Architecture des machines

Jean-Baptiste.Yunes@univ-paris-diderot.fr
Université Paris Diderot
http://www.liafa.univ-paris-diderot.fr/~yunes/

Octobre 2012

Le modèle en couches

Une couche (*layer*) en informatique désigne un niveau d'abstraction.

> *Par exemple : l'utilisateur d'une calculatrice ne se préoccupe pas de savoir comment les manipulations qu'il effectue sont effectivement réalisées (comment un nombre est-il représenté dans la machine ?, quel mécanisme est mis en œuvre pour afficher ce nombre à l'écran ?, etc).*

Note : l'abstraction est une pratique courante (*i.e.* de tous les jours) ; lorsqu'on conduit une voiture on ne se préoccupe pas des mécanismes réellement mis en œuvre pour obtenir son fonctionnement.

Le modèle en couches

Les couches qui peuvent nous préoccuper dans le cadre de ce cours sont :

5	Langage de programmation
	Compilation
4	Langage d'assemblage
	Assemblage
3	Système d'exploitation
	Interprétation
2	Machine
	Interprétation
1	Micro-programme
	Exécution matérielle
0	Matériel

Le modèle multi-couches

La couche matérielle

C'est une couche assez complexe car elle fait le lien entre les aspects logiques et physiques de l'informatique.
La couche matérielle pourrait être elle-même découpée en différents niveaux d'abstraction.
L'informaticien se contente généralement de l'aspect logique de cette couche (lorsqu'il s'y intéresse).
Dans ce cadre on y voit essentiellement des portes logiques connectées entre elles et pilotées par une horloge.
Il n'y a pas ici de notion de programme, mais de quoi les exécuter. Des fonctions sont calculées, et les combinaisons matériellement possibles de celles-ci permet d'envisager l'exécution de programmes exprimés au niveau supérieur.

Le modèle multi-couches

Le couche micro-programmée

Cette couche peut ne pas exister. Si elle existe, le processeur est dit micro-programmé.
Dans ce niveau se trouve un logiciel/un programme (le micro-programme).
Ce programme a pour rôle d'interpréter les instructions de la couche immédiatement supérieure en pilotant la couche matérielle de façon à exécuter les instructions.
Il est chargé d'interpréter les instructions en provenance de la couche supérieure.
Certains processeurs ne possèdent pas cette couche. Les instructions du niveau supérieur sont alors exécutées directement par le niveau matériel.

Le modèle multi-couches

La couche machine

Le manuel de référence d'un processeur décrit ce niveau-ci.
Cette couche définit le jeu d'instruction du processeur.
Ce jeu est découpé en différents type d'instructions :

- entrées/sorties (lecture/écriture mémoire pour simplifier) ;
- calcul (arithmétiques, logiques) ;
- contrôle (saut, test).

Le modèle multi-couches

Le système d'exploitation

Cette couche est constituée d'un ensemble logiciel permettant de rendre la vie pus facile aux utilisateurs de la machine (simples utilisateurs ou programmeurs).
Les particularités physiques de la machine y sont cachées, et tout particulièrement en ce qui concerne les périphériques.
La manière de les rendre accessible les différents objets de façon uniforme définit le système.

Le modèle multi-couches

La couche d'assemblage

Cette couche concerne les programmeurs.
C'est à partir de ce niveau que l'on peut réaliser des applications, même si ce niveau n'est que très rarement employé dans ce cadre. En effet, la programmation à ce niveau est encore bien désagréable.
Le langage de programmation employé est appelé langage d'assemblage ou assembleur.

Le modèle multi-couches

Les langages de programmation

À ce niveau, la programmation est facilitée par emploi d'un langage structuré de sorte que l'expression du calcul souhaité soit assez naturelle (paradigmes de programmation).
La traduction de ce langage (dit de haut-niveau) est réalisée par l'emploi d'un compilateur : un programme capable de traduire un programme dans un langage de haut-niveau en un programme assembleur correspondant.
Le programmeur est débarrassé de nombreuses contingences matérielles.
On y trouve des langages comme ADA, ALGOL, APL, BASIC, C, C++, CAML, COBOL, FORTRAN, HASKELL, LISP , Pascal, PL/1, etc.

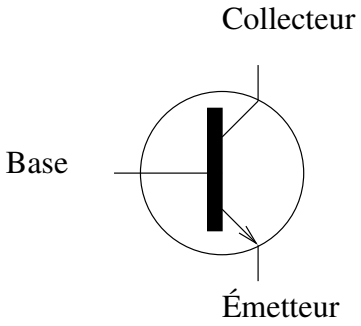
Thèmes de réflexion

- retrouver diverses description en couches utilisées en informatique (couches logicielles, couches matérielles, couches réseau, etc)
- dans le modèle en couche évoqué, retrouver les variantes (langages de programmation, systèmes, technologie, etc)
- retrouver les architectures micro-programmée et celles qui ne le sont pas
- distinguer la notion de compilation et d'interprétation
- retrouver les différents paradigmes de programmation et y classer au moins deux langages.

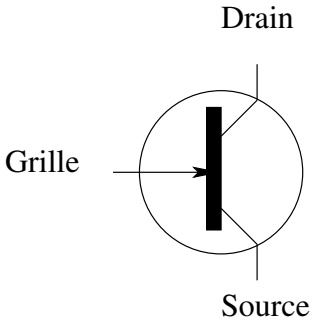
La couche matérielle

Le transistor

Un dispositif utilisé en informatique comme interrupteur (de façon plus générale il peut aussi être employé comme amplificateur ou stabilisateur de signal).
Deux technologies.
Le transistor bipolaire (TTL) :

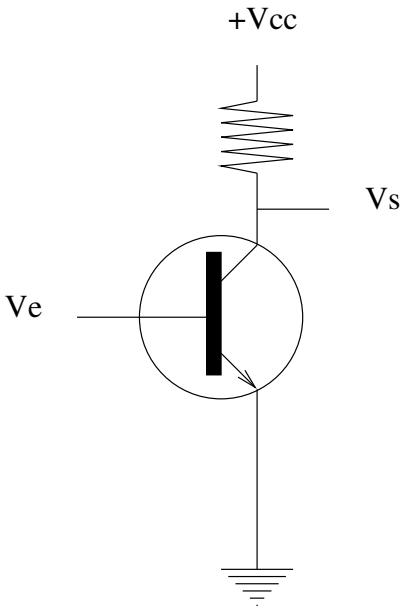


Le transistor à effet de champ (MOS) :



La couche matérielle

Une porte de base NON (bipolaire)



Rappel : un transistor peut se comporter comme un interrupteur, si une tension suffisante est présente sur la base alors le circuit entre le collecteur et l'émetteur est fermé, sinon il est ouvert.
Fonctionnement de la porte non : Lorsque V_e est inférieure à une valeur critique le transistor se ≪ bloque ≫ et se comporte comme un interrupteur ouvert (très grande résistance). Donc V_s est très proche de V_{CC}.
Si V_e dépasse la valeur critique le transistor se ≪ ferme ≫ et l'interrupteur se ferme (résistance quasi-nulle). Donc V_s est très proche de la masse.
On peut noter que V_s est en ≪ haut ≫ lorsque V_e est en ≪ bas ≫ et réciproquement.
Il s'agit d'un inverseur ou porte logique NON.

Cette construction a un défaut majeur :

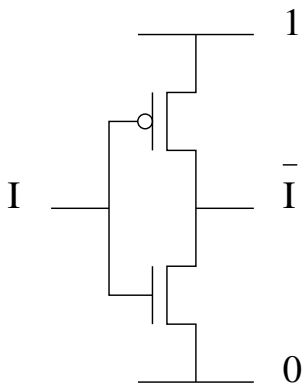
- si l'entrée est basse, pas de courant traversant donc pas de consommation,
- sinon avec l'entrée haute, du courant traverse et donc une consommation induite.

La dissipation est importante, et trop importante lorsque l'intégration est grande. Inutilisable en pratique pour de fortes intégrations.

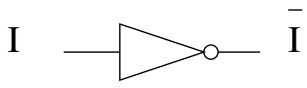
La couche matérielle

Une porte de base NON (effet de champ et C-MOS)

Même fonctionnement mais le transistor existe en deux versions : PNP, NPN.
Dans la version PNP, cela fonctionne comme pour le transistor bipolaire.
Dans la version NPN, c'est l'inverse, si la grille n'est pas sous tension le transistor se ferme et la sortie est donc proche de la masse ; si la grille est sous tension le transistor s'ouvre et la sortie est donc proche de la tension d'entrée.
On couple alors un PNP et un NPN :

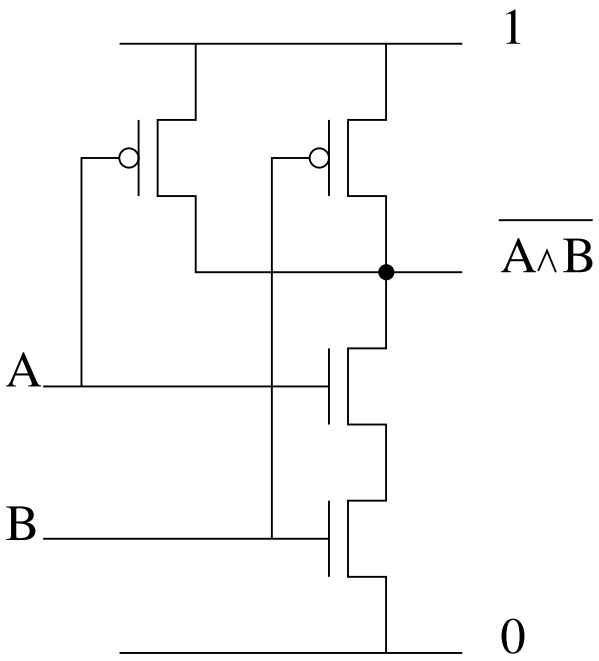


Dans le cas idéal, il n'y a pas de consommation... Deux fois plus de transistors...
Le symbole représentant la fonction logique est :

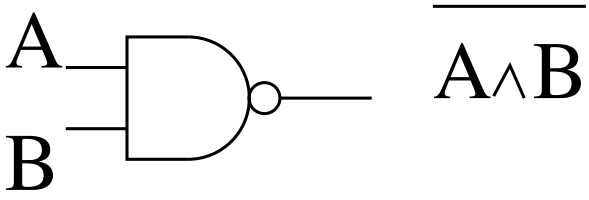


La couche matérielle

La porte NON-ET (C-MOS)



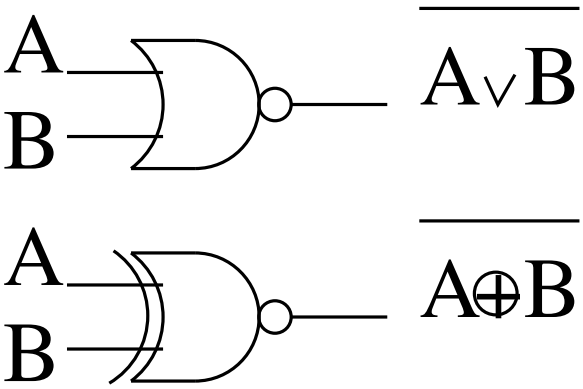
Le symbole représentant la fonction logique est :



La couche matérielle

Les portes NON-OU et NON-OU-EXCLUSIF

Les symboles correspondants sont :



La couche matérielle

Les tables de vérité

A	$\overline{A}$
0	1
1	0

A	B	$\overline{A \wedge B}$
0	0	1
0	1	1
1	0	1
1	1	0

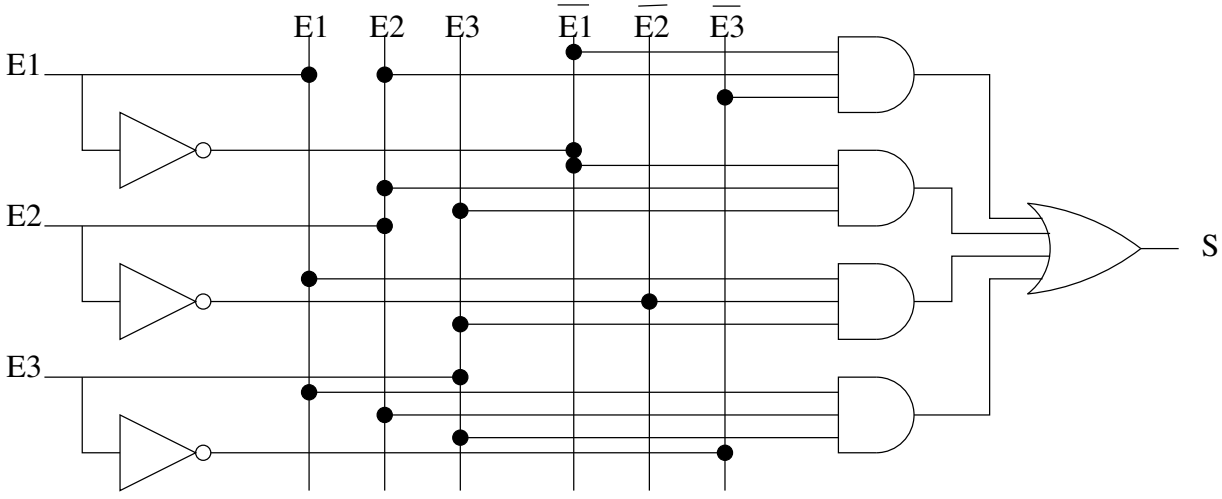
A	B	$\overline{A \vee B}$
0	0	1
0	1	0
1	0	0
1	1	0

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Réalisation d'une fonction booléenne

E1	E2	E3	S
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



La couche matérielle

Les familles d'intégration

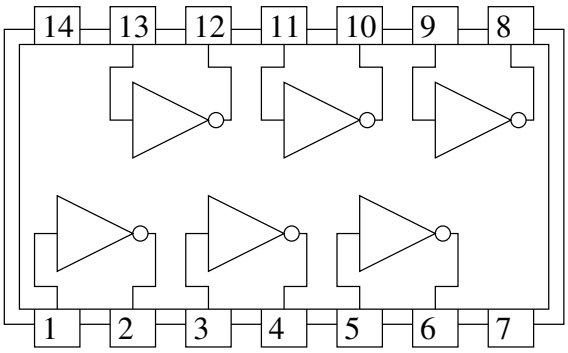
Il en existe quatre familles. Le classement est effectué en fonction de la densité d'intégration (nombre de portes ou de transistors par circuit ou unité de surface) :

- **SSI** : Small Scale Integration. Quelques dizaines de portes par circuit.
- **MSI** : Medium Scale Integration. Quelques centaines portes par circuit.
- **LSI** : Large Scale Integration. Quelques dizaines de milliers de portes par circuit.
- **VLSI** : Very Large Scale Integration. Quelques centaines de milliers de portes par circuit.
- **ULSI** : Ultra Large Scale Integration. Au-dessus du millions de portes.

