Eléments d'Informatique

Cours 3- Architecture des ordinateurs, langage machine, langage assembleur, AMIL.

Catherine Recanati

UNIVERSITÉ PARIS 13 NORD

Plan général

- **Représentation des nombres. Notion de variable.**
- **Programme. Expressions.**
- **Architecture des ordinateurs: langage machine, langage assembleur, AMIL.**
- Systèmes d'exploitation : fichiers, processus, compilation.
- Instructions de contrôle: boucles et branchements.
- Programme, définition de fonction, appel fonctionnel.
- Tableaux de variables et fonctions d'arguments de type tableau.
- Sens d'un programme, pile d'exécution, compilation.
- Pointeurs et tableaux.
- Chaines de caractères, bibliothèque <string.h>.
- Allocation dynamique, liste chaînées.
- Révisions.

- Cours 3 – Architecture des ordinateurs, langage machine, langage assembleur, AMIL.

- Architecture des ordinateurs
- Niveau de langages
- Mini assembleur AMIL

Architecture de *von Neumann*

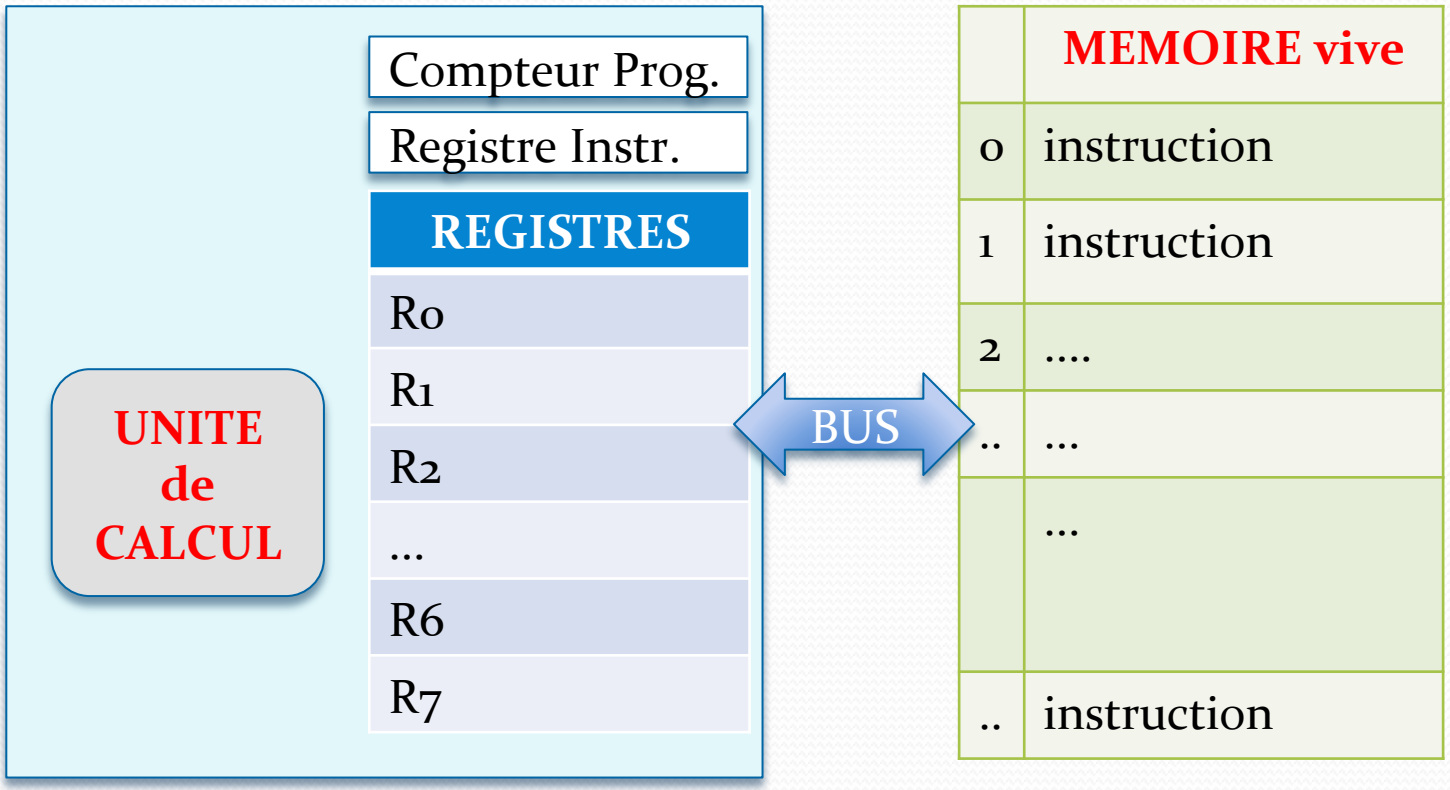
John William Mauchly et *John Eckert* y ont aussi beaucoup contribué. Cette machine, ancêtre de nos processeurs, permet d'exécuter un programme stocké en mémoire. Elle est faite :

