ko à la FLASH en sélectionnant l'EPROM à sa place lorsque l'adresse est comprise entre 0 et 0FFFH.
Le quantum de partage de l'espace d'adresse est maintenant de 4Ko soit 1/16ème de l'espace d'adresse. Le décodeur doit dont travailler sur les 4 bits de poids fort (A15-A12) pour partager l'espace d'adresse en 16 blocs, dont 1 est affecté à l'EPROM et 15 à la FLASH.

Le décodeur sera réalisé soit
- à l'aide de portes (exercice : écrire les équations logiques et synthétiser ce décodeur)
- à l'aide d'un décodeur 4 vers 16 en circuit intégré, dont les sorties Y1' à Y15' sont rassemblées par un ET logique
- à l'aide d'une logique programmable (PLD, CPLD) ce qui est généralement le plus économique en place et en coût. (voir le chapitre sur logique programmable); il suffit alors d'écrire les équations logiques, puis de placer le décodeur sur un programmateur adéquat pour lui conférer les propriétés désirées.

On insistera sur le caractère mutuellement exclusif des sorties du décodeur : seul un boîtier peut sortir de l'état haute impédance, sous peine de conflit sur le bus de données.

La division de la mémoire en zones disjointes affectées aux différents boîtiers est le "***memory mapping***" ou cartographie de la mémoire.