La couche matérielle

Les circuits logiques

Ils contiennent en général plusieurs portes identiques et indépendantes. Lesquelles réalisent une fonction logique déterminée.
Voici le 7404, circuit TTL (Transistor-Transistor Logic) de la famille 7400 de chez Texas Instruments :



La couche matérielle

Les circuits combinatoires

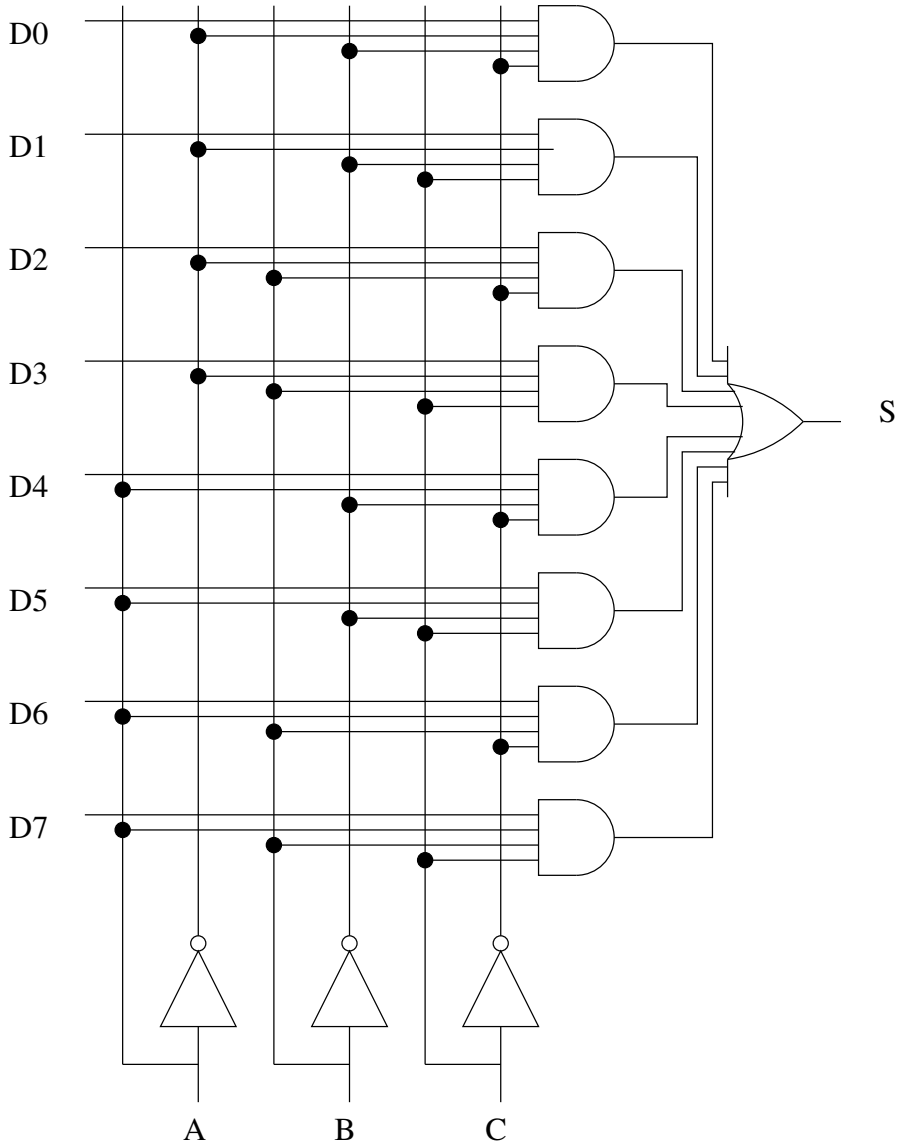
Un circuit combinatoire est une combinaison de portes élémentaires réalisant le calcul d'une fonction sans faire intervenir le temps sauf celui de la traversée du circuit (pas de retour arrière - *feedback*).

La couche matérielle

Le multiplexeur

Il possède 2^n entrées à sélectionner (e_i), n entrées de sélection (un nombre x) et 1 sortie (s) :

$$s = e_x$$



On peut réaliser un convertisseur parallèle/série à l'aide d'un multiplexeur :
Si l'on suppose qu'un compteur fournit à intervalles de temps réguliers les 2^n valeurs possibles dans la partie sélection, et qu'en permanence une valeur est présente dans la partie à sélectionner, alors en sortie on dispose de chacune des entrées, au fur et à mesure de l'écoulement du temps.

La couche matérielle

Le démultiplexeur

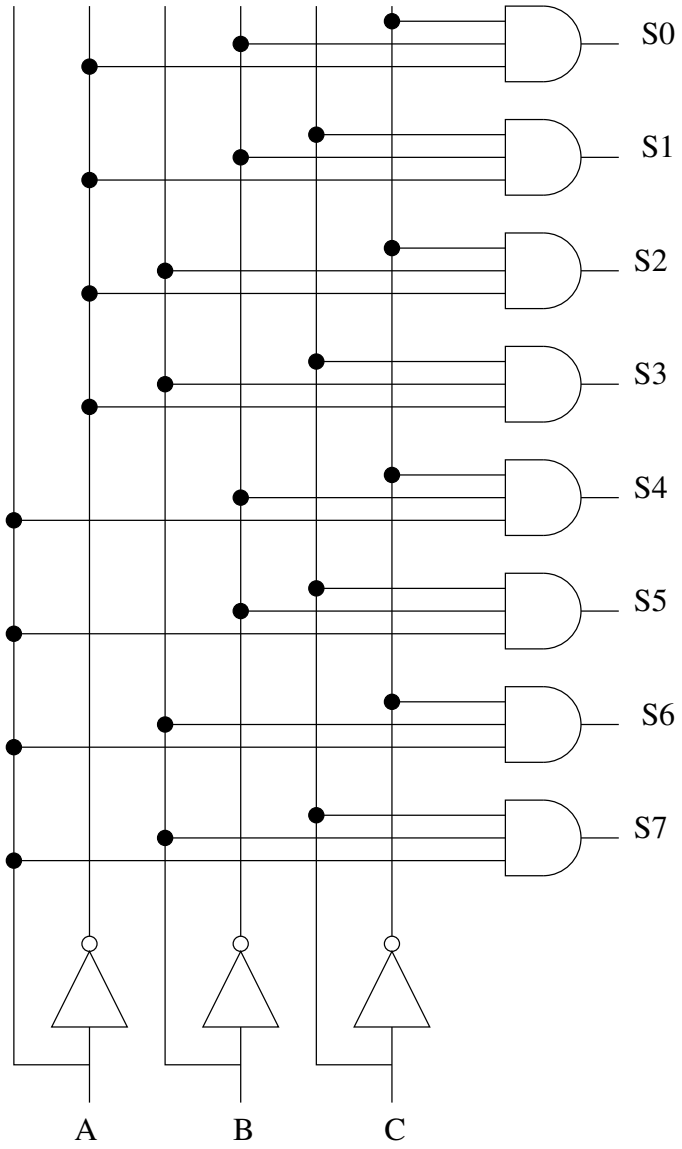
Il existe aussi un circuit appelé démultiplexeur qui réalise exactement l'inverse. On lui fournit une entrée de donnée e, et une valeur sur n entrées de sélection (un nombre $x \in [0, 2^n[$). L'une des 2^n sorties contient la valeur en entrée :

$$s_x = e, \forall i \neq x, s_i = 0$$

La couche matérielle

Le décodeur

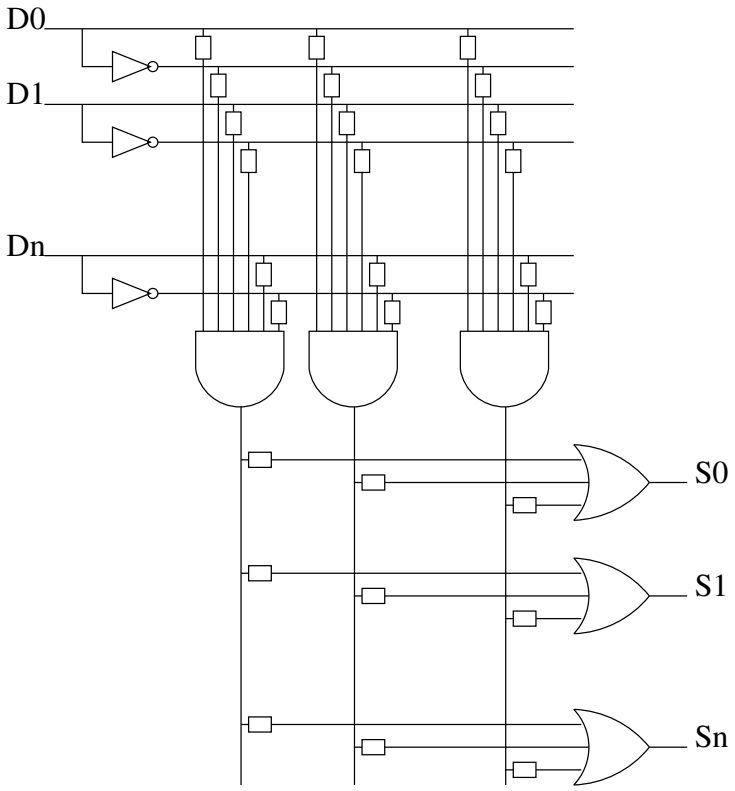
Il possède n lignes d'entrées (un nombre x) et 2^n lignes de sortie (s_i).



Il permet par exemple la sélection de boîtiers mémoire (entre autres).

La couche matérielle

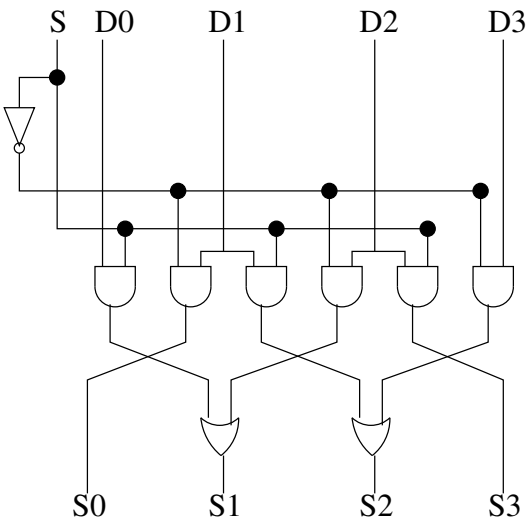
Les réseaux programmables



Les fusibles sont conducteurs. La programmation est réalisée (par l'intermédiaire d'un dispositif qui n'est pas apparent) en brûlant les fusibles. Cela permet de couper les connexions et donc de programmer les sorties désirées.
Les fusibles peuvent être reversibles.

La couche matérielle

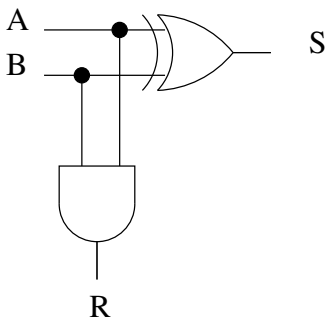
Le décaleur



Selon la valeur de S, les *n* entrées sont décalées vers la gauche ou vers la droite. Dans cet exemple, le bit *entrant* est un zéro.

La couche matérielle

Le demi-additionneur

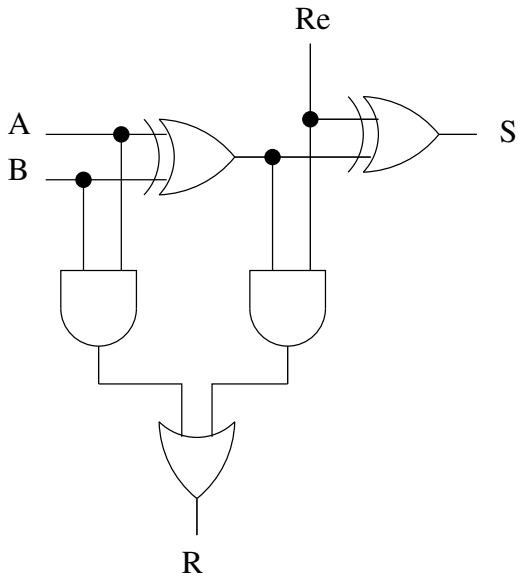


Il permet d'additionner deux bits et de calculer la retenue éventuelle.
Malheureusement il n'est pas capable de prendre en compte la retenue précédente.

