Entrées/Sorties

L'ordinateur doit pourvoir communiquer avec des unités externes (disque, clavier, écran, . . .)

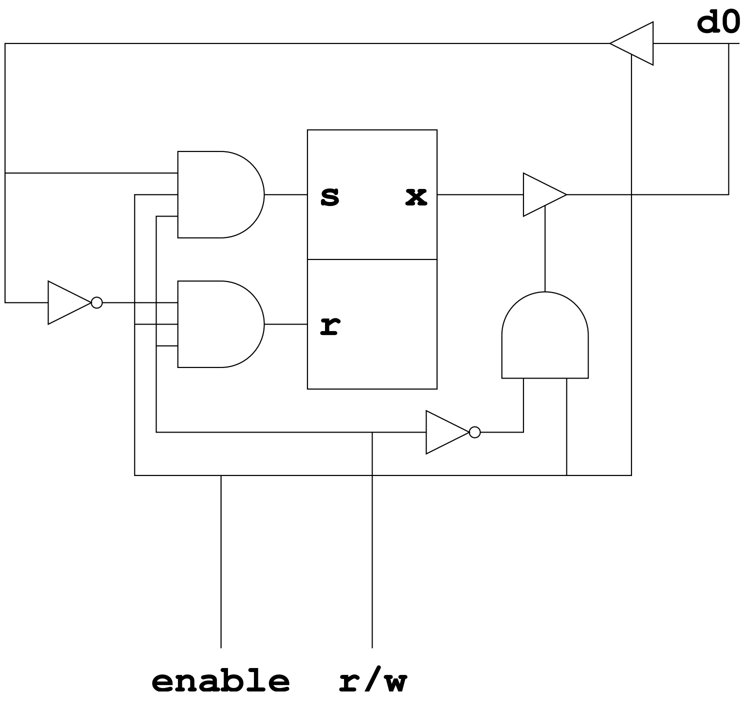
Les unités sont banchées sur le bus d'adresses et le bus de données

Elles se comportent de la même façon que la mémoire principale

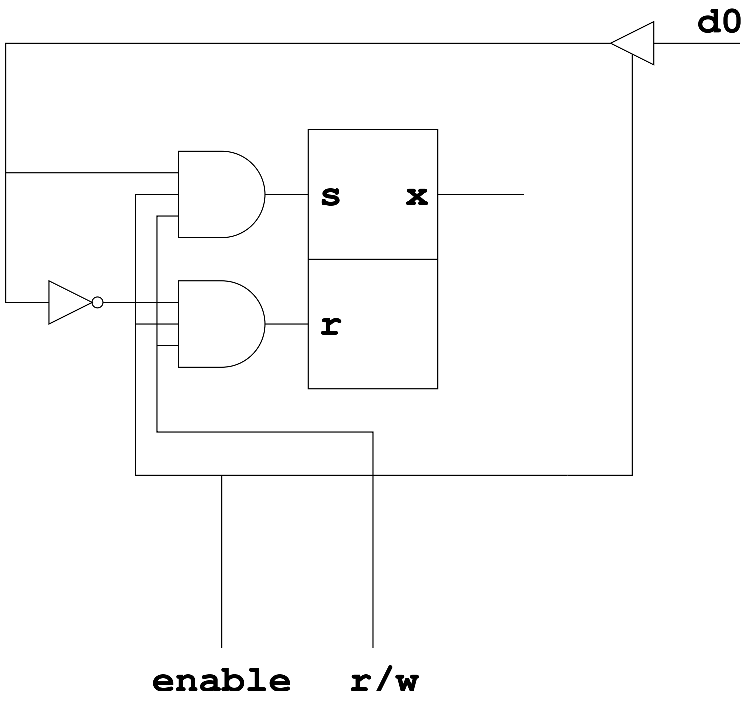
De cette manière, on peut utiliser l'instruction st pour la sortie, et l'instruction ld pour l'entrée de données

Sortie

Une cellule de mémoire:

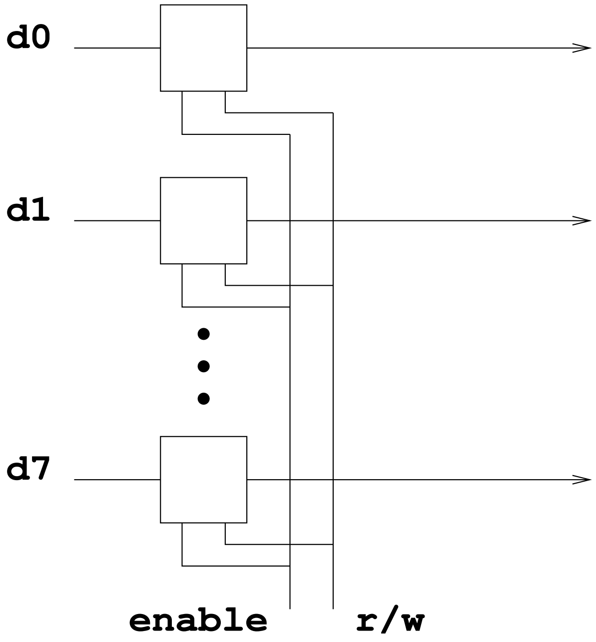


Une cellule de registre de sortie:



Sortie (suite)

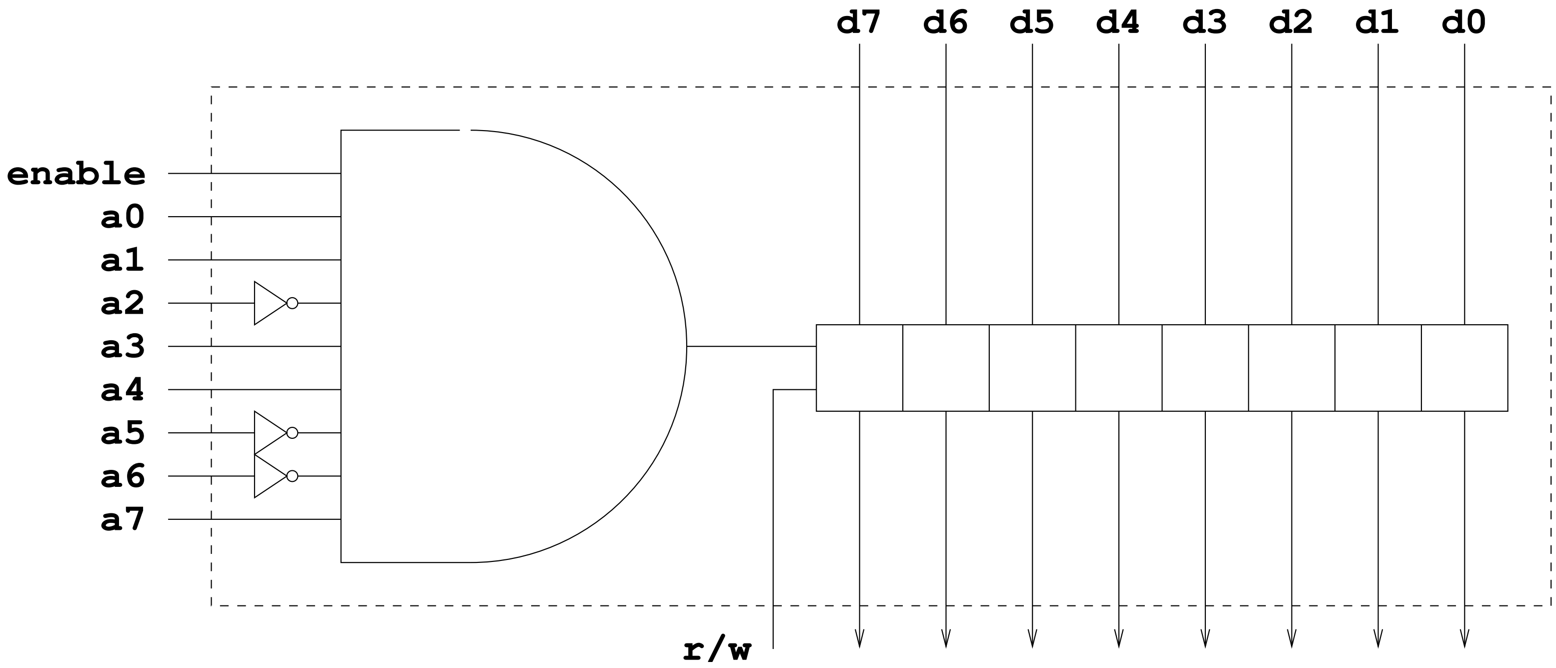
Pour avoir un registre à (par exemple) 8 bit, il suffit d'en mettre 8 en parallèle:



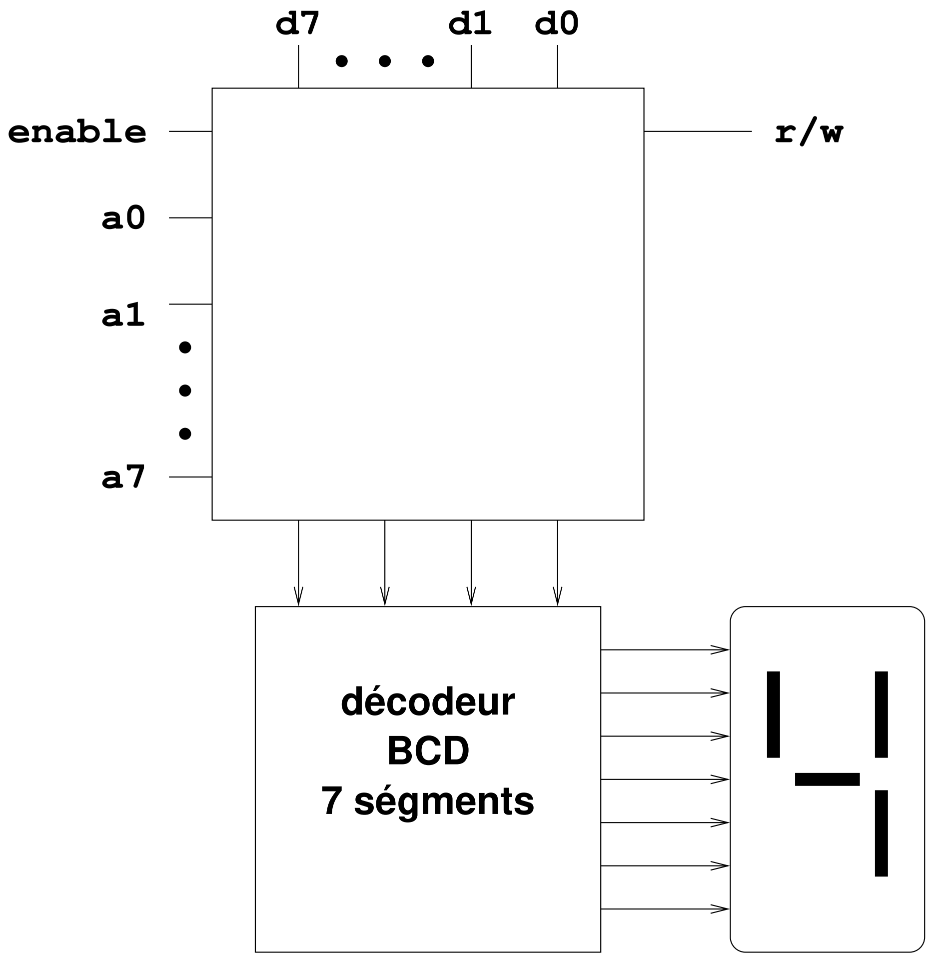
Sortie (suite)
Décodage d'adresses

Pour le décodage d'adresses, il faut une porte et des inverseurs.

