Numérique et Sciences Informatiques
Chapitre VIII - Architecture de Von Neumann

Classe de première

I. Le modèle de Von Neumann

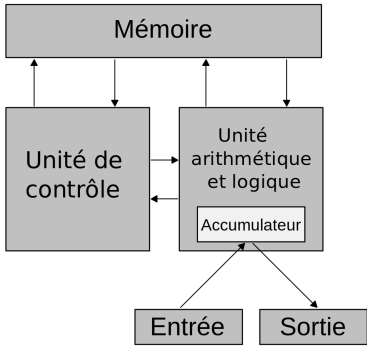
I.1. Histoire

En 1936, la machine universelle d'Alan Turing et le lambda-calcul du mathématicien Alonzo Church sont proposés comme base de la définition de l'algorithme.

C'est en 1944 que John Von Neumann invente le modèle qui porte son nom et qui reste encore d'actualité (à quelques évolutions près). Ce modèle représente encore l'architecture de nos ordinateurs.

I.2. Le modèle

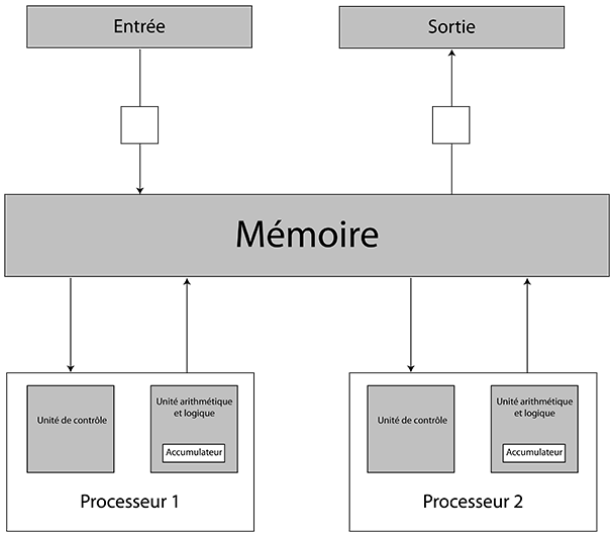
Le modèle de Von Neumann peut être représenté par le schéma ci-dessous.



Les différents constituants :

- L'unité de contrôle (UC) : elle est en charge du séquençage des opérations (les faire les unes après les autres)
- L'unité arithmétique et logique (UAL) : elle effectue les opérations de base
- L'accumulateur : c'est une petite mémoire proche du processeur. Il possède plusieurs registres.
- La mémoire : elle contient les données et le programme
- L'entrée : permet de récupérer une donnée extérieure
- La sortie : permet d'envoyer une donnée à l'extérieur

La seule différence avec les ordinateurs actuels réside dans le fait qu'il y ait plusieurs unité de contrôle (UC) et unités arithmétiques et logiques (UAL) connectées à la mémoire. De plus les entrées et sorties sont directement connectées à la mémoire sous le contrôle de processus indépendants.
On a alors le schéma suivant :



Source : interstices

II. Langage machine

Le **langage machine** est la suite de bits qui est interprétée par le processeur d'un ordinateur exécutant un programme informatique. C'est le langage natif d'un processeur, c'est-à-dire le seul qu'il puisse traiter. Il est composé d'instructions et de données à traiter codées en binaire.

Pour plus d'informations, voir sur le site Wikipedia

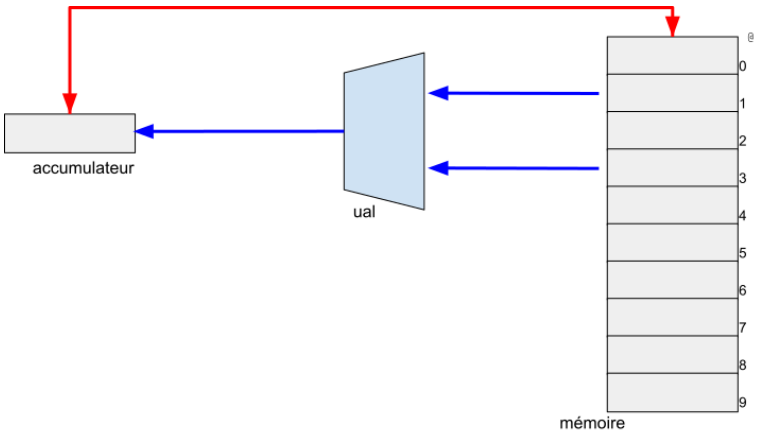
III. Exercices

III.1. Exercice 1

M10 est une petite machine informatique constituée :