La couche matérielle

L'additionneur

Il est composé de deux demi-additionneurs couplés de sorte qu'une retenue soit incluse dans le calcul :

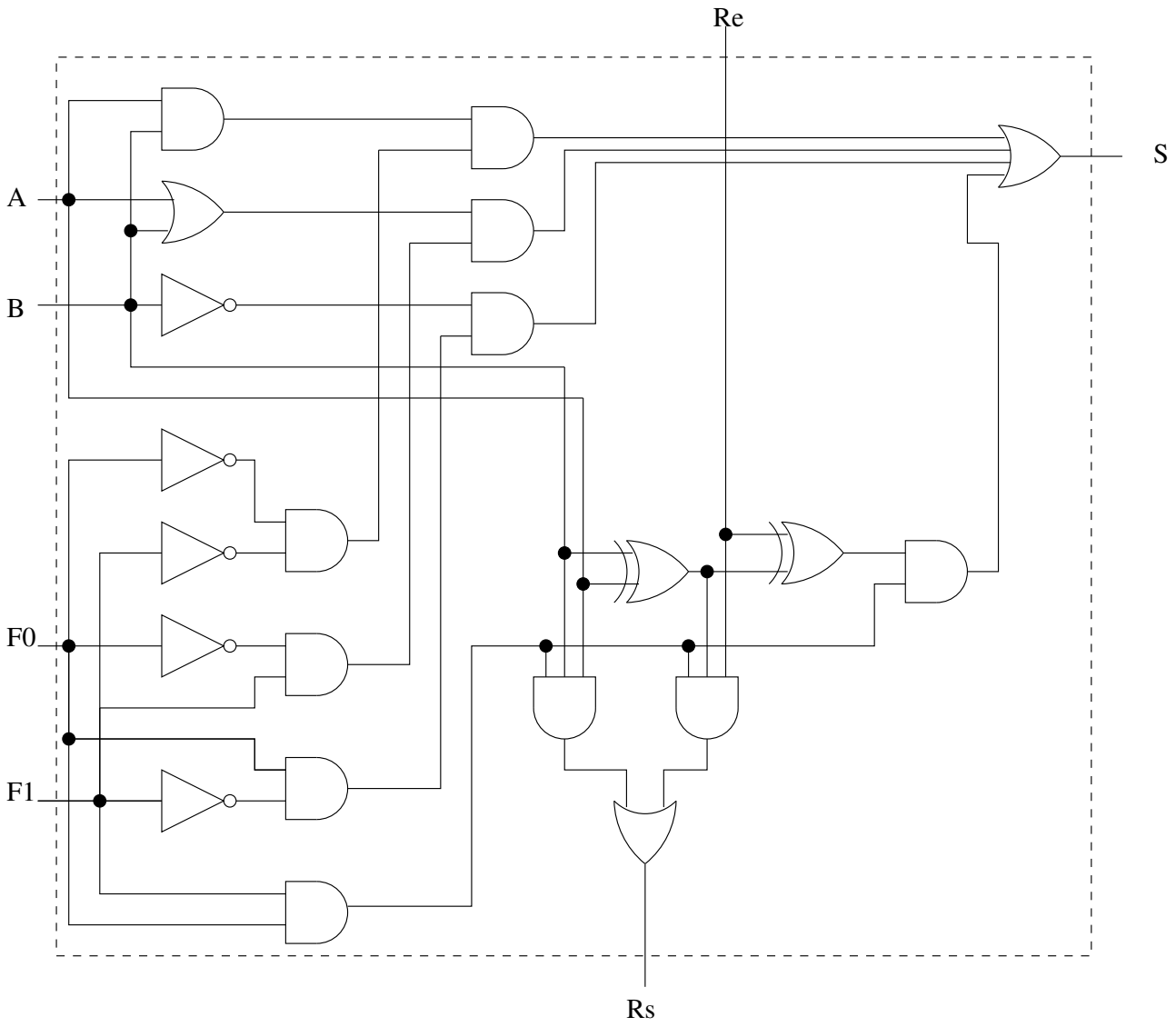


Il est utilisé pour réaliser un additionneur complet. Ce dispositif est dit à propagation de retenue.
Il existe d'autres techniques notamment celle appelée à anticipation de retenue.

La couche matérielle

L'Unité Arithmétique et Logique

C'est une partie importante des ordinateurs, puisqu'elle permet de réaliser au niveau physique différentes fonctions sur des nombres d'une taille donnée.
Elle est composée de deux parties : l'unité logique et arithmétique et l'unité de décodage.
L'unité logique et arithmétique se propose de calculer les différentes fonctions logiques que l'on a décidé de fournir en sortie.
L'unité de décodage s'occupe de sélectionner l'une des fonctions possibles calculées par l'unité logique et arithmétique en validant la sortie correspondante.



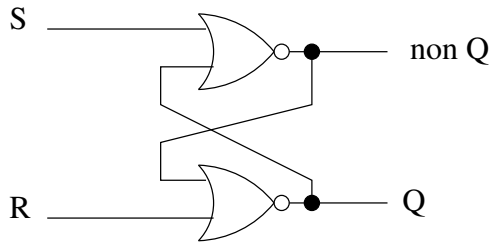
Les circuits séquentiels

Ce sont des circuits dans lesquels le temps intervient.
La sortie de ces circuits est fonction des entrées et du temps.
Ils peuvent être dotés d'une mémoire.
Cette mémoire est obtenue par utilisation d'un bouclage arrière (*feedback*).

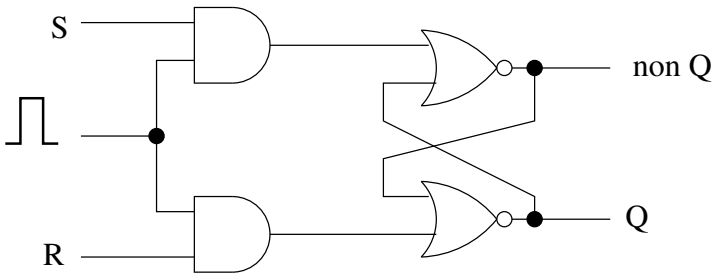
La couche matérielle

La bascule RS

Ce circuit possède deux entrées R et S.
S pour Set (mise à 1) et R pour Reset (mise à zéro).
Il y a deux sorties. L'une étant toujours complémentaire de l'autre.



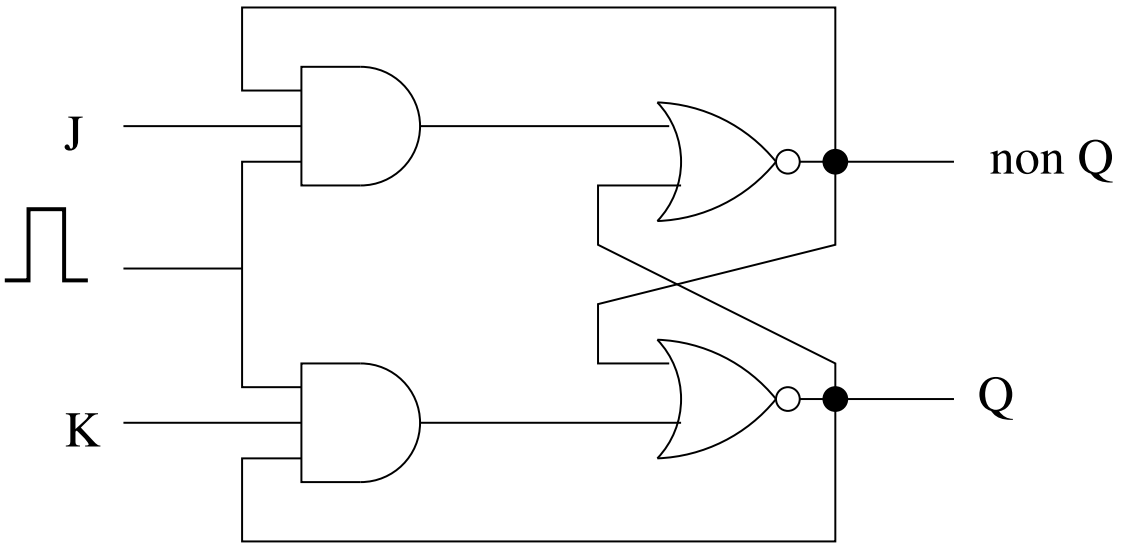
Les états stables sont Q=0, $\overline{Q=1}$ pour les entrées S=R=0, et Q=1, $\overline{Q}$=0 pour les entrées S=R=0.
Les bascules se produisent lorsque Q=0 et S passe de 0 à 1, et lorsque Q=1 et R passe de 0 à 1.
L'état instable se produit lorsque R=S=1.
Il est peut être intéressant d'autoriser une bascule à ne changer d'état qu'à certains moments bien précis. La bascule suivante est une bascule RS commandée par un signal (en général une horloge).



La couche matérielle

La bascule JK

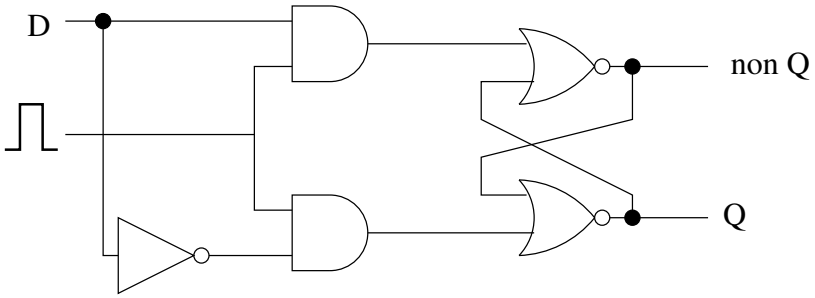
L'état instable de la bascule RS est désagréable.
Pour y remédier on a créé la bascule JK.
L'idée est de bloquer l'une des deux entrées, cela est réalisé par un bouclage arrière des deux sorties sur les deux entrées. Comme une des sorties est toujours égale à 0, une des entrées est toujours bloquée.



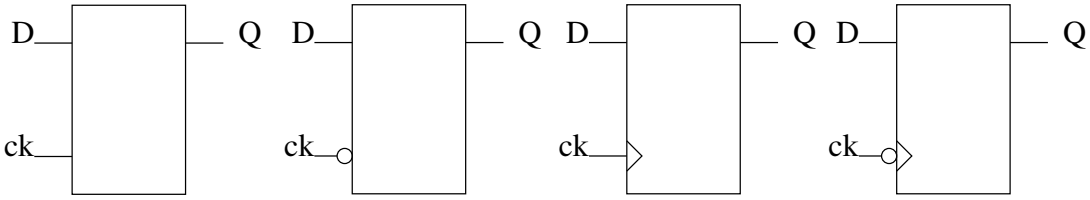
La couche matérielle

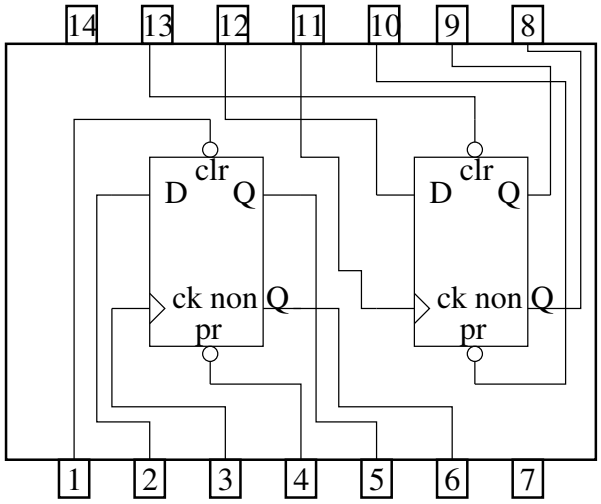
La bascule D

Le problème de l'état instable R=S=1 est définitivement réglé en utilisant qu'une seule entrée et sa négation.
La bascule D est une mémoire de 1 bit. Ce bit étant enregistré lorsque l'horloge l'autorise.



Il existe différents types de bascules. Sur niveau d'horloge (haut ou bas) et sur front d'horloge (haut ou bas).