Exemple: nous souhaitons utiliser l'adresse 10011011

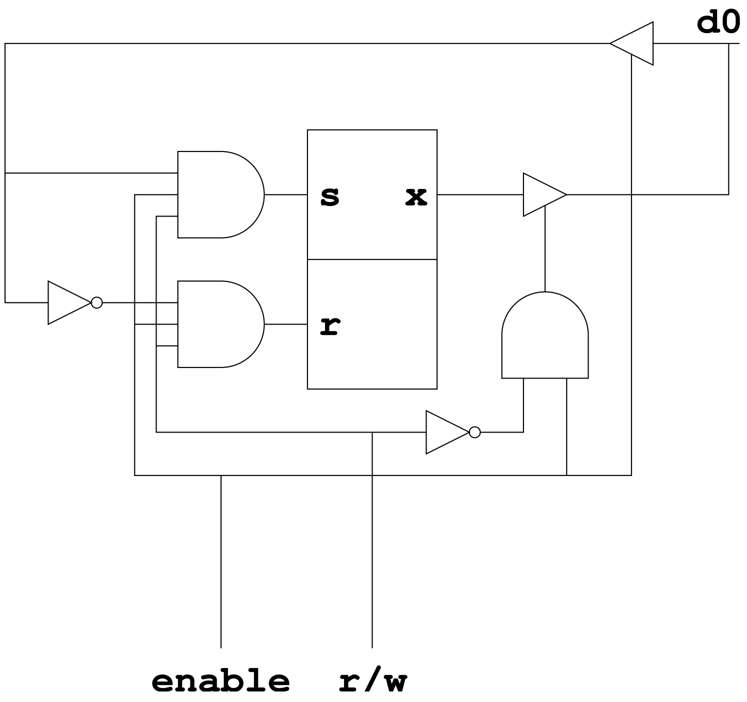


Sortie (suite)
Exemple

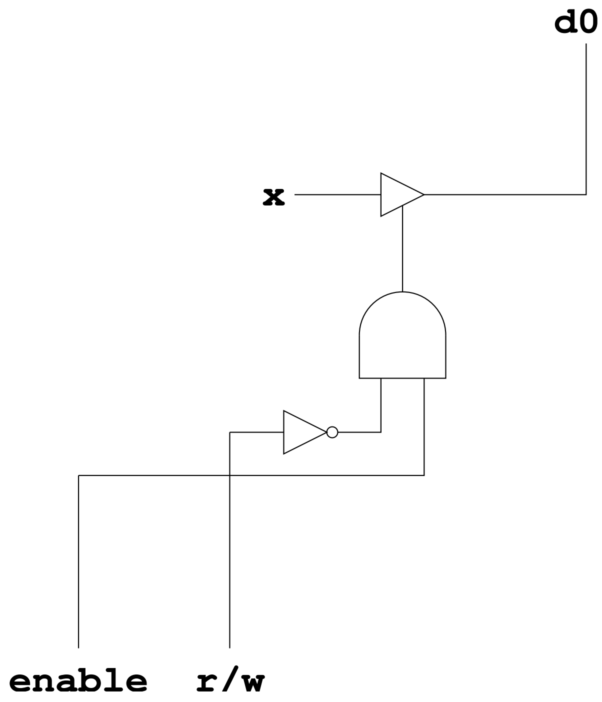


Entrée

Une cellule de mémoire:

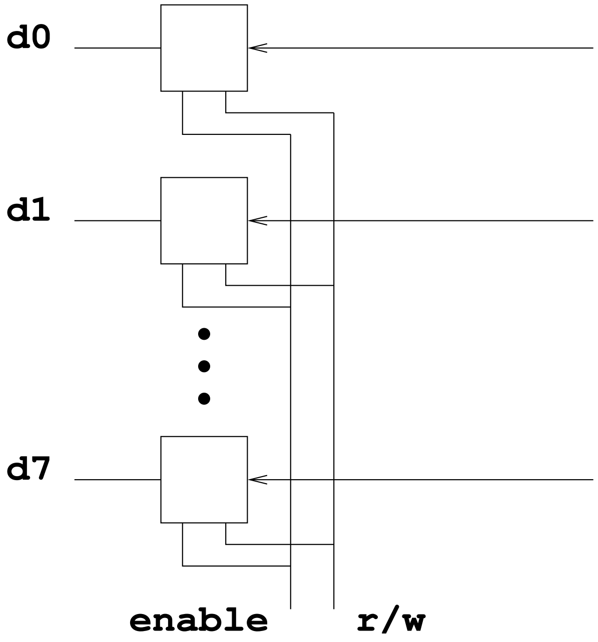


Une cellule d'entrée:



Entrée (suite)