- d'une mémoire (cases numérotées)
- d'une Unité de Calcul (UC)
- de registres numérotés (+ le Compteur de Programme et le Registre d'Instruction)
- d'un bus (circuits pour les échanges entre les registres et la Mémoire)

Plan

- Architecture des ordinateurs
- Niveaux de langages
- Mini assembleur AMIL

Architecture de *von Neumann*



Architecture de *von Neumann*

Le programme (chargé en mémoire vive) est un ensemble d'instructions (codées en binaire). Il va s'exécuter en réitérant un cycle :

- lire le registre compteur de programme (CP) qui contient l'adresse du mot mémoire représentant la prochaine instruction.
- transférer ce mot dans le registre d'instruction (RI)
- incrémenter le compteur de programme CP
- décoder l'instruction de RI, et l'exécuter (=> modifie les registres ou une case mémoire)

Niveaux de langages

Les cases mémoires et les registres contiennent des mots mémoire : des suites de n bits, où n est fixé par l'architecture matérielle (souvent 16, 32, ou 64 bits).

Les instructions du langage machine sont écrites en binaire sur un mot.

Par exemple:

111 0 0 0 1 0 (ici, mot de 8 bits)

Les 3 premiers bits permettront de coder l'instruction (par exemple ajouter au registre R1, le contenu de la mémoire indiquée par les 5 derniers bits).

Niveaux de langages

Les suites d'instructions binaires de la mémoire sont écrites en langage machine.

Un langage assembleur est un langage machine, traduit de manière symbolique pour être compris par des humains.

Ainsi, au lieu de noter l'instruction précédente

```
1 1 1  0 0 0 1 0
```

on l'écrira par exemple

```
add r1 2
```

signifiant ajouter à `r1` le contenu de la mémoire d'adresse `2`.

Niveaux de langages

Bien qu'écrit de manière symbolique, un langage assembleur ne permet pas de déclarer des variables. Le type d'instruction manipulée est un simple jeu d'opérations arithmétiques entre les mémoires et les registres.

Un langage de programmation comme le C est, par contraste, dit « de haut niveau », ou « évolué », car il permet de déclarer des variables, de définir des fonctions, et il possède des instructions de contrôle plus sophistiquées que les sauts.