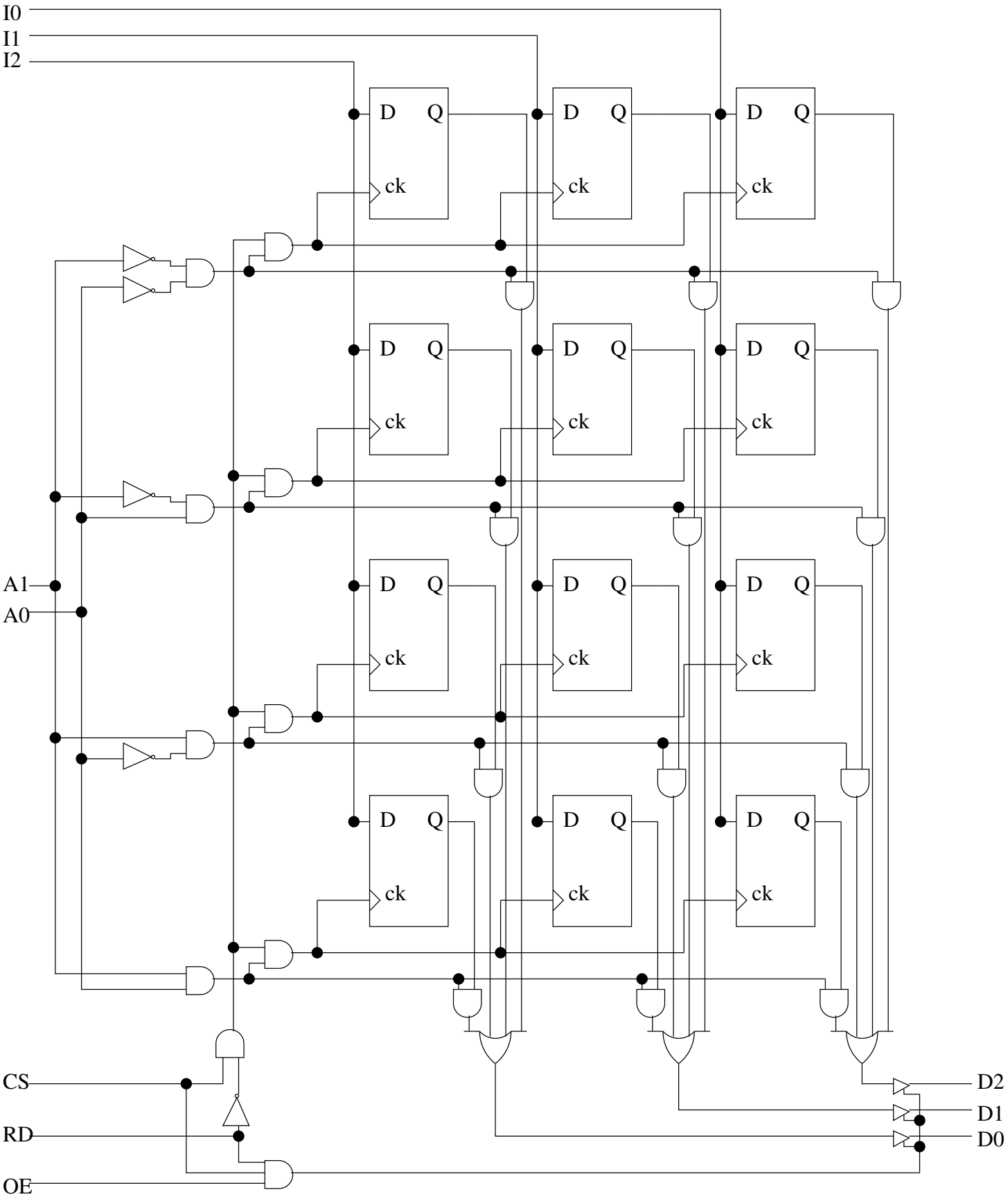


Le boitier 7474

La couche matérielle

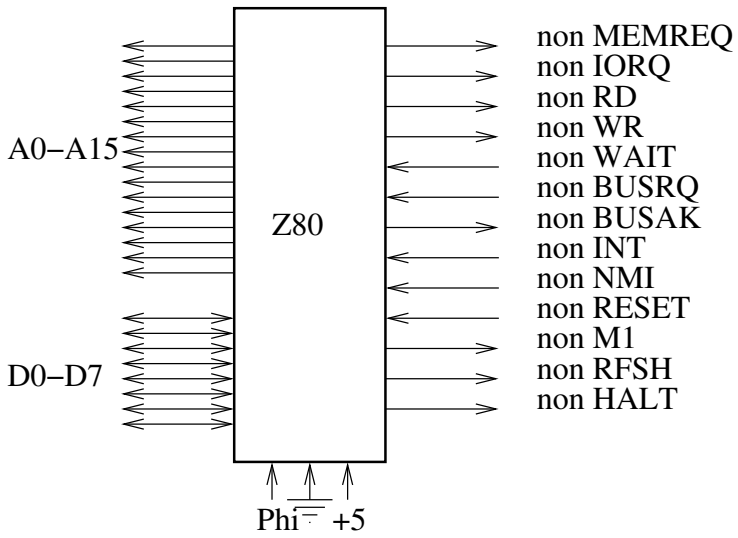
Les mémoires

Voici un boitier offrant 4 emplacements mémoire de trois bits chacun.



La couche matérielle

Le Z80



Il possède un bus d'adresses de 16 bits ($2^{16} = 65536$ adresses possibles),
Un bus de données de 8 bits.
Un bus de commandes de 13 signaux.
Les signaux de commande sont actifs à l'état bas, c'est à dire que la commande correspondante est active si c'est un 0 qui circule.
Il existe un état très particulier appelé
≪ flottant ≫ permettant à un autre dispositif d'accéder au bus et d'en devenir le maître.

Les entrées/sorties

Pour lire un mot mémoire le Z80 positionne l'adresse sur le bus d'adresses et positionne $\overline{\text{MEMREQ}}$ et $\overline{\text{RD}}$ à 0. Au bout d'un temps déterminé, il lit les valeurs présentes sur le bus de données. Le circuit mémoire doit réagir en au plus 365ns.

Pour écrire un mot, le Z80 positionne l'adresse sur le bus d'adresses et la donnée sur le bus de données et positionne $\overline{\text{MEMREQ}}$ et $\overline{\text{RD}}$ à 0.

Pour réaliser des entrées/sorties c'est IOREQ qui est utilisé à la place de MEMREQ.

Le signal WAIT est utilisé par le dispositif de mémoire, afin d'indiquer au Z80 que la réponse est doit être attendue plus longtemps que prévu. Tant que ce signal est activé, le Z80 est ≪ figé ≫ .

Le partage du bus

Le dispositif désireux de prendre le contrôle du bus doit en faire la requête par l'intermédiaire de BUSRQ. Alors le Z80 termine son cycle courant, place les signaux MEMREQ, RD, WR et IORQ dans un état flottant et positionne le signal BUSAK.

La couche matérielle

Le Z80

Les interruptions

Si un dispositif externe est désireux de prévenir le Z80 qu'un événement s'est produit, il dispose des signaux NMI et INT pour cela. La différence étant que INT peut être ignoré par le Z80 et pas NMI.

La remise à zéro

Le signal RESET permet de forcer le compteur ordinal à 0. Il est généralement relié à un bouton poussoir.

Le rafraîchissement de la mémoire

Les signaux M1 et RFSH sont utilisés pour détecter le cycle fonctionnel du microprocesseur. Lorsque RFSH est actif, le bus d'adresse contient une adresse à laquelle le rafraîchissement doit être effectué (en fait un banc de mémoire). Un compteur interne est en même temps incrémenté pour permettre de rafraîchir un autre banc au coup suivant. M1 correspond au cycle fonctionnel de décodage du code opération.

Le fonctionnement en pas à pas

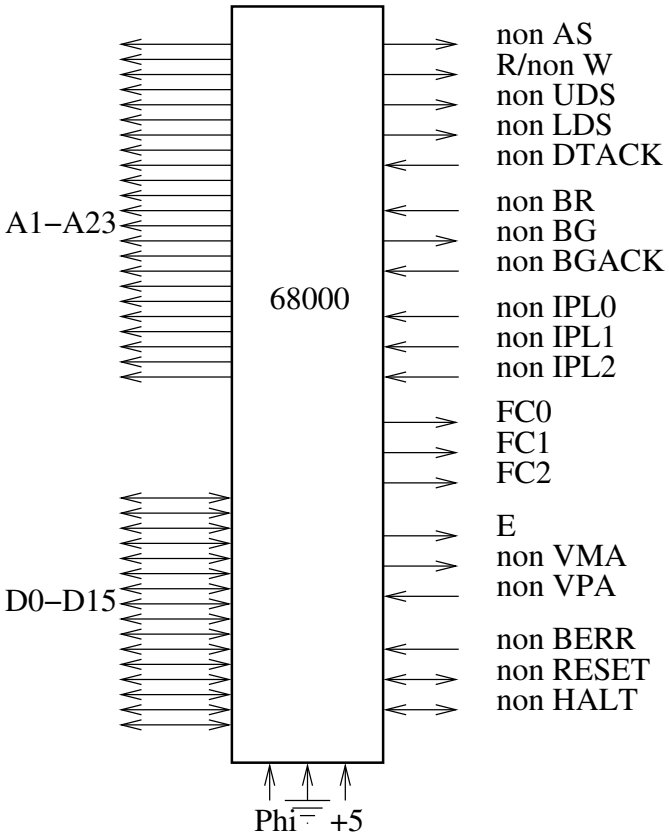
Le fonctionnement normal est au plus de 6Mhz (selon les versions).

Il est possible de faire fonctionner le Z80 en mode très ralenti (pas à pas) en jouant sur la durée des cycles d'horloge.

Le signal HALT est émis lorsque le processeur désire indiquer qu'il est en état oisif (il ne fait rien, sauf rafraîchir la mémoire).

La couche matérielle

Le 68000



Il possède un bus d'adresses de 23 bits, mais il s'agit d'adresses de mots de 16 bits. Donc l'espace d'adressage est de $2^{23} \times 2 = 16$Mo.

Un bus de données de 16 bits.

Un bus de commandes de 20 signaux.

Les entrées/sorties

Pour lire il faut activer le signal R (position active à 1) et la signal AS (sélection d'adresse).
Pour écrire il faut activer le signal W (position active à 0).
Les signaux UDS et LDS sont utilisés pour effectuer des opérations sur 8 bits ou 16 bits. UDS (Upper Data Select) permet d'indiquer que les 8 bits de poids forts sont valides. LDS (Lower Data Select) indique que les 8 bits de poids faibles sont valides. Pour valider un mot de 16 bits il suffit d'activer les deux signaux à la fois.
Dans le cas du 68000, la mémoire n'a pas de contrainte particulière quant au temps de réponse (on dit que le Z80 est à ≪ transferts synchrones ≫). Le 68000 est à ≪ transferts asynchrones ≫ . Lorsque le dispositif de mémoire a terminé son opération, il le signal en activant DTACK.

La gestion du bus

Le signal BR est utilisé par un dispositif externe désireux d'acquérir la maîtrise du bus. Le 68000 répond en activant BG, indiquant ainsi que le bus est accordé au demandeur. Le signal BGACK est activé en retour par le dispositif externe jusqu'à ce qu'il ne désire plus le bus. la réception de BGACK le 68000 libère BR pour permettre à quelqu'un d'autre d'effectuer une requête anticipée.

Le signal BERR permet de détecter une erreur survenue sur le bus. Par exemple un dispositif ne répond pas en temps raisonnable, alors un dispositif de détection d'erreur (chien de garde) active ce signal permettant ainsi au 68000 de reprendre un fonctionnement normal (par exemple s'il attends DTACK trop longtemps).

Les interruptions

Il existe trois niveaux d'interruptions, c'est à dire une priorité entre celles-ci.

Les états fonctionnels

Les signaux FC permettent d'indiquer dans quel état fonctionnel se trouve le processeur.

La communication avec les périphériques MOTOROLA 6800

Les trois signaux E, VMA et VPA permettent de communiquer efficacement avec les circuits périphériques de la famille 6800.

La remise à zéro

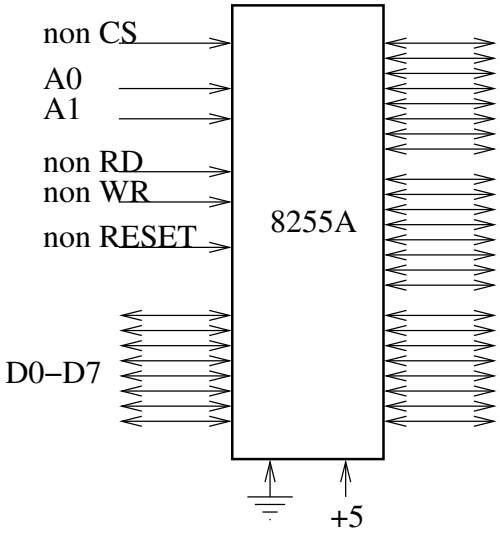
Ce signal est identique à celui du Z80.

Le fonctionnement pas à pas

C'est le signal HALT qui sert à cela. Ici il est interdit de maintenir l'horloge à zéro pour ralentir le processeur. Il faut activer ce signal.

La couche matérielle

Les entrées/sorties

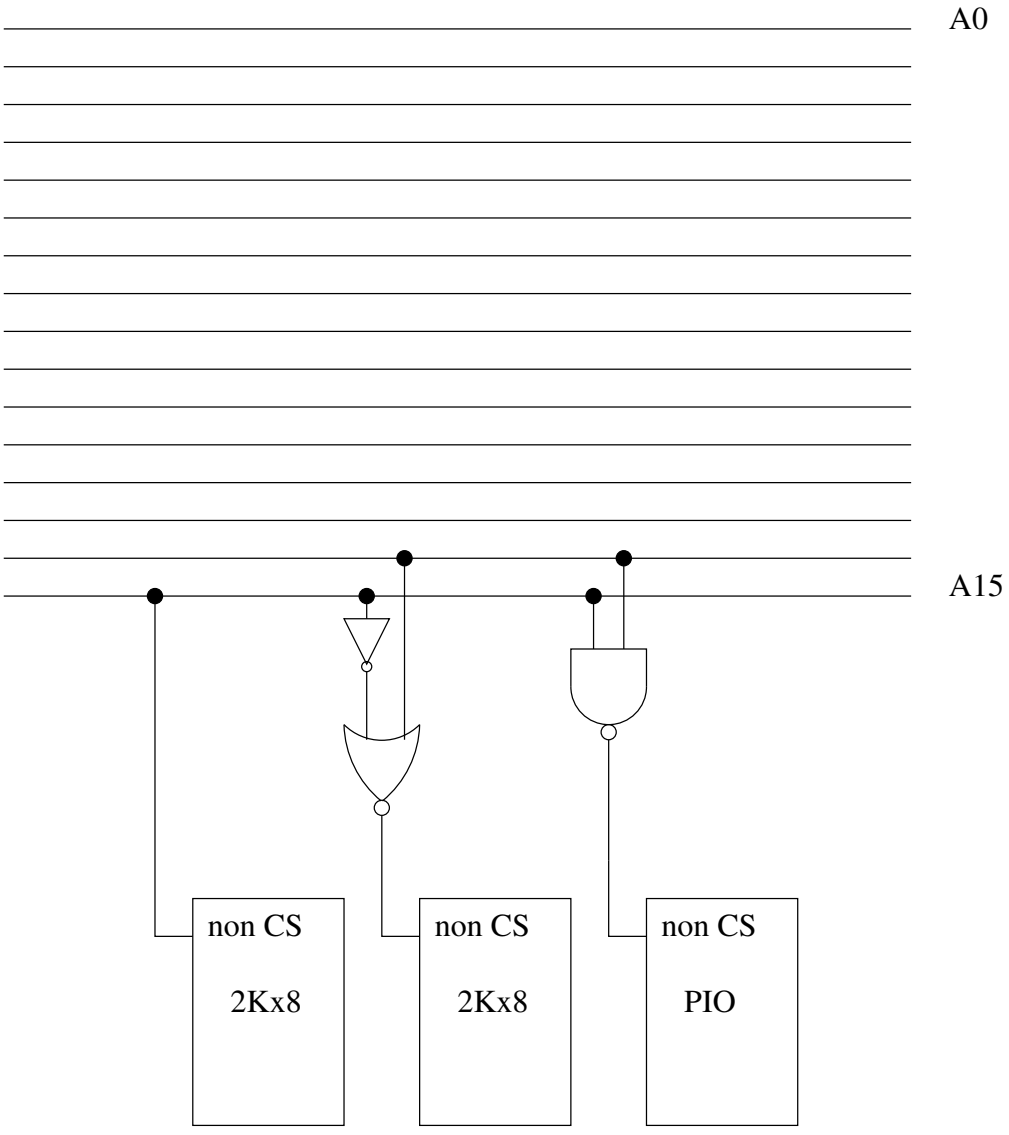


Il offre trois ports d'entrées ou sorties A, B et C qui chacune contient 8 lignes.
À chaque porte est associé un registre interne de 8 bits.
Pour transmettre une information il suffit de charger le registre concerné, ensuite le 8255A se charge d'émettre cette donnée en permanence sur les lignes concernées.
Évidemment les lignes D0-D7 sont connectés sur le bus de données.
Le signal CS permet de sélectionner le circuit. Les signaux RD et WR sont utilisés pour lire ou écrire.
Les lignes A0 et A1 permettent d'adresser l'une des trois portes ou un registre interne très particulier permettant de réaliser certaines choses.