- d'une mémoire qui contient des données,
- d'une UAL, unité arithmétique et logique capable de réaliser une opération telle une addition,
- d'un accumulateur qui recevra le résultat des opérations réalisées par l'UAL.

La mémoire est composée de 10 "cases". Ces cases sont numérotées de 0 à 9. Ce numéro est l'**adresse** de la case.



Le M-10 peut faire deux choses :

- Transférer les valeurs de l'accumulateur vers une case mémoire, et inversement.
- Faire des calculs grâce à l'unité arithmétique et logique.

Voici la liste des instructions disponibles sur le M-10 :

	Instructions	Commentaires
instructions de Transfert de données	`LOAD @1`	transfère une donnée de la mémoire d'adresse `@1` vers l'accumulateur
	`STORE @1`	transfère une donnée de l'accumulateur vers la mémoire d'adresse `@1`
instructions de calculs	`ADD @1 @2`	ajoute les valeurs stockées dans la mémoire aux adresse `@1` et `@2` : `@1+@2`
	`SUB @1 @2`	soustrait les valeurs stockées dans la mémoire aux adresse `@1` et `@2` : `@1-@2`
	`MUL @1 @2`	multiplie les valeurs stockées dans la mémoire aux adresse `@1` et `@2` : `@1×@2`
	`DIV @1 @2`	divise les valeurs stockées dans la mémoire aux adresse `@1` et `@2` : `@1÷@2`

1. (a) Placer un nombre en mémoire à l'adresse 0.
   (b) Écrire une séquence d'instructions à donner au M-10 pour que ce nombre soit recopié en mémoire à l'adresse 5.

2. (a) Placer un nombre en mémoire à l'adresse 0.
   (b) Écrire une séquence d' instructions à donner au M-10 pour mettre en mémoire à l'adresse 9 le carré du nombre placé à l'adresse 0.
   (c) Écrire une séquence d' instructions à donner au M-10 pour mettre en mémoire à l'adresse 9 le cube du nombre placé à l'adresse 0.

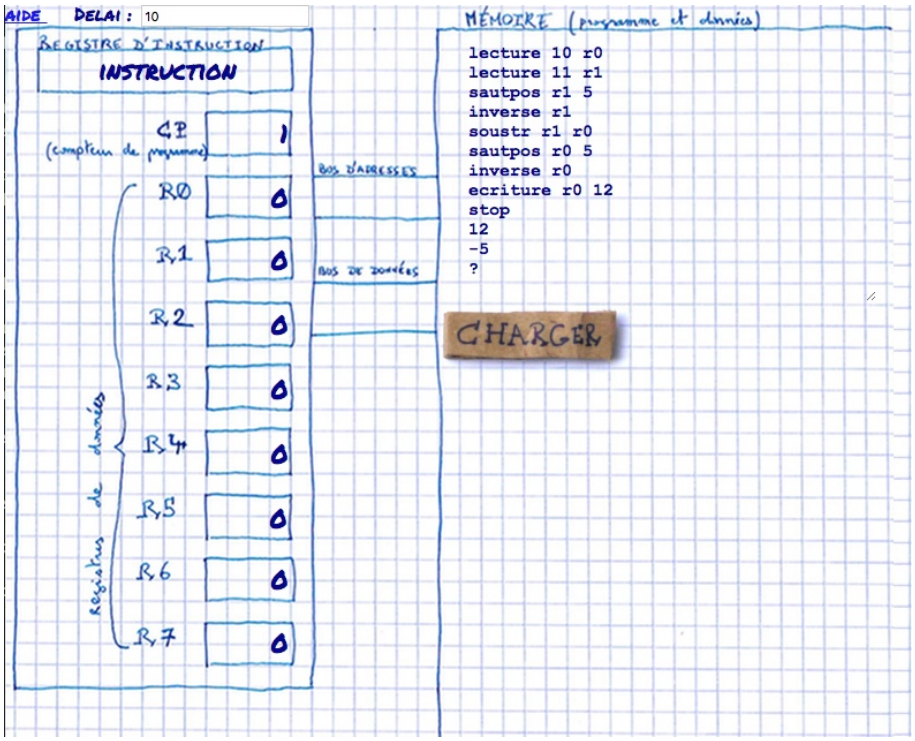
3. (a) Placer un nombre en mémoire à l'adresse 0 et un autre à l'adresse 3.
   (b) Écrire une séquence d' instructions à donner au M-10 pour mettre en mémoire à l'adresse 9 le double de la somme du nombre placé à l'adresse 0 et de 3.