Opcode Fetch en mémoire externe

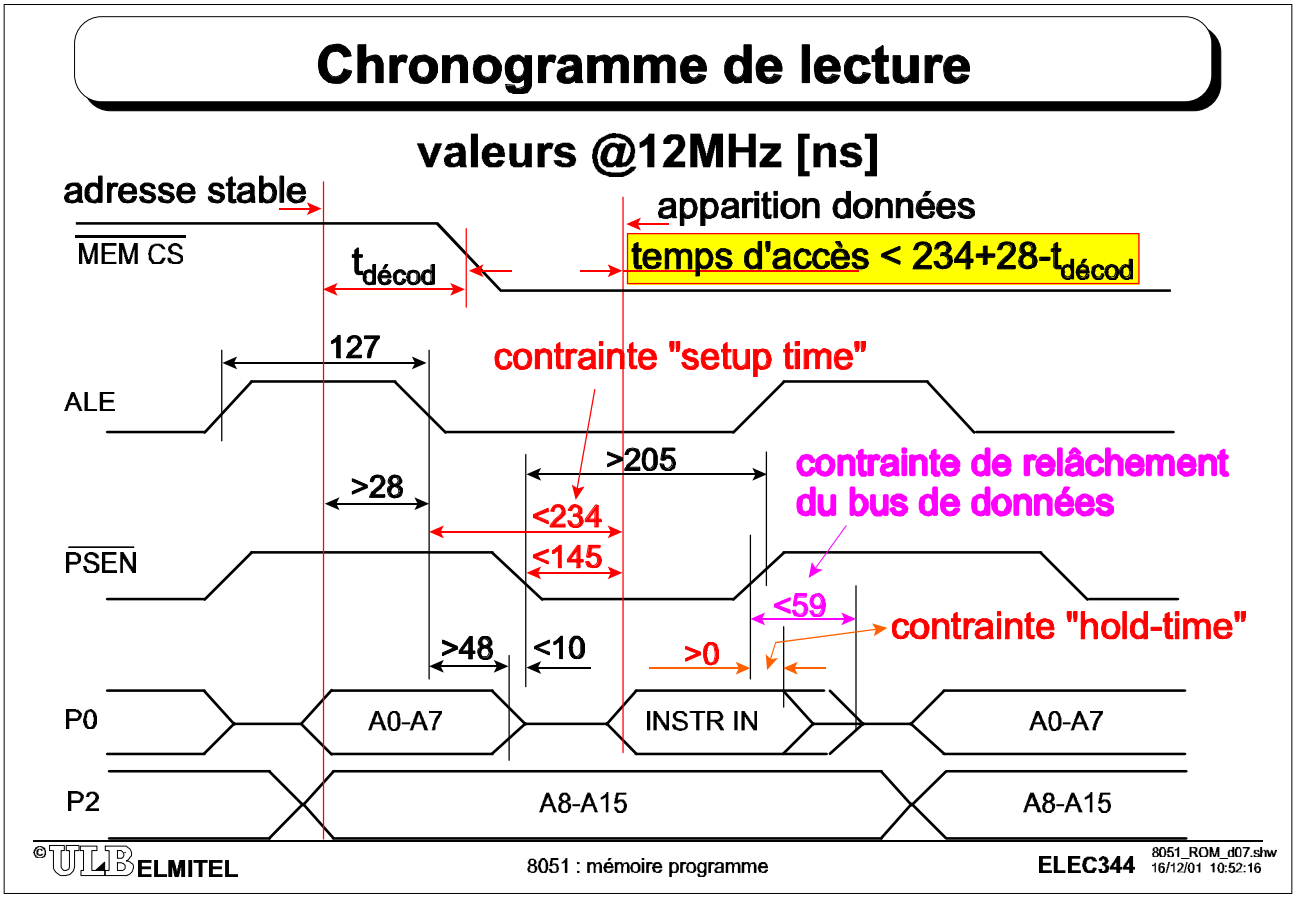


Le multiplexage adresses/données et le séquencement des différents signaux de contrôle se voient clairement sur cette figure.

Le cycle machine d'extraction ou ("op-code fetch") prend 12 coups d'horloge partagés en 6 états ou stades, ce qui montre bien le caractère séquentiel de l'unité de commande des micro-processeurs :

- au stade 1, on termine la lecture précédente en mémoire programme. Le signal ALE devient actif au milieu du stade 1, pour que la sortie du latch de démultiplexage recopie l'adresse basse correspondant à la nouvelle lecture
- au début du stade 2, on place la partie basse du compteur de programme (PCL) sur le port P0 et la partie haute (PCH) sur le port P2. Au milieu du stade 2, ALE est désactivé, et le LATCH de démultiplexage mémorise la partie basse de l'adresse jusqu'au cycle suivant
- au début du stade 3, PSEN est activé pour que les buffers de sortie de la mémoire quittent leur état HiZ
- pour laisser à la mémoire le temps de faire son décodage d'adresse et d'extraire la donnée de la matrice (ce qui définit son "temps d'accès") l'unité de commande du 8051 attend le milieu du stade 4 pour échantillonner les données et les transférer en interne.

Le 8051 effectue deux lectures en mémoire programme par cycle machine. Dans certains cas, le résultat de la deuxième lecture est ignoré (voir chapitre sur le jeu d'instruction).



Cette figure présente un chronogramme typique pour un 8051 cadencé à 12 MHz.
Les temps (en ns) comprennent en général une partie fixe et une partie dépendant de la fréquence d'horloge (par exemple 25ns+3 périodes d'horloge) et ne peuvent donc pas être simplement multipliés ou divisés par une constante si l'on change la fréquence d'horloge.

Les différents délais sur cette figure appartiennent à 2 catégories:
- une indication sur les instants des transitions des signaux issus du processeur et sur l'instant d'échantillonnage de données (en général dans le cas le plus défavorable)
- les contraintes sur les délais liés à la mémoire et au décodeur d'adresse

Grâce au fait que l'on active ALE avant le changement d'adresse et que le latch est ainsi rendu transparent, le décodeur peut commencer à travailler dès que l'adresse est stable, c'est-à-dire au moins 28ns avant la désactivation de ALE. Après le temps de propagation du décodeur, le CS' de la mémoire est activé et celle-ci commence son propre décodage interne pour activer les lignes de mot et de bit correspondant à l'adresse actuelle.
La mémoire doit présenter les données sur le bus au plus tard 234ns après la désactivation de ALE et 145ns après l'activation de PSEN', faute de quoi le "setup time" ne sera pas respecté. Le temps d'accès de la mémoire doit donc être inférieur à 234ns+28ns-$t_{décod}$.

L'échantillonnage des données par le 8051 se produit avant que PSEN' ne soit désactivé, aussi le "hold-time" peut-il être nul.

La dernière contrainte est que la mémoire doit repasser en HiZ au plus tard 59ns après la désactivation de PSEN', pour éviter la contention de bus (voir plus loin).

Temps d'accès insuffisant : que faire ?