La couche matérielle

Le décodage d'adresse

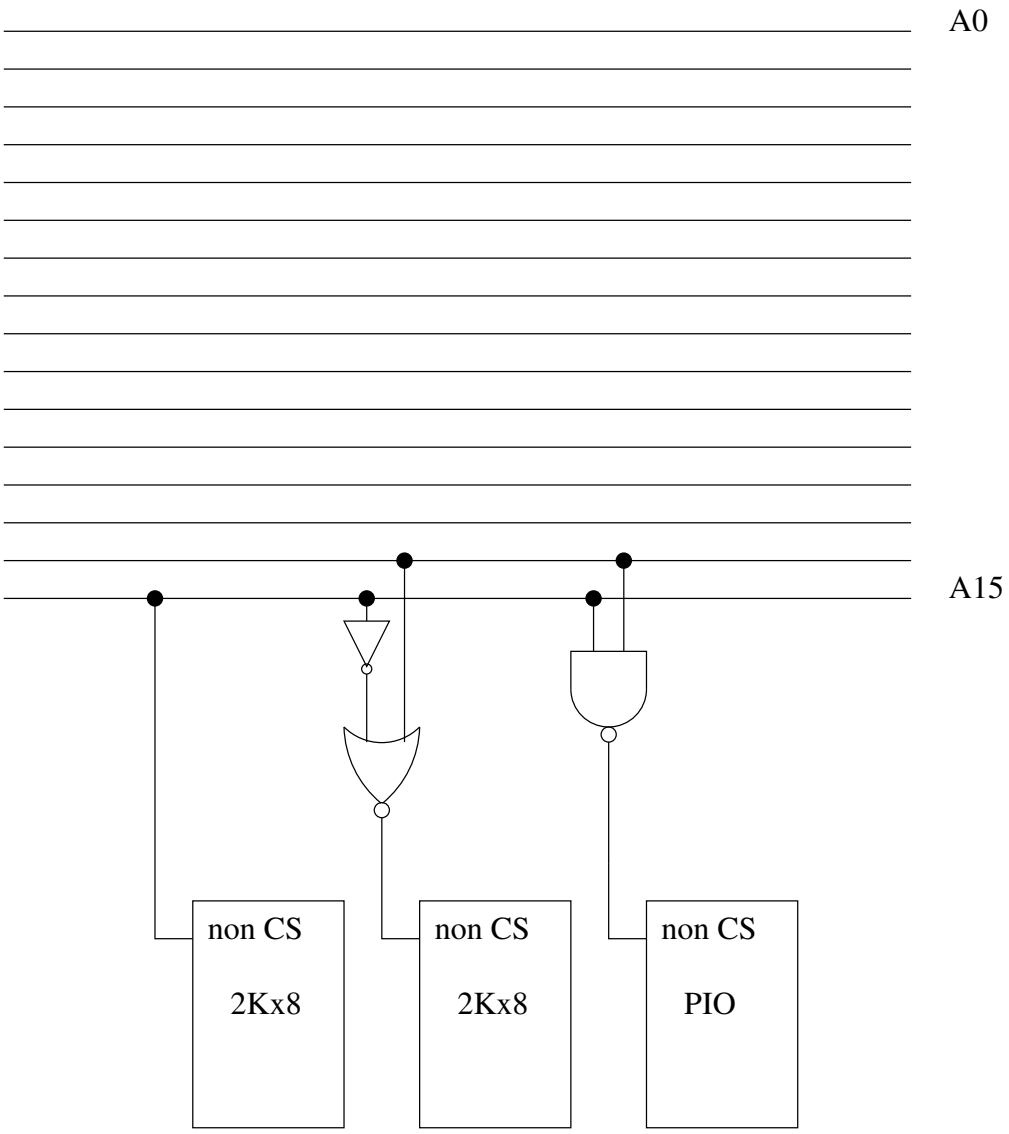
Supposons que l'on veuille construire un micro-ordinateur composé d'un Z80, de deux circuits mémoires (1 ROM de 2Ko située aux adresses 0x0000 à 0x07FF, 1 RAM de 2Ko située aux adresses 0x8000 et 0x87FF) ainsi que d'un dispositif d'entrées/sorties aux adresses 0xFFFC à 0xFFFF.



La couche matérielle

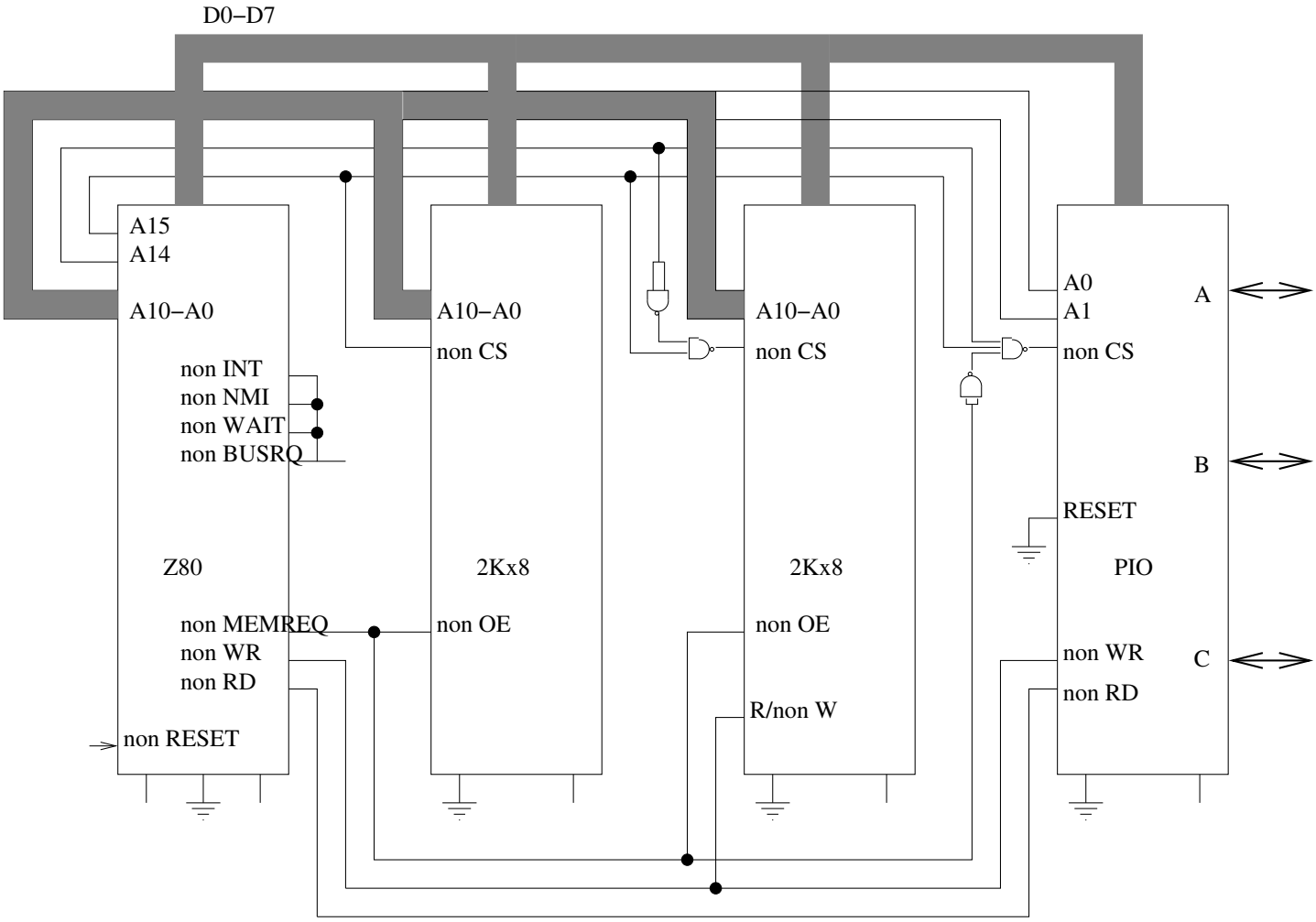
Le décodage d'adresse

Pour économiser des portes il est possible de décoder partiellement les adresses.