III.2. Exercice 2

AMIL permet de simuler un processeur disposant de :

- 8 registres de données `R0` à `R7`
- un registre compteur de programme `CP` indiquant le numéro de la ligne de la prochaine instruction à exécuter
- un registre instruction indiquant l'instruction en cours d'exécution
- une unité arithmétique effectuant des calculs entiers et flottants

Le processeur est relié par des bus d'adresse et de données à une mémoire contenant le programme (des instructions) et des données (des valeurs numériques).



Les instructions disponibles sont les suivantes :

Mnémonique	Détail	Action
stop	Arrête l'exécution du programme.	
noop	N'effectue aucune opération.	
saut i	Met le compteur ordinal à la valeur i.	PC ← i
sautpos ri j	Si la valeur contenue dans le registre i est positive ou nulle, met le compteur ordinal à la valeur j.	si ri 0 PC ← j sinon PC ← PC+1
valeur x ri	Initialise le registre i avec la valeur x.	ri ← x
lecture i rj	Charge, dans le registre j, le contenu de la mémoire d'adresse i.	rj ← men(i)
lecture *ri rj	Charge, dans le registre j, le contenu de la mémoire dont l'adresse est la valeur du registre i.	rj ← men(ri)
ecriture ri j	Écrit le contenu du registre i dans la mémoire d'adresse j.	ri → men(j)
ecriture ri *rj	Écrit le contenu du registre i dans la mémoire dont l'adresse est la valeur du registre j.	ri → men(rj)
inverse ri	change le signe du contenu du registre i.	ri ← − ri
add ri rj	Ajoute la valeur du registre i à celle du registre j.	rj ← ri + rj
soustr ri rj	Soustrait la valeur du registre i à celle du registre j.	rj ← rj − ri
mult, div, et	Même syntaxe que pour add et soustr mais pour la multiplication, la division entière et le et bit à bit.	rj ← rj (*, /, and) ri

Les question numériques sont indépendantes.

1. Écrire un programme qui lit un nombre en entrée à l'adresse 5 et le restitue en sortie à l'adresse 6.
2. (a) Ecrire un nombre à l'emplacement mémoire 12 et un autre à l'emplacement mémoire 10.
   (b) Écrire un programme qui écrit (si nécessaire) à l'emplacement mémoire 12 le maximum des deux nombres.
3. (a) Écrire le nombre 20 à l'emplacement mémoire 18.
   (b) Écrire à partir de la ligne 20 une suite de nombres positifs.
   (c) Écrire à la suite de ces nombres un nombre négatif.
   (d) Écrire un programme qui recopie cette suite de nombres positifs juste après le nombre négatif.

III.3. Exercice 3

Le fonctionnement du processeur M999 est caractéristique des architectures dites Von Neumann.
M999 est constitué :

- d'une mémoire qui contient à la fois les données et le programme
- d'une unité arithmétique et logique – UAL, ALU en anglais – en charge de réaliser les opérations telles addition, comparaison...
- d'une unité de commande – ou unité de contrôle – qui pilote l'ordinateur
- de dispositifs d'entrée-sortie.