Mini assembleur AMIL

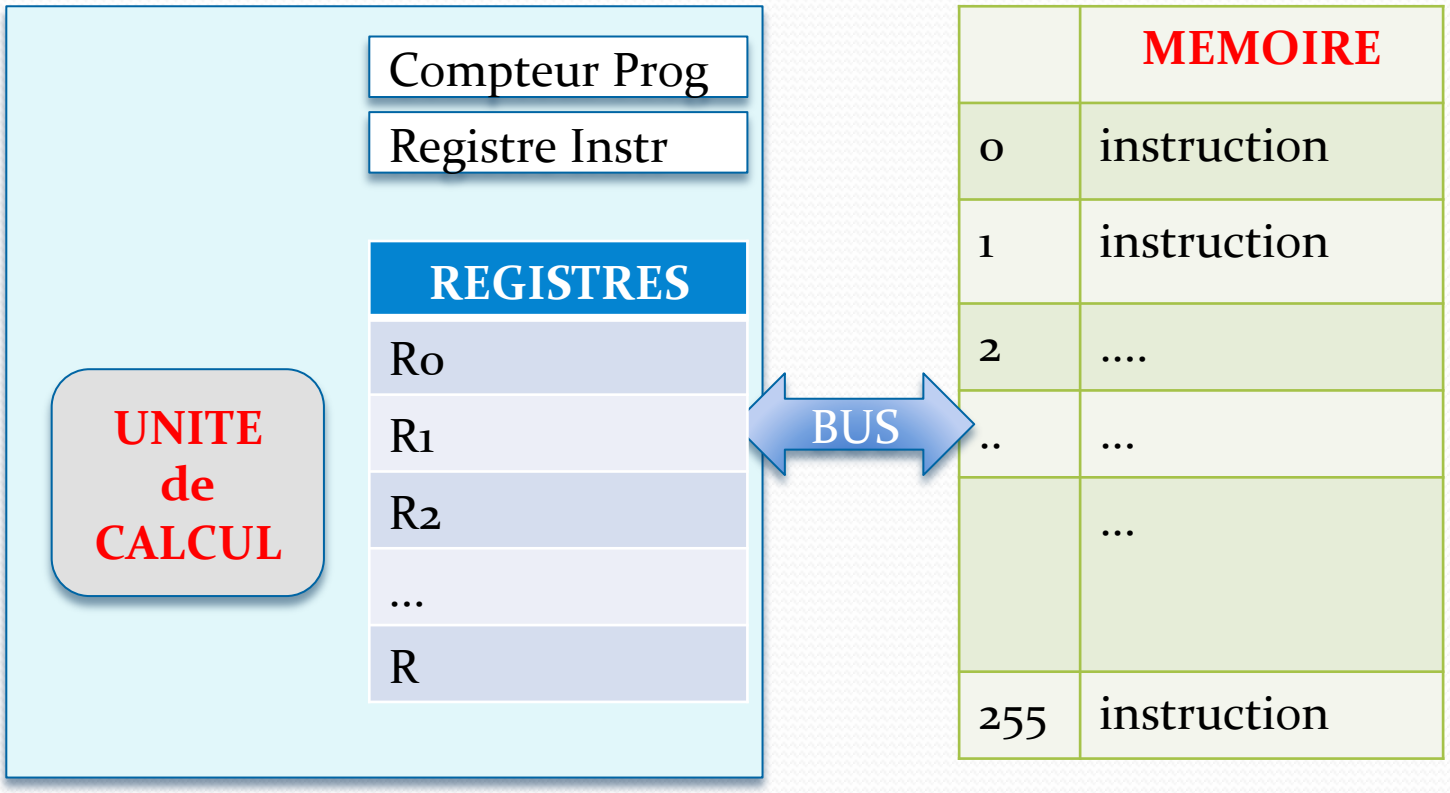
Le mini assembleur AMIL est un langage assembleur imaginaire, purement pédagogique, destiné à illustrer les principes d'une architecture de von Neumann. C'est un langage jouet, pour une machine jouet, dont les mots mémoire feraient 16 bits (2 octets).

En AMIL, les 4 premiers bits servent à coder l'instruction. Il peut donc y avoir 16 instructions différentes, et le reste des permettront d'adresser des registres ou des cases mémoire. On supposera ici les registres encodés sur 4 bits, et la mémoire adressable sur 8 bits.

Plan

- Architecture des ordinateurs
- Niveaux de langages
- Mini assembleur AMIL

Architecture de *von Neumann*



Mini assembleur AMIL

`stop`	Arrête l'exécution du programme
`noop`	N'effectue aucune opération
`saut i`	Met le compteur de programme à la valeur i (saut à l'adresse i)
`sautpos ri j`	Si la valeur contenue dans le registre i est positive ou nulle, met le compteur de programme à la valeur j
`valeur x ri`	Initialise le registre i avec la valeur x
`lecture i rj`	Charge, dans le registre j, le contenu de la mémoire d'adresse i
`ecriture ri j`	Écrit le contenu du registre i dans la mémoire d'adresse j
`inverse ri`	Inverse le signe dans le registre i

Mini assembleur AMIL

`add ri rj`	Ajoute la valeur du registre i à celle du registre j (la somme obtenue est placée dans le registre j)
`soustr ri rj`	Soustrait la valeur du registre i à celle du registre j (la différence obtenue est placée dans le registre j)
`mult ri rj`	Multiplie par la valeur du registre i celle du registre j (le produit obtenu est placé dans le registre j)
`div ri rj`	Divise par la valeur du registre i celle du registre j (le quotient obtenu, arrondi à la valeur entière inférieure, est placé dans le registre j)

Mini assembleur AMIL

Instructions plus avancées :

`et ri rj` Effectue le ET bit à bit de la valeur du registre i et de celle du registre j. Le résultat est placé dans le registre j

`lecture *ri rj` Charge, dans le registre j, le contenu de la mémoire dont l'adresse est la valeur du registre i

`ecriture ri *rj` Écrit le contenu du registre i dans la mémoire dont l'adresse est la valeur du registre j

MEMOIRE

1. lecture 10 r0
2. lecture 11 r2
3. soustr r2 r0
4. sautpos r0 8
5. lecture 10 r2
6. add r2 r0
7. saut 4
8. ecriture r0 12
9. stop
10. 14
11. 5
12. ?