La couche matérielle

Un ordinateur

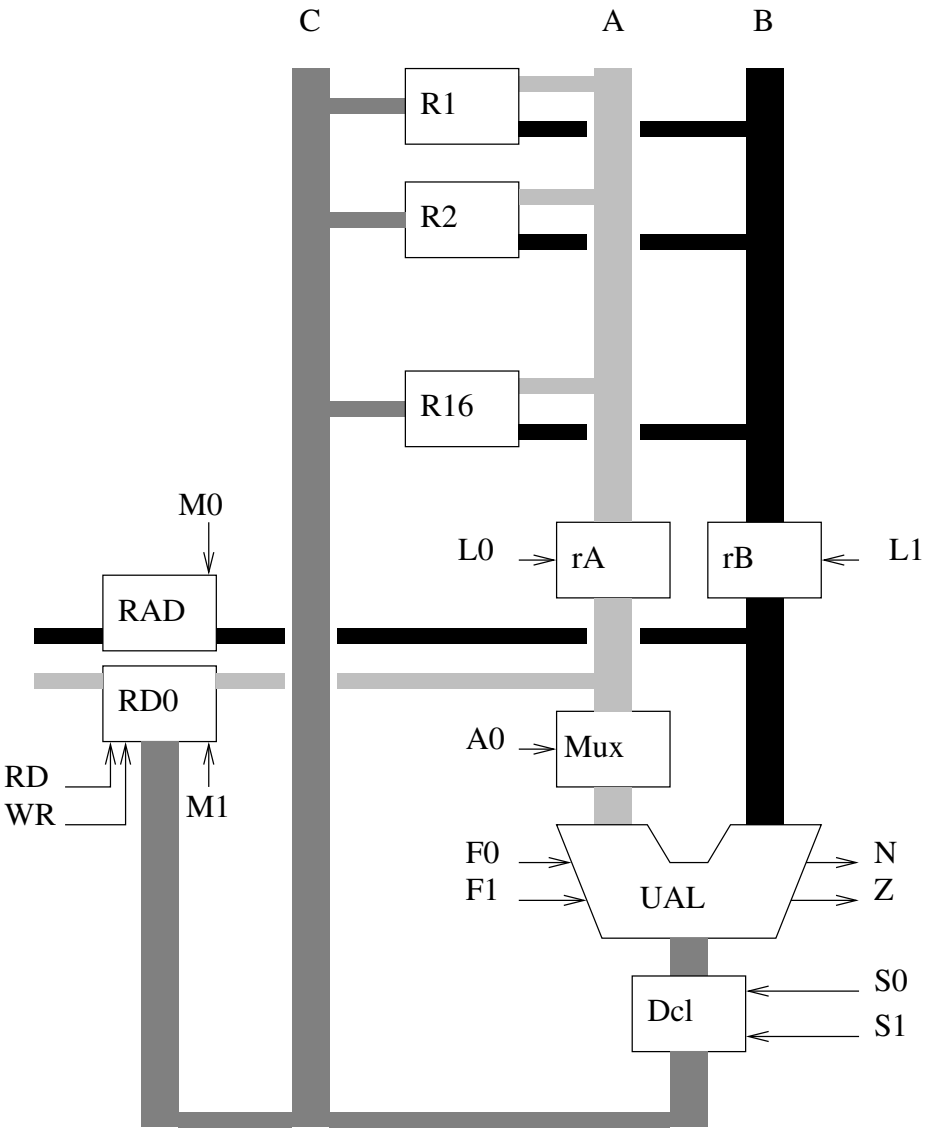


Thèmes de réflexion

- réaliser un démultiplexeur

La couche micro-programmée

Le chemin des données

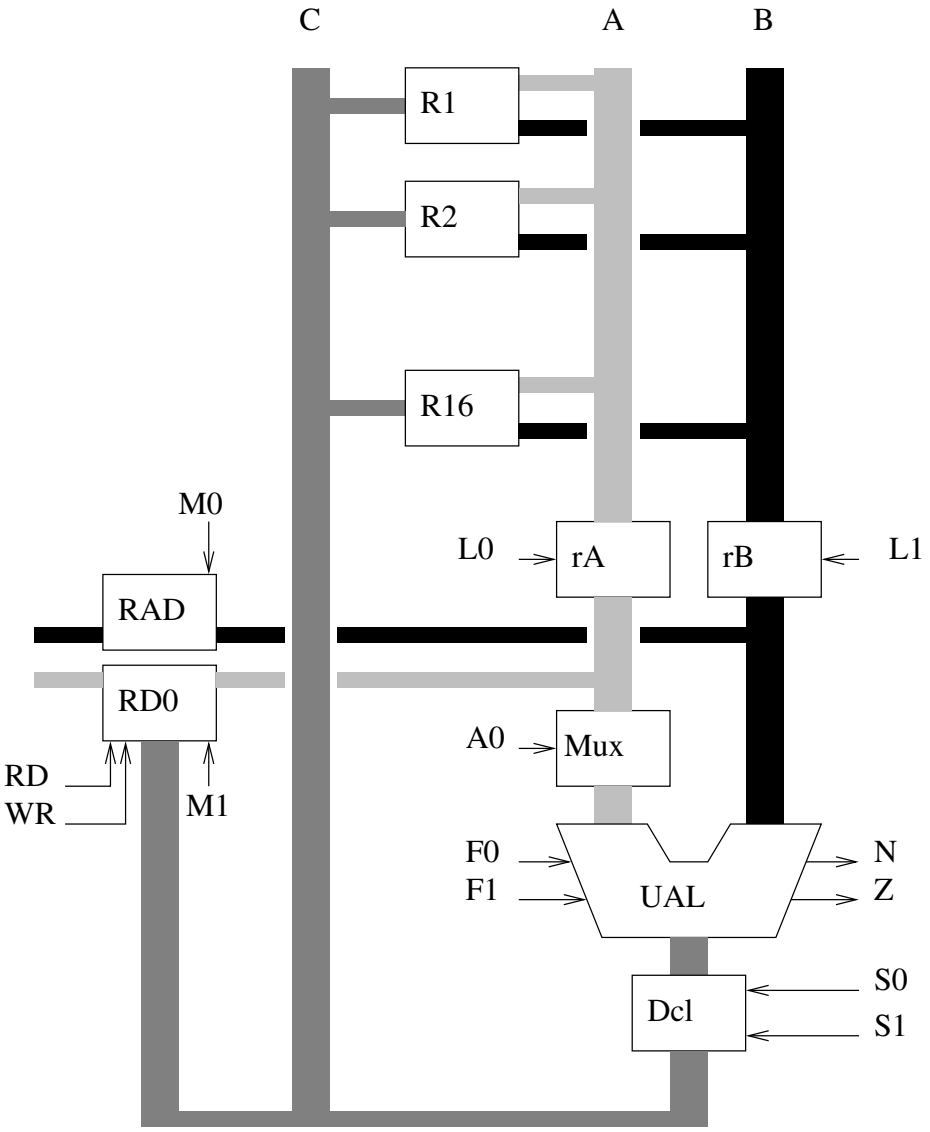


La partie centrale est un banc de 16 registres dits ≪ internes ≫ c'est avec eux que l'on calcule ordinairement et le plus possible (ils sont au cœur de la machine :

- CO (Compteur Ordinal),
- AC (Accumulateur),
- PP (Pointeur de Pile),
- RI (Registre d'Instruction),
- RIT (Registre Instruction Temporaire),

La couche micro-programmée

Le chemin des données

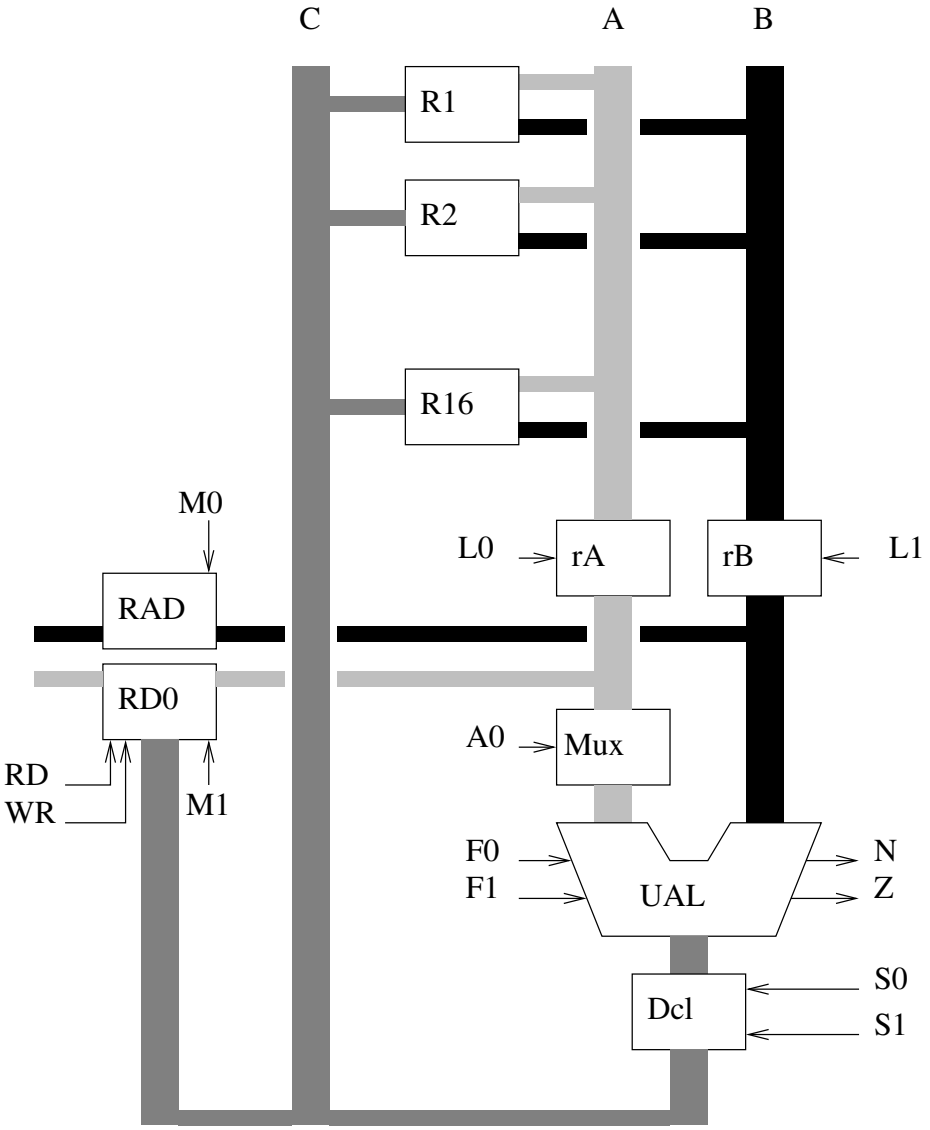


La partie centrale est un banc de 16 registres dits ≪ internes ≫ c'est avec eux que l'on calcule ordinairement et le plus possible (ils sont au cœur de la machine :

- 0 (Constante 0),
- +1 (Constante +1),
- -1 (Constante -1),
- AMASQ (Masquage d'Adresses),
- PMASQ (Masquage de Pile),
- et A, B, C, D, E, F.

La couche micro-programmée

Le chemin des données

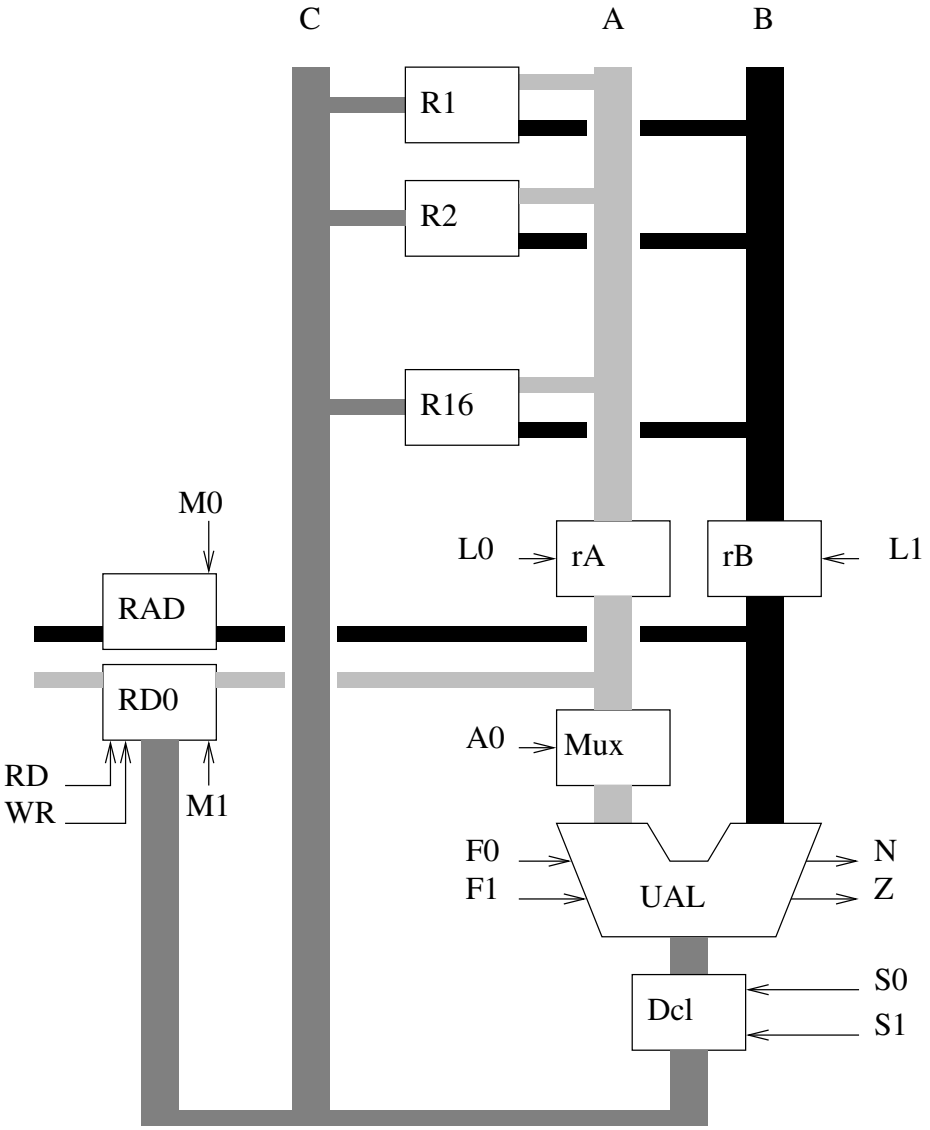


Les signaux :

- L0 et L1 contrôlent le chargement des deux tampons A et B permettant de stabiliser l'entrée de l'UAL.
- A0 contrôle le passage des données en provenance de la mémoire ou du bus A.
- F0 et F1 sélectionnent le fonctionnement de l'UAL. N est actif si le résultat est négatif et Z est actif si le résultat est nul.

La couche micro-programmée

Le chemin des données



Les signaux :

- S0 et S1 permettent de décaler à gauche ou à droite.
- M0 permet de charger le registre du bus d'adresses pour stabiliser la valeur.
- M1 permet de charger le registre du bus de données avec la valeur en provenance de la sortie de l'UAL. WR permet de positionner la valeur du registre sur le bus de données. RD charge la valeur en provenance du bus des données dans le registre.

La couche micro-programmée

Les micro-instructions

Une micro-instruction est un ≪ champ de bits ≫ permettant d'activer ou de désactiver l'ensemble des signaux de contrôle du processeur.
La micro-instruction de notre processeur est découpée en 13 champs :

- MUX : 1 bit. Il contrôle quelle opérande doit entrer dans l'UAL. 0=bus A, 1=bus données. C'est A0.
- COND : 2 bits. 0=rien, 1=saut si N=1, 2=saut si Z=1, 3=saut.
- UAL : 2 bits. Opérations à réaliser par l'UAL. 0=A+B, 1=A&B, 2=A, 3=A (par exemple). C'est F0 et F1.
- DECAL : 2 bits. Décalage à droite ou à gauche en sortie de l'UAL. C'est S0 et S1. 0=rien, 1=à droite et 2=à gauche.
- RAD : 1 bit. Chargement du bus B dans le registre d'adresse. 0=rien, 1=chargement. C'est M0.
- RD0 : 1 bit. Chargement du bus C dans le registre de données. 0=rien, 1=chargement. C'est M1.
- RD : 1 bit. lecture de la mémoire principale. 0=rien, 1=lecture.
- WR : 1 bit. écriture sur la mémoire principale. 0=rien, 1=écriture.

La couche micro-programmée

Les micro-instructions

Une micro-instruction est un ≪ champ de bits ≫ permettant d'activer ou de désactiver l'ensemble des signaux de contrôle du processeur.
La micro-instruction de notre processeur est découpée en 13 champs :

VALC : 1 bit. Autorise le chargement d'un des registres du banc.

C : 4 bits. Adresse d'un registre à charger avec une valeur en provenance du bus C.