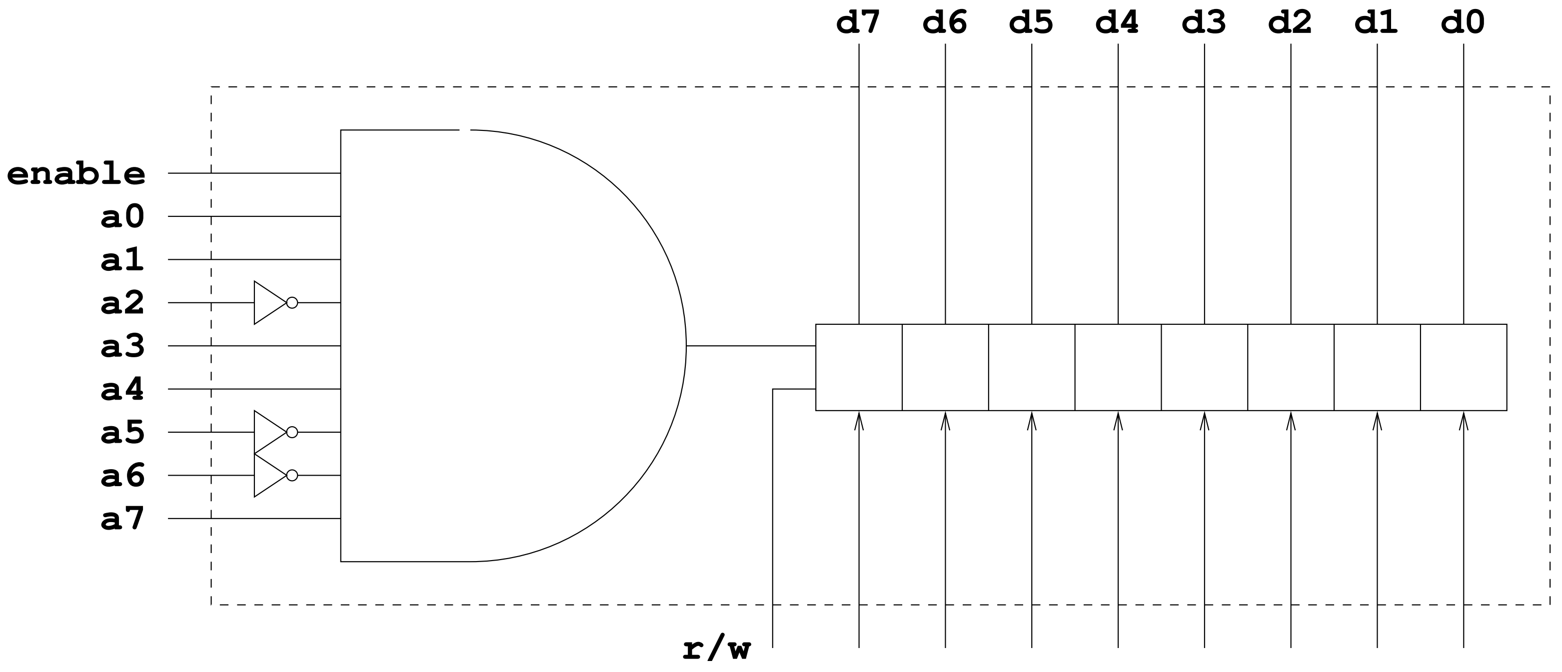
Pour avoir (par exemple) 8 bit, il suffit d'en mettre 8 en parallèle:



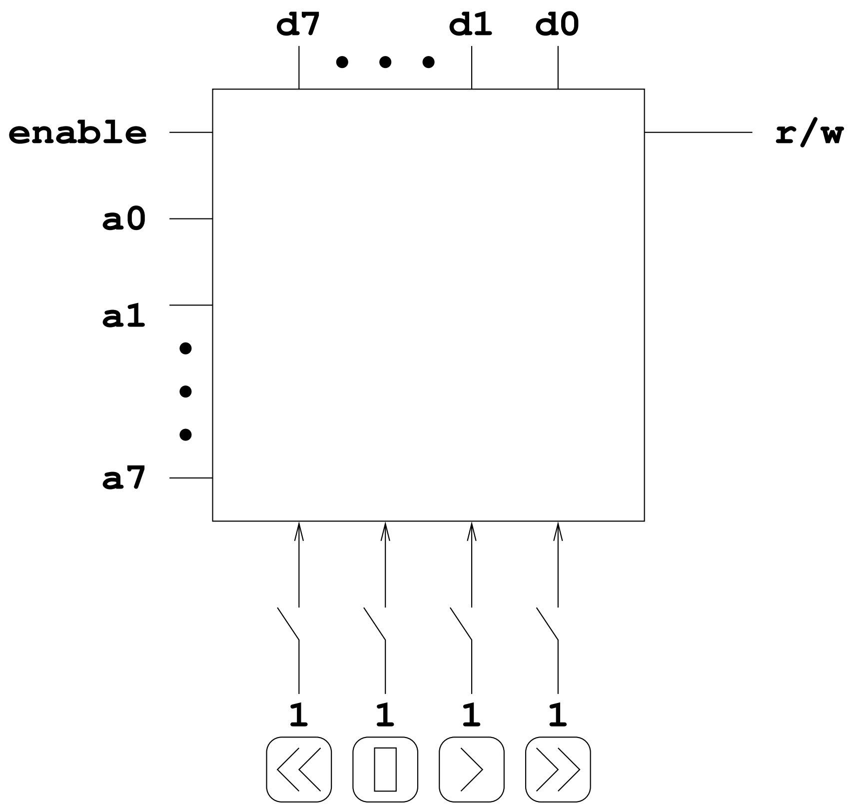
Entrée (suite)
Décodage d’adresses

Pour le décodage d’adresses, il faut une porte et des inverseurs.

Exemple: nous souhaitons utiliser l’adresse 10011011



Entrée (suite)
Exemple



Entrée (suite)

Problème:

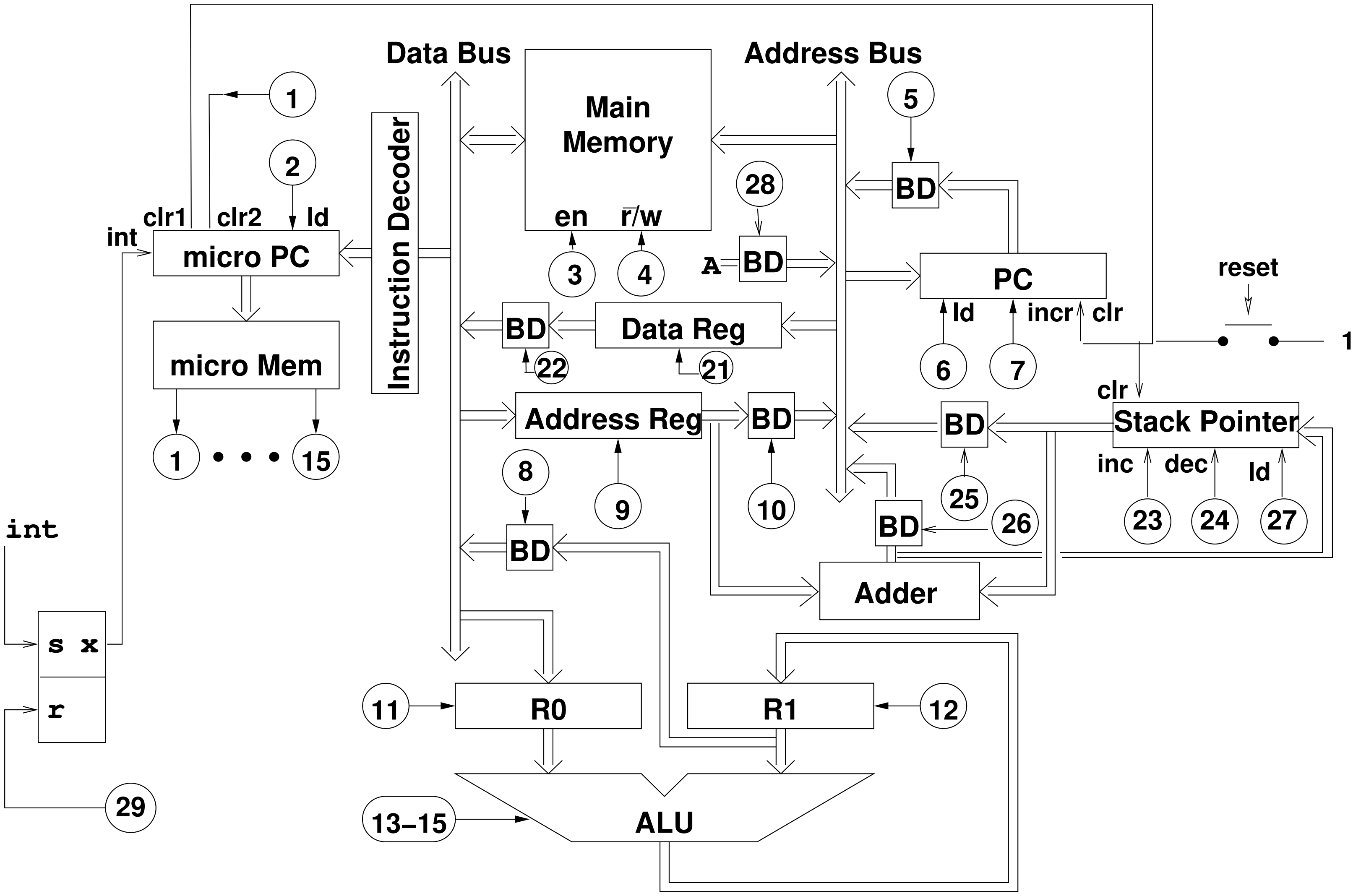
L'unité centrale doit tester périodiquement la disponibilité d'entrée.