La mémoire et les registres
La mémoire est composée de 100 mots mémoire de 3 chiffres (valeur de 000 à 999). Ces 100 mots mémoire sont adressables par des adresses codées sur 2 chiffres.

Cette mémoire va contenir données et instructions.

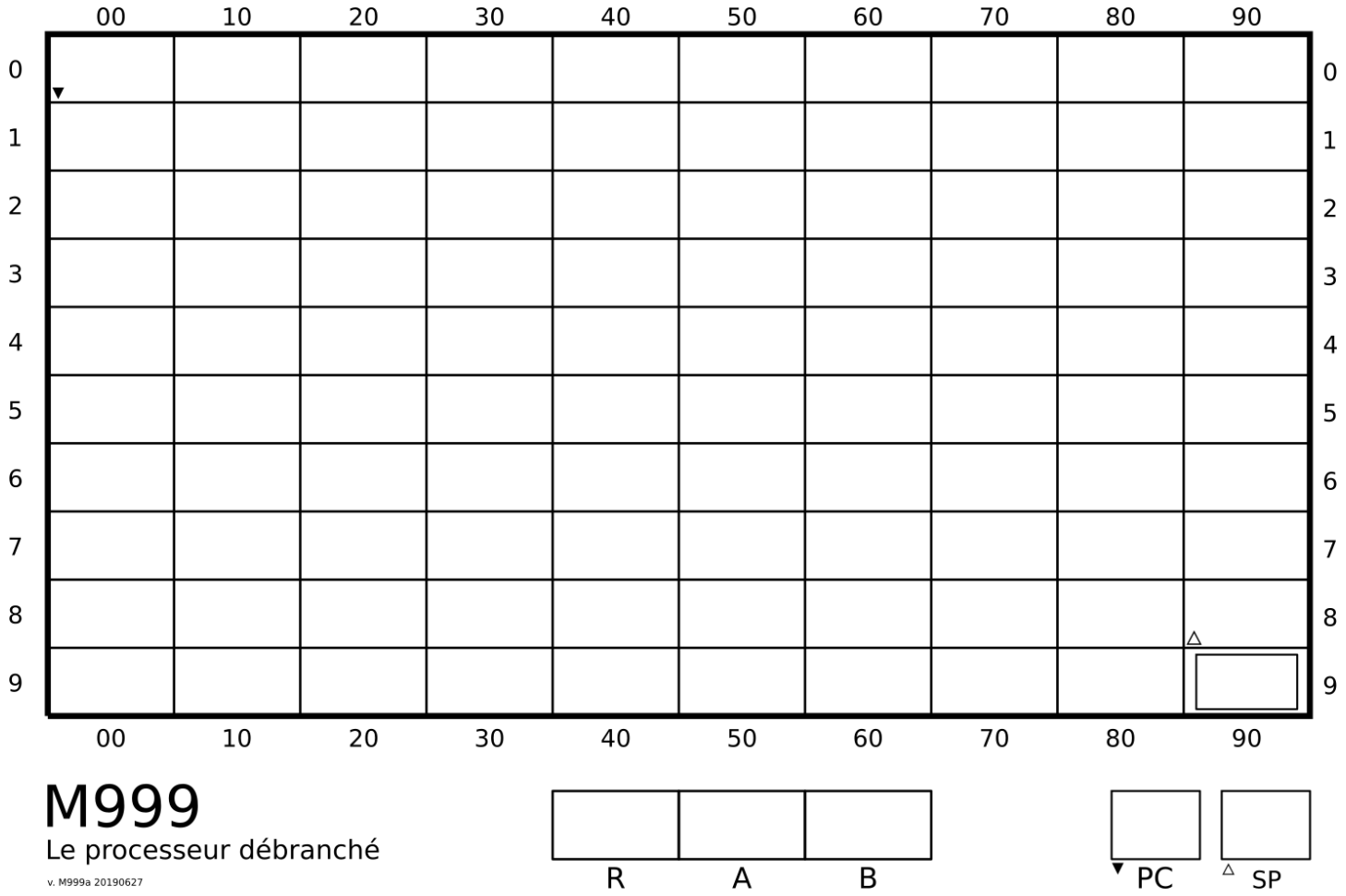
Le processeur dispose de deux registres généraux A et B, et d'un registre accumulateur/résultat R.