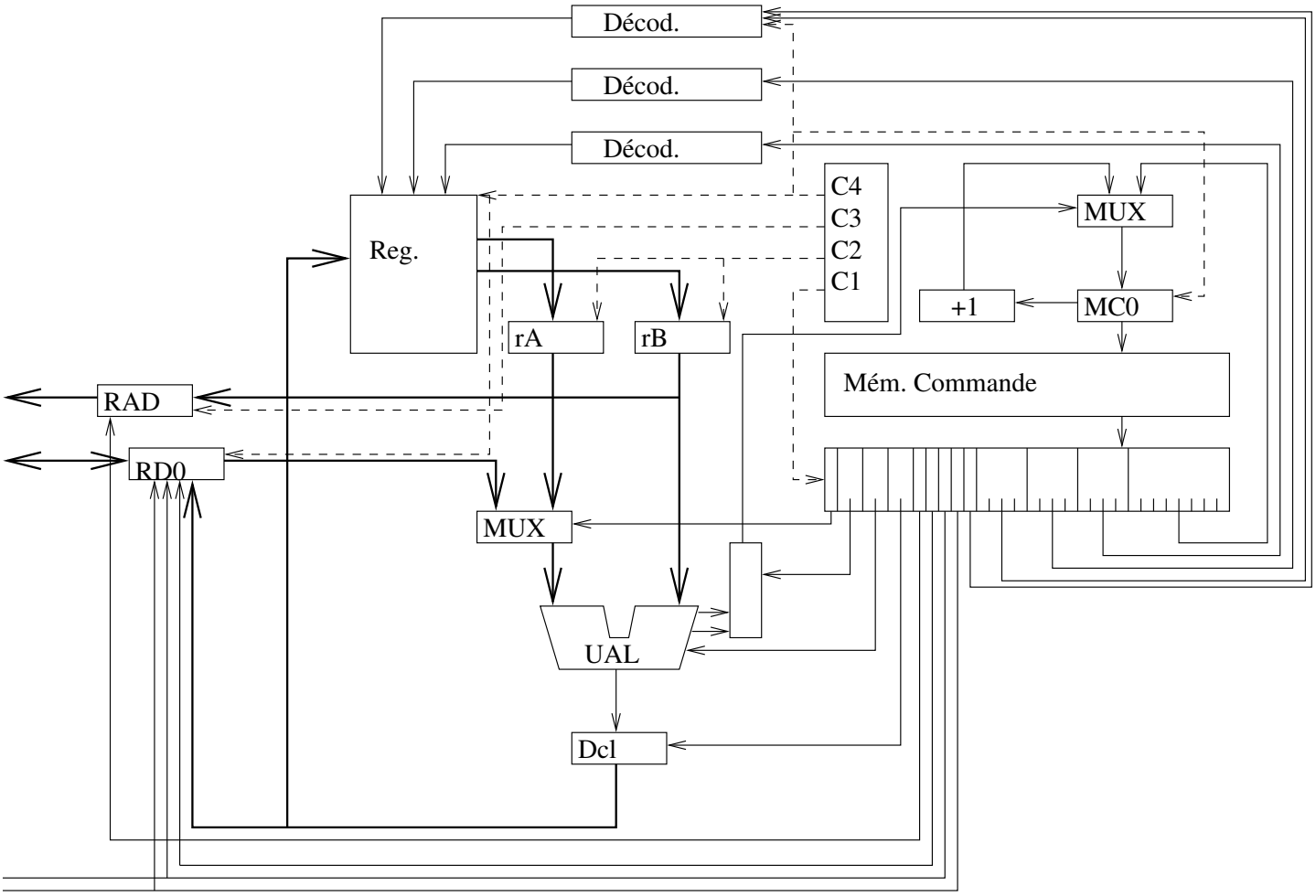
B : 4 bits. Adresse d'un registre à charger avec une valeur en provenance du bus B.

A : 4 bits. Adresse d'un registre à charger avec une valeur en provenance du bus A.

ADR : 8 bits. Adresse de la prochaine micro-instruction à exécuter.

La couche micro-programmée

La micro-machine



La mémoire de commande contient le micro-programme.
MCO est un registre contenant l'adresse de l'instruction du micro-programme à exécuter.

La couche micro-programmée

Exécution d'une instruction de chargement direct

On suppose que nos instructions (niveau 2) sont toutes exprimées sur 2 octets. Donc 16 bits : 4 bits pour le code de l'instruction, et 12 bits de données.
On désire offrir une instruction comme `LDA x`, qui permet de charger l'accumulateur (AC) avec le contenu de la mémoire dont l'adresse est x (attention x est sur 12 bits).
Supposons que le registre CO contient l'adresse du premier octet de l'instruction.
Alors, il faut :

- Charger l'octet CO dans le registre RI :

1. Pour cela il faut positionner CO dans RAD, et lire le résultat dans RDO : 0.00.10.00.1.0.1.0.0.0000.0000.0000.00000000. UAL=2, RAD=1, RD=1 et A=0.
2. Incrémenter CO : 0.00.00.00.0.0.1.0.1.0000.0110.0000.00000000. VALC=1, C=0, B=6, A=0, RD=1.
3. Puis charger RDO dans RI : 1.00.10.00.0.0.0.0.1.0000.0000.0000.00000000. MUX=1. UAL=2, VALC=1. C=3.

La couche micro-programmée

Exécution d'une instruction de chargement direct

Puis :

- Charger le contenu de l'adresse dans AC :

1. Charger les 12 bits faibles de RI dans RAD (facile si RAD est un registre de 12 bits) : 0.00.10.00.1.0.1.0.0.0000.0000.0000.00000000. UAL=2, RAD=1, RD=1 et A=3.
2. On attend la réponse de la mémoire : 0.00.10.00.0.0.1.0.0.0000.0000.0000.00000000. UAL=2, RD=1.
3. On charge la mémoire dans l'accumulateur : 1.00.10.00.0.0.0.0.1.0001.0000.0000.00000000. MUX=1, UAL=2, VALC=1, C=1.
4. Il faut passer à l'instruction suivante...

La couche micro-programmée

Exécution d'une instruction d'addition directe

Cette fois on désire offrir une instruction d'addition `ADD x` qui permet d'additionner le contenu de l'adresse x avec le contenu de l'accumulateur.
On suppose encore que CO contient l'adresse du premier octet de l'instruction.

- Il faut charger CO dans RAD :

1. Pour cela il faut positionner CO dans RAD, et lire le résultat dans RDO : 0.00.10.00.1.0.1.0.0.0000.0000.0000.00000000. UAL=2, RAD=1, RD=1 et A=0.
2. Incrémenter CO : 0.00.00.00.0.0.1.0.1.0000.0110.0000.00000000. VALC=1, C=0, B=6, A=0, RD=1.
3. Puis charger RDO dans RI : 1.00.10.00.0.0.0.0.1.0000.0000.0000.00000000. MUX=1. UAL=2, VALC=1. C=3.

La couche micro-programmée

Exécution d'une instruction d'addition directe

- Puis additionner AC et le contenu de l'adresse contenue dans RI :

1. Charger les 12 bits faibles de RI dans RAD (facile si RAD est un registre de 12 bits) : 0.00.10.00.1.0.1.0.0.0000.0000.0000.00000000. UAL=2, RAD=1, RD=1 et A=3.
2. On attends la réponse de la mémoire : 0.00.10.00.0.0.1.0.0.0000.0000.0000.00000000. UAL=2, RD=1.
3. Additionner AC et le contenu de la mémoire dans AC : 1.00.10.00.0.0.0.0.1.0001.0001.0000.00000000. MUX=1, UAL=0, VALC=1, B=1, C=1.
4. Il faut passer à l'instruction suivante...

La couche micro-programmée

Exécution d'une instruction de saut conditionnel

Cette fois on désire réaliser une instruction de saut `JMPZ x` qui permet de sauter à l'instruction de numéro x si le contenu de l'accumulateur est nul.
On suppose encore que CO contient l'adresse du premier octet de l'instruction.

- Il faut charger CO dans RAD :

1. Pour cela il faut positionner CO dans RAD, et lire le résultat dans RDO : 0.00.10.00.1.0.1.0.0.0000.0000.0000.00000000. UAL=2, RAD=1, RD=1 et A=0.
2. Incrémenter CO : 0.00.00.00.0.0.1.0.1.0000.0110.0000.00000000. VALC=1, C=0, B=6, A=0, RD=1.
3. Puis charger RDO dans RI : 1.00.10.00.0.0.0.0.1.0000.0000.0000.00000000. MUX=1. UAL=2, VALC=1. C=3.

La couche micro-programmée

Exécution d'une instruction de saut conditionnel

- Vérifier le contenu de AC :

1. Charger AC dans l'UAL et tester si c'est nul : 0.10.10.00.0.0.0.0.0.0000.0000.0000.xxxxxxxx. COND=2, UAL=2, et ADR=xxxxxxxx.
2. Si c'est pas nul il faut passer à l'instruction suivante...
3. Si c'est nul, il faut charger les 12 bits de poids faibles de RI dans CO :
   - 3.1 Pour cela il faut calculer le ET logique entre RI et AMASQ : 0.00.01.00.0.0.0.0.1.0000.1000.0011.00000000. UAL=1, VALC=1, A=3, B=8, C=0.
   - 3.2 Additionner AC et le contenu de la mémoire dans AC : 1.00.10.00.0.0.0.0.1.0001.0001.0000.00000000. MUX=1, UAL=0, VALC=1, B=1, C=1.
   - 3.3 Il faut passer à l'instruction suivante...

La couche micro-programmée

La nano-programmation

Ici nous avons utilisé une mémoire de $n \times w$ bits pour le micro-programme. n instructions sur w bits chacune. La plupart du temps les 2^w possibilités ne sont pas utilisées, soit m le nombre effectif des micro-instructions utilisées. Alors on prend k bits pour numéroter ces m instructions, et une mémoire de $m \times w$ pour obtenir les vraies micro-instructions. On a donc $(k \times n) + (m \times w)$ bits utilisés.
Évidemment l'exécution est plus longue car il faut d'abord charger le mot de k bits, puis aller chercher la micro-instruction de w bits puis l'exécuter...

Remarque : c'est un principe assez général, si l'on désire économiser de l'espace (de la mémoire) le prix à payer est celui du temps et vice-versa...

La couche machine

Le Z80

Il possède huit registres de 8 bits principaux : A, B, C, D, E, F, H et L.
B, C, D. E, H et L sont des registres temporaires *i.e.* qui ne doivent pas être utilisés pour accumuler des valeurs...
F est le registre des drapeaux (résultats de tests précédents).
Il peuvent être groupés par paires BC, DE et HL pour former des mots de seize bits. HL permet de manipuler des adresses.
Il existe un autre jeu de registres internes appelés complémentaires et nommés A', B', C', D', E', F', H' et L'. Ils sont utilisés en cas d'interruption.
On trouve aussi IX et IY deux registres de 16 bits, qui servent à indexer des tableaux en mémoire (256 octets par tableau).
SP est le pointeur de pile, il contient 16 bits.
PC est le compteur ordinal ou pointeur d'instruction.
Il existe aussi un registre appelé R servant au rafraîchissement des bancs mémoire.

La couche machine

Le Z80

Voici quelques instructions :

`LD A,src` : Chargement du registre A avec une valeur.

`LD BC,src` : Chargement du registre double BC avec une valeur.

`EX DE,HL` : Échange du contenu des registres doubles DE et HL.

`CPI` : Comparer A au contenu pointé par HL, incrémenter HL et décrémenter BC.

`ADD src` : Additionner src au registre A.

`ADD HL,BC` : Additionner BC à HL.

`NEG` : Complémenter à deux le registre A.

`JP cond,DST` : Saut conditionnel à DST si la condition est vraie.

`OUT (n),A` : Sortie du registre A vers le port de numéro n.

La couche machine

Des programmes pour Z80

```
                090 ; Cette routine permet de remplir une zone de mémoire
                091 ; avec la valeur contenue dans le registre D
                092 ; Le début de zone est pointé par HL,
                093 ; le nombre d'octets à remplir est contenu dans BC.
8000            100       ORG   8000H       ; Origine d'implantation en 8000
8000 72         110 REMP  LD    (HL),D      ; Enregistre la donnée
8001 23         120       INC   HL          ; Adresse suivante
8002 0B         130       DEC   BC          ; Décrémente le compteur
8003 78         140       LD    A,B         ; Charge la partie basse du compteur
8004 B1         150       OR    C           ; Vérifie la nullité du compteur
8005 20F9       160       JR    NZ,FILL     ; Si c'est pas fini on continue
8007 C9         170       RET               ; Fin de la sous-routine
                180       END
```

```
                090 ; C'est une addition en précision multiple
                091 ; IX pointe sur l'octet de poids fort de la première
                092 ; opérande, IY sur la deuxième.
                093 ; B contient la longueur des opérandes
8000            100       ORG   8000H
8000 D5         110 LADD  PUSH  DE          ; Sauvegarde de DE
8001 58         120       LD    E,B         ; La longueur est dans E
8002 1600       130       LD    D,0         ; DE contient la longueur
8004 1B         140       DEC   DE          ; DE contient la longueur - 1
8005 DD19       150       ADD   IX,DE       ; IX pointe sur l'octet de poids faible
8007 FD19       160       ADD   IY,DE       ; Idem IY
8009 D1         170       POP   DE          ; On retrouve le DE original
800A AF         180       XOR   A           ; La retenue est mise à 0
800B DD7E00     190 BCLE  LD    A,(IX)      ; On prend un chiffre
800E FD8E00     200       ADC   A,(IY)      ; On additionne avec un chiffre
8011 DD7700     210       LD    (IX),A      ; On range le résultat
8014 1001       220       DJNZ  BCL1        ; Si c'est pas fini
8016 C9         230       RET               ; Fin de la sous-routine
8017 DD2B       240 BCL1  DEC   IX          ; On passe au suivant
8019 FD2B       250       DEC   IY          ; Idem
801B C30B80     260       JP    BCLE        ; On continue
                270       END
```

La couche machine

Les modes d'adressages

L'adressage immédiat : On charge un registre avec une constante.
L'adressage direct : On charge un registre avec la valeur située à une adresse constante.
L'adressage par registre : On charge un registre avec la valeur située dans un registre.
L'adressage indirect : On charge dans un registre la valeur située à l'adresse que l'on trouve dans un registre ou dans une autre adresse mémoire.
L'adressage indexé : L'adresse effective d'écriture et/ou de lecture est calculée en additionnant le contenu d'un registre dit ≪ d'index ≫ (lequel est incrémenté d'une unité après l'opération) avec une constante.
L'adresse par registre de base : Les adresses ne sont plus absolues mais relatives à une valeur contenue dans un registre dit de ≪ base ≫ .
L'adressage par pile : Il s'agit de charger des données les unes ≪ au-dessus ≫ des autres. On peut alors les décharger dans l'ordre inverse.