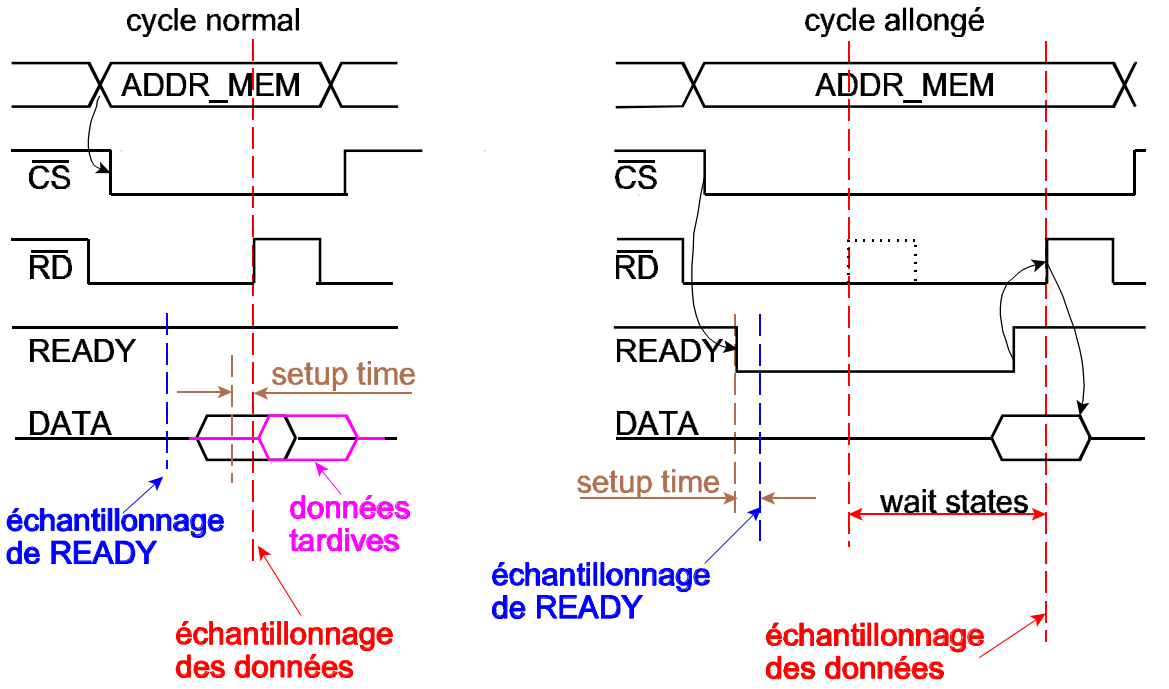
- 8051: aucune action possible
  - changer de type de mémoire
  - peu probable aux fréquences usuelles
- la plupart des autres µP
  - "wait states" : action de retardement du signal de lecture
    - "synchrone" : le µP n'attend que si on désactive le signal "READY"
    - "asynchrone" : le µP attend par défaut et la mémoire doit fournir un signal d'acquittement qui provoque la lecture

Le temps d'accès des mémoires actuelles est de l'ordre de quelques dizaines de ns et donc le temps offert par un 8051 à 12MHz est plus que suffisant.
Pour des versions à cadence plus élevée, les mémoires les plus lentes pourraient avoir un temps d'accès trop long, c'est-à-dire présenter les données sur le bus après l'instant d'échantillonnage par le processeur. Dans ce cas le 8051 la seule solution est d'adopter une mémoire plus rapide.

La plupart des autres processeurs offrent une possibilité d'utiliser des mémoires plus lentes. L'instant d'échantillonnage doit être retardé via une borne spéciale du processeur appelée READY. On ajoute alors des états d'attente supplémentaires (d'une durée généralement égale à la période d'horloge) appelés "***wait states***".

- avec les processeurs dits "synchrones", la ligne READY est active par défaut; c'est le circuit de décodage qui doit provoquer la désactivation de READY au cas où l'on accède à une zone d'adresse affectée à une mémoire lente
- avec les processeurs dits "asynchrones" le processeur attend d'office, jusqu'à ce que le décodage d'adresse fournisse un signal indiquant au processeur que les données sont prêtes à être échantillonnées.

Wait states en mode synchrone



L'action d'une borne de type READY sur un processeur synchrone est illustrée sur cette figure.

Le cycle normal est présenté à la figure de gauche. Le cycle commence par l'apparition de la nouvelle adresse. Après le temps de propagation du décodeur, celui-ci active le CS' de la mémoire. L'unité de commande laisse un délai raisonnable au décodeur avant d'échantillonner le signal READY. Celui-ci étant actif, l'unité de commande présume que la mémoire répondra dans les délais prescrits et provoque la remontée du signal de lecture RD', sur laquelle les données sont échantillonnées. Pour être lue avec certitudes les données doivent être stables un "setup time" avant la remontée de RD'.