Trace du programme

RI	Cycle	CP	r0	r2	10	11	12
(état initial)	0	1	?	?	14	5	?

MEMOIRE

1. lecture 10 r0
2. lecture 11 r2
3. soustr r2 r0
4. sautpos r0 8
5. lecture 10 r2
6. add r2 r0
7. saut 4
8. ecriture r0 12
9. stop
10. 14
11. 5
12. ?

Trace du programme

RI	Cycle	CP	r0	r2	10	11	12
(état initial)	0	1	?	?	14	5	?
lecture 10 r0	1	2	14				

MEMOIRE

1. lecture 10 r0
2. lecture 11 r2
3. soustr r2 r0
4. sautpos r0 8
5. lecture 10 r2
6. add r2 r0
7. saut 4
8. ecriture r0 12
9. stop
10. 14
11. 5
12. ?

Trace du programme

RI	Cycle	CP	r0	r2	10	11	12
(état initial)	0	1	?	?	14	5	?
lecture 10 r0	1	2	14				
lecture 11 r2	2	3		5			

MEMOIRE

1. lecture 10 r0
2. lecture 11 r2
3. soustr r2 r0
4. sautpos r0 8
5. lecture 10 r2
6. add r2 r0
7. saut 4
8. ecriture r0 12
9. stop
10. 14
11. 5
12. ?

Trace du programme

RI	Cycle	CP	r0	r2	10	11	12
(état initial)	0	1	?	?	14	5	?
lecture 10 r0	1	2	14				
lecture 11 r2	2	3		5			
soustr r2 r0	3	4	9				

MEMOIRE

1. lecture 10 r0
2. lecture 11 r2
3. soustr r2 r0
4. sautpos r0 8
5. lecture 10 r2
6. add r2 r0
7. saut 4
8. ecriture r0 12
9. stop
10. 14
11. 5
12. ?

Trace du programme

RI	Cycle	CP	r0	r2	10	11	12
(état initial)	0	1	?	?	14	5	?
lecture 10 r0	1	2	14				
lecture 11 r2	2	3		5			
soustr r2 r0	3	4	9				
sautpos r0 8	4	8					

MEMOIRE

1. lecture 10 r0
2. lecture 11 r2
3. soustr r2 r0
4. sautpos r0 8
5. lecture 10 r2
6. add r2 r0
7. saut 4
8. ecriture r0 12
9. stop
10. 14
11. 5
12. ?

Trace du programme

RI	Cycle	CP	r0	r2	10	11	12
(état initial)	0	1	?	?	14	5	?
lecture 10 r0	1	2	14				
lecture 11 r2	2	3		5			
soustr r2 r0	3	4	9				
sautpos r0 8	4	8					
ecriture r0 12	5	9					9

MEMOIRE

1. lecture 10 r0
2. lecture 11 r2
3. soustr r2 r0
4. sautpos r0 8
5. lecture 10 r2
6. add r2 r0
7. saut 4
8. ecriture r0 12
9. stop
10. 14
11. 5
12. ?

Trace du programme

RI	Cycle	CP	r0	r2	10	11	12
(état initial)	0	1	?	?	14	5	?
lecture 10 r0	1	2	14				
lecture 11 r2	2	3		5			
soustr r2 r0	3	4	9				
sautpos r0 8	4	8					
ecriture r0 12	5	9					9
stop	6	10					

MEMOIRE

1. lecture 10 r0
2. lecture 11 r2
3. soustr r2 r0
4. sautpos r0 8
5. lecture 10 r2
6. add r2 r0
7. saut 4
8. ecriture r0 12
9. stop
10. 14
11. 5
12. ?

Trace du programme

RI	Cycle	CP	r0	r2	10	11	12
(état initial)	0	1	?	?	14	5	?
lecture 10 r0	1	2	14				
lecture 11 r2	2	3		5			
soustr r2 r0	3	4	9				
sautpos r0 8	4	8					
ecriture r0 12	5	9					9
stop	6	10	9	5	14	5	9

Présentation de Pierre Boudes

Plan

Architecture des ordinateurs

Niveaux de langages

Mini assembleur AMIL

Merci pour votre attention.

Des questions ?