Comme la mémoire, ces registres sont de 3 chiffres, et contiennent donc des valeurs entre 0 et 999.

Le processeur dispose aussi d'un registre pointeur d'instruction PC contenant l'adresse mémoire de la prochaine instruction à exécuter.

La mémoire et les registres peuvent être matérialisés par une grille de 10*10 et des cases supplémentaires pour les registres A, B, R.

Le registre PC peut être matérialisé par un "pion" situé sur une des cases de la grille mémoire.



Unité arithmétique et logique
L'unité arithmétique et logique est en charge d'effectuer les calculs. Les opérandes sont dans les registres A et B, et résultats dans le registre R.

Unité de commande
L'unité de commande pilote l'ordinateur.

Son cycle de fonctionnement comporte 3 étapes :

- charger l'instruction depuis la mémoire pointée par PC. Incrémenter PC.
- décoder l'instruction : à partir des 3 chiffres codant l'instruction, identifier quelle est l'opération à réaliser, quelles sont les opérandes.
- exécuter l'instruction.

Jeu d'instruction

op0	op1 op2	mnémonique	instruction à réaliser
0	*addr*	`LDA`	copie le mot mémoire d'adresse *addr* dans le registre A
1	*addr*	`LDB`	copie le mot mémoire d'adresse *addr* dans le registre B
2	*addr*	`STR`	copie le contenu du registre R dans le mot mémoire d'adresse *addr*
3	- -	-	**opérations arithmétiques et logiques**
3	00	`ADD`	Ajoute les valeurs des registres A et B, produit le résultat dans R
3	01	`SUB`	Soustrait la valeur du registre B à celle du registre A, produit le résultat dans R
3	..	*etc.*	...
3	99	*NOP*	ne fait rien
4	*rs rd*	*MOV*	copie la valeur du registre source *rs* dans le registre destination *rd*
5	*addr*	*JMP*	branche en *addr*, c'est-à-dire que PC reçoit la valeur *addr*
6	*addr*	*JPP*	branche en *addr* si la valeur du registre R est positive

Les registres *rs* et *rd* lorsque op0 vaut 4 sont désignés par les valeurs suivantes :

valeur	registre
0	A
1	B
2	R

Boot et arrêt

La machine démarre avec la valeur nulle comme pointeur d'instruction.
La machine stoppe si le pointeur d'instruction vaut 99.
On peut donc utiliser le mnémonique 'HLT' comme synonyme de 'JMP 99'.

Entrée/sorties

Les entrées/sorties sont "mappées" en mémoire.
Écrire sur le mot mémoire 99 écrit sur le terminal.
Les valeurs saisies sur le terminal seront lues dans le mot mémoire 99.

1. Soit l'état suivant de la mémoire.

	00	01	02 ...
0	399	599	...
1	011	123	...
2	112	042	...
3	301	...	...
4	608	...	...
5	402	...	...
6	299	...	...
7	599	...	...
8	412	...	...
9	299	...	...

   Que provoque la mise en route de la machine ?

2. Même question avec l'état suivant , en supposant qu'il y a en entrées successivement les nombres 100, 200, 300 avec un changement dès que l'adresse 99 est utilisée.

	00	01	02 ...
0	514	599	412
1	011	123	299
2	112	042	599
3	301	531	...
4	608	099	...
5	402	199	...
6	299	301	...
7	599	620	...
8	412	402	...
9	299	521	...

3. Créer un programme ajoute deux nombres donnés en entrée et renvoie le résultat en sortie.

4. Créer un programme qui renvoie en sortie le reste de la division euclidienne de deux nombres donnés en entrée.