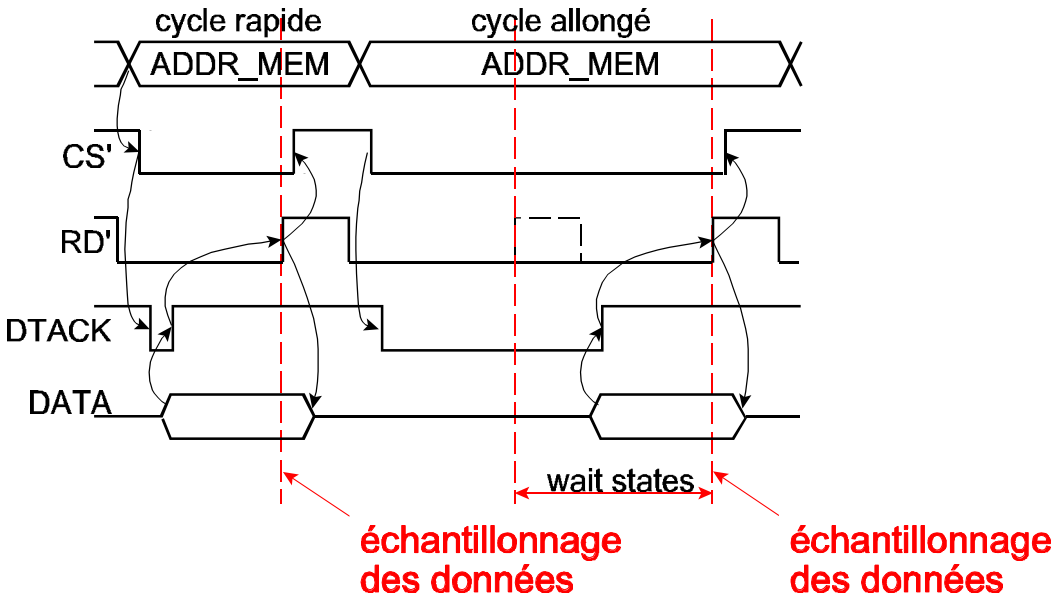
Si cette condition n'est pas respectée, la lecture est ratée et les données n'ont pas de signification, ce qui empêche le système de fonctionner. En effet, dans le cas d'une lecture en mémoire programme le décodeur d'instruction sera
- soit incapable de faire son travail parce que l'instruction erronée n'existe pas
- soit entamera l'exécution d'une fausse instruction

Dans le cas d'une lecture dans une mémoire de données, celles-ci seront invalides, ce qui n'est pas mieux.

Dans le cycle de droite, l'unité de commande échantillonne le signal READY comme inactif au début du cycle, qui est alors rallongé d'un "wait state" correspondant généralement à la durée d'une période d'horloge. Durant le "wait state", READY est rééchantillonné et l'attende se poursuit jusqu'à ce que le READY soit vu actif; à ce moment l'unité de commande termine le cycle par la remontée du signal RD'.

REM : il existe souvent un nombre maximum de wait states, car
- ce mécanisme peut empêcher le rafraîchissement des registres internes dynamiques
- on veut éviter le blocage du système et on considère qu'il existe une limite à la lenteur des mémoires