Les processeurs modernes

L'architecture RISC

Aujourd'hui la tendance est de ne construire que des processeurs dit ≪ RISC ≫ (Reduced Instruction Set Computer) : MIPS, SPARC, HPPA, Power PC, Alpha. Exception faite des processeurs d'Intel qui restent dans la tradition CISC (Complex Instruction Set Computer). On a longtemps cru que plus les instructions fournies étaient puissantes plus les performances étaient grandes. En fait, il est possible de s'apercevoir que les instructions les plus puissantes sont les moins utilisées. On a donc imaginé de supprimer les instructions les plus puissantes (cosinus en flottant, etc.), cela a donc permis de simplifier grandement la circuiterie et par conséquence a diminué le chemin parcouru par les données et donc augmenté la vitesse de traitement. Les processeurs RISC ont été développés en respectant les règles suivantes :

- La régularité favorise la simplicité (ex : nombre d'opérandes).
- Plus c'est petit, plus c'est rapide (ex : nombre de registres).
- Une bonne implantation requière des compromis (ex : longueur des instructions et format).
- Exécuter les instructions les plus fréquentes le plus rapidement possible.

Un exemple : le R2000 de MIPS

Toutes les instructions sont sur 32 bits. Le chargement de celles-ci est donc uniforme.
Comme le bus des données est de 32 bits, les instructions sont chargées en une seule fois.
Il contient 32 registres. Ce nombre peut paraître petit, mais il permet de diminuer la distance parcourue par les signaux.
Aujourd'hui il n'est plus possible de programmer en assembleur de façon plus efficace que le code généré par la compilation (pipelines, etc.).
Les processeurs RISC sont développés en gardant un œil fixé sur les langages de niveaux supérieurs.

Les processeurs modernes

Le pipelining

Jusqu'ici nous avons vu que pour exécuter une instruction il est nécessaire d'attendre la fin de l'instruction précédente.
Mais il est possible d'accélérer en remarquant que dans la circulation des données certains chemins ne sont pas connexes. On peut imaginer une architecture dans laquelle les différents cycles fonctionnels sont indépendants (du point de vue de leurs chemins).
On peut construire une architecture permettant de décoder une instruction (déterminer quel sous-micro-programme est à exécuter) pendant que l'instruction précédente finit de se terminer (réalise son calcul dans l'UAL), le tout pendant que l'instruction encore avant range ses données après calcul.
Alors on voit bien qu'ici trois instructions s'exécutent presque en même temps. En tout cas il y a chevauchement des exécutions, donc gain de temps.
Tout les processeurs modernes utilisent cette technique. Le processeur MIPS contient jusqu'à cinq étages. C'est pourquoi, en régime stabilisé, il est possible d'affirmer que ce processeur exécute une instruction par cycle d'horloge ! Même si une instruction nécessite plusieurs cycles à elle toute seule.

Les processeurs modernes

La hiérarchie mémoire

Cette technique est utilisée depuis les années 60.
Plus une mémoire est rapide plus elle est chère.
D'autre part une mémoire lente ralentit le processeur qui passe son temps à attendre la réponse.
L'idée est donc d'utiliser une petite mémoire rapide (cache) entre le processeur et la mémoire lente.
Lorsque le processeur demande une valeur en mémoire, un dispositif permettra de lui donner la réponse rapidement si elle se trouve déjà dans le cache (cache hit), sinon (cache miss) il se chargera de récupérer la valeur dans la mémoire lente, et de la garder dans le cache pour une utilisation future.
Dans la pratique le cache ne fonctionne pas tout a fait comme cela. Lorsqu'il doit charger une valeur depuis la mémoire lente, il charge en fait un paquet de valeurs. Ce paquet est simplement un petit morceau contigu de la mémoire principale.
L'intérêt est que si l'on demande l'accès à une case d'adresse x, il est très probable que peu de temps après on demande à accéder aux cases voisines de x.
Le dispositif chargé de réaliser cela s'appelle une MMU (Memory Management Unit).
Évidemment, un programme mal écrit peut tout à fait faire en sorte qu'il ne fasse que des ≪ cache misses ≫ .
Dans ce cas les performances se dégradent rapidement.

La mémoire virtuelle

Index	V	Signature	Donnée
000			
001			
010			
011			
100			
101			
110			
111			

Un cache de 8 octets.

La mémoire virtuelle

C'est un peu le même principe que les caches. Mais ici il s'agit d'augmenter de façon artificielle la mémoire principale.
Si l'on possède un processeur capable d'adresser $2^{32} = 4Go$, il est fort peu probable que l'on possède effectivement autant de mémoire physique. Mais il suffit de posséder un disque de 4Go, et d'un dispositif permettant de faire jouer à la mémoire principale le rôle de cache vis-à-vis de cet espace disque.
Ici on parle alors de « page faults » et « TLB misses » (Translation-Lookaside Buffer). Une « page fault » apparaît lorsqu'une page n'est pas présente en mémoire principale, il faut la charger depuis le disque.

Les machines parallèles

Il existe deux grandes classes : SIMD (Single Instruction stream - Multiple Data streams) et MIMD (Multiple Instruction streams - Mutiple Data streams).
Les processeurs que nous avons étudiés sont SISD.
Parmi les SIMD, on trouve l'ILLIAC IV de l'Université d'Illinois, le DAP de chez ICL, le MPP de Goodyear, la CM-2 de Thinking Machines et la MP-1216 de Maspar.
La CM-2 possède jusqu'à 65536 processeurs 1-bit, quatre registres 1-bit et 64Ko de mémoire.
Aujourd'hui presque tous les constructeurs offre un modèle MIMD : IBM, DEC, Cray (les T3D, T3E, T90), Thinking Machines (CM-5)

Performances

Temps d'exécution d'un programme

$T_e(p)$: Temps d'exécution d'un programme p.
$N_c(p)$: Nombre de cycles d'horloge nécessaires à l'exécution d'un programme p.
T_c : Durée d'un cycle d'horloge.
F_h : Fréquence de l'horloge.
$N_i(p)$: Nombre d'instructions du programme p.
On notera CPI (Cycle Per Instruction) la quantité $N_c(i)$ qui est le nombre de cycles nécessaires pour exécuter l'instruction i.

$$T_e(p) = N_c(p) \times T_c$$

et comme $F_h = \frac{1}{T_c}$ on a :

$$T_e(p) = \frac{N_c(p)}{F_h}$$

$$N_c(p) = N_i(p) \times \overline{N_c(i)} = N_i(p) \times CPI$$

Performances

Temps d'exécution d'un programme

Comparaison :
Deux implantations différentes du même ensemble d'instructions.
$\mathcal{A}$: cycle d'horloge de 10 ns et CPI de 2,0.
$\mathcal{B}$: cycle d'horloge de 20 ns et CPI de 1,2.
Quelle machine est la plus rapide ?
On a :

$$N_c^{\mathcal{A}}(p) = N_i(p) \times 2.0$$

et

$$N_c^{\mathcal{B}}(p) = N_i(p) \times 1.2$$

Donc :

$$\begin{aligned} T_e^{\mathcal{A}}(p) &= N_c^{\mathcal{A}}(p) \times T_c^{\mathcal{A}} \\ &= N_i(p) \times 2.0 \times 10 \\ &= N_i(p) \times 20 \end{aligned}$$

et

$$\begin{aligned} T_e^{\mathcal{B}}(p) &= N_c^{\mathcal{B}}(p) \times T_c^{\mathcal{B}} \\ &= N_i(p) \times 1.2 \times 20 \\ &= N_i(p) \times 24 \end{aligned}$$

Performances

Temps d'exécution d'un programme

La performance relative est donc

$$\frac{T_e^{\mathcal{A}}(p)}{T_e^{\mathcal{B}}(p)} = \frac{N_i(p) \times 24}{N_i(p) \times 20} = 1.2$$

C'est la machine $\mathcal{B}$ qui est plus rapide (alors même que son horloge est plus lente).

Performances

Temps d'exécution d'un programme

On utilise parfois le compte exact du nombre de cycles nécessaires pour chacune des instructions et non pas une moyenne. Alors :

$$N_c(p) = \sum_{j=1}^{n} (i_j)$$

en général les instructions sont classées, par exemple en fonction de leur durée d'exécution.
Comparaison :
On dispose d'une machine ayant trois classes d'instructions :

Classe	CPI
A	1
B	2
C	3

Une même fonction est écrite par deux programmeurs différents. Le comptage des instructions de chaque classe est le suivant :

	Instructions de classe		
Programmeur	A	B	C
Robert	2	1	2
Jules	4	1	1

Performances

Temps d'exécution d'un programme

Quel programmeur utilise le plus d'instructions ? Quel programme est le plus rapide ? Quel est le CPI de chaque séquence ?
Robert utilise $2 + 1 + 2 = 5$ instructions.
Jules utilise $4 + 1 + 1 = 6$ instructions.
Robert est économe.

Nombre de cycles utilisés par Robert :
$N_c^{\text{robert}} = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10.$
Nombre de cycles utilisés par Jules :
$N_c^{\text{jules}} = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9.$
Jules est plus rapide.

$$\begin{aligned} CPI_{\text{robert}} &= \frac{N_c^{\text{robert}}}{N_i(p_{\text{robert}})} \\ &= \frac{10}{5} = 2 \end{aligned}$$

$$\begin{aligned} CPI_{\text{jules}} &= \frac{N_c^{\text{jules}}}{N_i(p_{\text{jules}})} \\ &= \frac{9}{6} = 1.5 \end{aligned}$$

Performances

La mesure MIPS

MIPS : Million d'Instructions Par Seconde (Millions of Instructions Per Second). Donc :

$$\begin{aligned} \text{MIPS} &= \frac{N_i(p)}{T_e(p) \times 10^6} \\ &= \frac{N_i(p)}{N_c(p) \times T_c \times 10^6} \\ &= \frac{N_i(p) \times F_h}{N_i(p) \times CPI \times 10^6} \\ &= \frac{F_h}{CPI \times 10^6} \end{aligned}$$

Coïncide avec l'intuition : Fréquence élevée, indice MIPS élevé.

Performances

La mesure MIPS

Comparaison :
Soit une machine avec trois classes d'instructions (la même que tout à l'heure) dont l'horloge bat à la fréquence de 100Mhz. Une même fonction est implantée par deux programmeurs différents. Le comptage des instructions donne :

	Instructions (en millions) par classe		
Programmeur	A	B	C
Robert	5	1	1
Jules	10	1	1

Quel programme est le plus rapide avec l'indice MIPS ? en comptant le temps d'exécution ?

$$MIPS = \frac{F_h}{CPI \times 10^6} = \frac{100}{CPI \times 10^6}$$

$$CPI = \frac{N_c(p)}{N_i(p)}$$

Performances

La mesure MIPS

Donc :

$$CPI_{\text{robert}} = \frac{(5 \times 1 + 1 \times 2 + 1 \times 3) \times 10^6}{(5 + 1 + 1) \times 10^6} = \frac{10}{7} \approx 1.428$$

et

$$MIPS_{\text{robert}} = \frac{100}{1.428 \times 10^6} = 70$$

$$CPI_{\text{jules}} = \frac{(10 \times 1 + 1 \times 2 + 1 \times 3) \times 10^6}{(10 + 1 + 1) \times 10^6} = \frac{15}{12} \approx 1.25$$

et

$$MIPS_{\text{jules}} = \frac{100}{1.25 \times 10^6} = 80$$

Jules a un processeur au débit plus important.

Performances

La mesure MIPS

On sait que :

$$T_e(p) = \frac{N_i(p) \times CPI}{F_h}$$

Donc :

$$\begin{aligned} T_e(p_{\text{robert}}) &= \frac{(5+1+1) \times 10^6 \times 1.428}{100 \times 10^6} \\ &= \frac{7 \times 1.428}{100} = 0.10 \text{ secondes} \end{aligned}$$

$$\begin{aligned} T_e(p_{\text{jules}}) &= \frac{(10+1+1) \times 10^6 \times 1.25}{100 \times 10^6} \\ &= \frac{12 \times 1.25}{100} = 0.15 \text{ secondes} \end{aligned}$$

Mais c'est Robert qui est plus rapide.

Performances

La mesure MFLOPS

MFLOPS : Million d'opérations en virgule flottante par seconde (Million FLoating-point Operation Per Second).
L'argument principal est d'affirmer que cette mesure est bien meilleure que MIPS, car les opérations en virgule flottante sont plus ≪ indépendantes de la machine ≫. Malheureusement la machine CRAY-2 ne possède pas d'opération de division en virgule flottante, alors que le MOTOROLA 68882 possède une division, une racine carrée, un sinus, un cosinus...
On notera qu'aujourd'hui on parle de GigaFlops voire de PetaFlops.

- Andrew Tanenbaum, *Architecture de l'ordinateur*, InterEditions,1991.
- David A. Patterson & John L. Hennessy, *Computer Organization & Design : The hardware/software interface*. Morgan Kaufmann Publishers, Inc, 1994.
- David A. Patterson & John L. Hennessy, *Computer Architecture : A quantitative approach*, Morgan Kaufmann Publishers, Inc, 1990.
- William Barden, *The Z80 Microcomputer Handbook*, Howard W. Sams & Co., Inc., 1978.
- *Éléments pour une histoire de l'informatique*, Traduction d'œuvres choisies de Donald E. Knuth. Traduction dirigée par Patrick Cégielski. Société Mathématique de France - CSLI Publications, Stanford, California. 2011. isbn : 978-1-57586-622-2
- Philippe Breton, *Une histoire de l'informatique*, collection ≪ Points Sciences ≫ (réimpr. 1990), 261 p. (ISBN 2020123487)
- Georges Ifrah, *Histoire Universelle des Chiffres*, Robert Laffont. 1994. Deux tomes.
- Raúl Rojas, Ulf Hashagen (Eds.). *The First Computers : History and Architectures*. 2000. MIT Press.