Solution:

Interruptions. L'unité externe sera capable d'interrompre le traitement normal de l'unité centrale.

C'est comme une instruction jsr provoquée par un événement externe

Modifications pour les interruptions



Modification de micro PC

clr1	clr2	int	ld	
1	–	–	–	zéro
0	0	0	0	incrémentation
0	0	0	1	chargement
0	0	1	0	incrémentation
0	0	1	1	chargement
0	1	0	0	zéro
0	1	0	1	ne peut pas arriver
0	1	1	0	chargement d'une constante c
0	1	1	1	ne peut pas arriver

La constante c correspond à la première adresse en micro mémoire de l'implementation du micro programme pour traiter les interruptions

Micro programme pour les interruptions

29	5	21	24
25	22	4	
25	22	4	3
25	22	4	
28	6	1	

Différences entre notre architecture et une architecture réelle

Principalement deux types d'architecture:

- **CISC (Complex Instruction Set Computer)**
  - **Pentium, Vax, 68000, . . .**

- **RISC (Reduced Instruction Set Computer)**
  - **MIPS, SPARC, PowerPC, Alpha, . . .**

CISC/RISC

CISC	RISC
Relativement peu de registres	Beaucoup de registres
Plusieurs modes d’adressage (immediate, direct, indirect, offset, indexed, indexed indirect, . . .)	Peu de mode d’adressage utilisation de load/store et load immediate
Format d’instructions irrégulier	Format d’instruction régulier
Instructions souvent très complexes	Instructions simples
Passage d’arguments dans la pile	Passage d’arguments dans des registres
JSR empile l’adresse	JSR copie PC dans un registre
Registres souvent asymétriques	Registres souvent symétriques

Optimisations

L’exécution d’une instruction peut commencer avant que l’exécution de l’instruction précédente ne soit terminée (pipeline)

Exécution d’une instruction simple en un cycle d’horloge

Plusieurs unité d’exécution d’instructions permettant à plusieurs instructions d’exécuter en parallèle

Unité flottante séparée permettant l’exécution d’une opération flottante et d’une autre opération en même temps

Opérations sur les flottants très rapides (addition en < 3 cycles, multiplication en < 5 cycles)

Prédiction de branchements (= sauts) permettant l’exécution d’une instruction jump en un cycle d’horloge

Prédiction de l’adresse de retour de l’instruction ret par une pile interne

Instructions pour les langages haut niveau

Registre supplémentaire (bp ou ap) pour les arguments d'une fonction

Instructions link/unlik pour allouer/desallouer les variables locales