Wait states en mode asynchrone



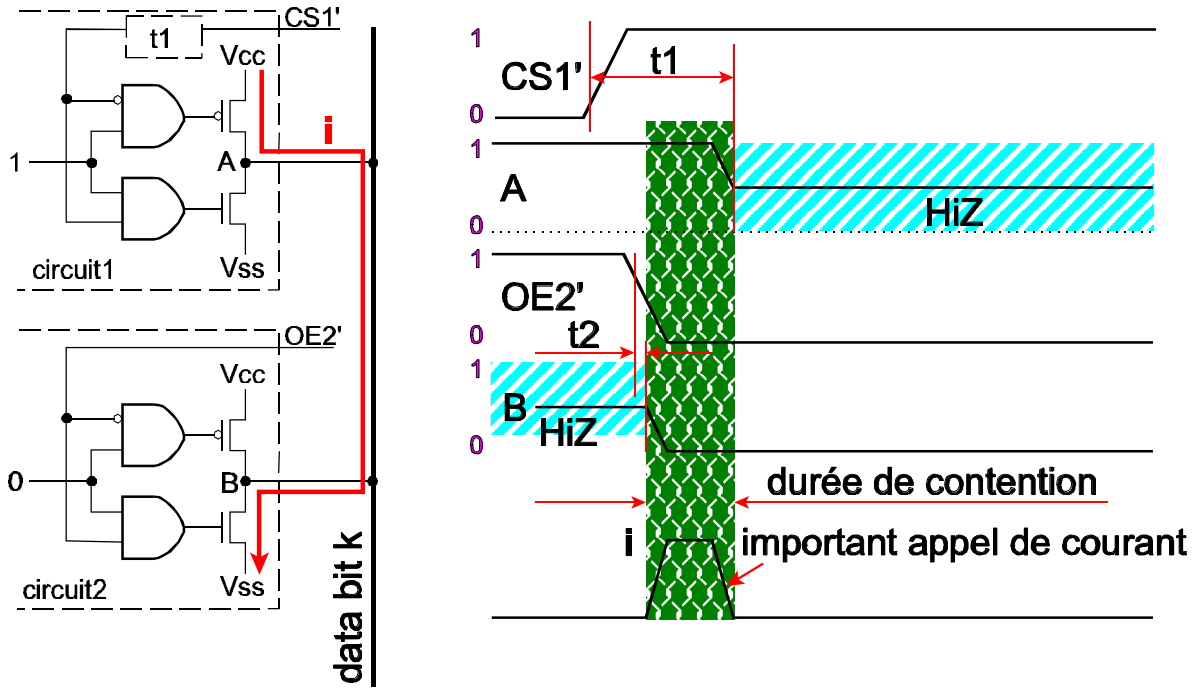
Dans les processeurs asynchrones, un signal DTACK (Data Transfer ACKnowledge) indique au processeur qu'il peut échantillonner les données.

Le décodeur d'adresse active la ligne CS' (Chip Select) de la mémoire, ce qui provoque la désactivation du signal DTACK pendant un délai qui dépend du temps d'accès de la mémoire. Dans le cycle de gauche, DTACK remonte presque immédiatement et le cycle du processeur prend la durée minimum. Dans le cycle de droite, l'allongement de l'inactivité de DTACK retarde d'autant l'échantillonnage des données.

La durée du DTACK peut se régler par un monostable déclenché par le CS' ou par un compteur qui compte un certain nombre de coups d'horloge. Le plus compact est de combiner le décodeur d'adresse et le générateur de "wait states" dans une logique programmable (PLD).

Certains micro-contrôleurs contiennent un décodeur d'adresse programmable, avec générateur de "wait states" incorporé. Le programme boute sur un cycle "ultra-long" permettant de s'accommoder de la mémoire la plus lente; dans les quelques premiers mots extraits de la mémoire programme figurent tous les paramètres du décodeur (frontières et nombres de wait-states associés à chaque zone de la mémoire).

Contention de bus - intérêt de l'Output Enable



La présence d'une borne de commande directe de l'état de sortie en haute impédance appelée OE' (OUPUT ENABLE) est justifiée par le souci d'éviter la "contention de bus".

Considérons deux circuits connectés au même bus de données. Supposons que le circuit 1 ne soit pas muni d'une borne OE'. C'est la désactivation du Chip Select (CS') qui est indirectement responsable du passage en haute impédance, après un délai t1.
Supposons que le circuit 2 sorte de l'état haute impédance suite à l'activation de sa borne OE2' avant que le circuit 1 ne l'ait quitté.

Si les deux états de sortie ne sont pas identiques, il en résulte un court-circuit de l'alimentation à travers deux transistors de sortie pendant le temps du recouvrement.

La présence d'une borne OE permet de minimiser le délai pour repasser en haute impédance.

Lecture en mémoire programme : résumé

- décodeur
  - faire en sorte qu'il puisse travailler le plus tôt possible pour augmenter le temps d'accès
  - on peut faire remonter le CS' sur la fin de la lecture
- Output Enable
  - toujours connecter au signal de lecture pour passer en tri-state dès la fin du cycle de lecture
- vérifier tous les timings dans le cas le plus défavorable
  - tension
  - température
  - .....