DIU – Enseigner l'informatique au lycée
Architecture des ordinateurs
Le modèle de von Neumann

D. Béchet

Denis.Bechet@univ-nantes.fr
Université de Nantes

6 juillet 2020

Comment fonctionnent les ordinateurs ?

Programme : le modèle de von Neumann et les circuits logiques

- Le modèle de Von Neumann
- Les jeux d'instructions des langages machines
- Le transistor et les portes logiques
- Les circuits séquentiels et les automates

Connaissances préalables :

- Calcul booléen
- Circuits combinatoires
- Codage des nombres et des caractères

Durée :

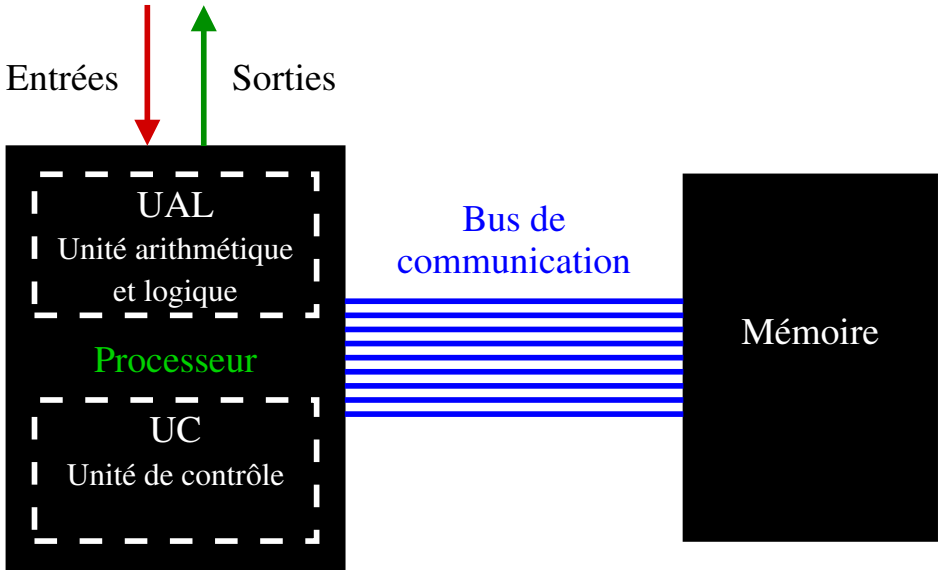
- 1H30 de présentation et 1H30 de TP cette semaine (von Neumann)
- 1H30 de présentation et 1H30 de TP à l'automne (circuits)

Ressources

De très nombreuses présentations (et cours) sur Internet

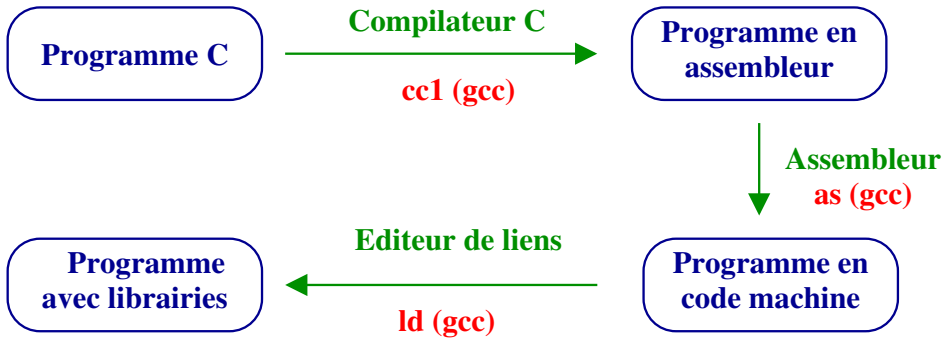
- Spécialité NSI - 1re, livre de Balabonski, Conchon, Filliâtre et Nguyen, editions Ellipses
- Spécialité NSI - Tle, livre de Bonnefoy et Petit, editions Ellipses
- Informatique et Sciences du Numérique - spécialité ISN en terminale S (version Python), livre de Gilles Dowek (en téléchargement libre) `https://wiki.inria.fr/wikis/sciencinfolycee/images/a/a7/Informatique_et_Sciences_du_Num%C3%A9rique_-_Sp%C3%A9cialit%C3%A9_ISN_en_Terminale_S._version_Python.pdf`
- WikiBooks, "Fonctionnement d'un ordinateur" `https://fr.wikibooks.org/wiki/Fonctionnement_d'un_ordinateur`
- Simulation d'un ordinateur : `http://www.peterhigginson.co.uk/LMC/` et `http://www.peterhigginson.co.uk/RISC/`

Question 8 : les 4 parties du modèle de von Neumann



- L'**unité de contrôle** lit les instructions du programme depuis la **mémoire** à travers un bus de communication
- Il commande l'**unité arithmétique et logique** qui effectue les calculs
- Le processeur (UC et UAL) possède aussi un bus pour les **entrées/sorties**

Question 9 : création d'un programme à partir d'instructions d'un processeur



- Les programmes exécutés par un processeur sont des suites de nombres placées en mémoire représentant des **instructions** exprimées en **langage machine**
- Un programme en langage machine est obtenue à partir d'un programme en **langage d'assemblage** par un **assembleur** comme as
- L'assembleur est en général appelé par un **compilateur** comme gcc

Question 9 : création d'un programme à partir d'instructions d'un processeur

Programme C

```
#include <stdio.h>

int a, b, c;

int main() {
    scanf("%d%d", &a, &b);
    c = a + b * 3;
    printf("Résultat : %d\n", c);
}
```

Langage d'assemblage

```
    ...
main:
    ...
    movl    $b, %edx
    movl    $a, %esi
    movl    $.LC0, %edi
    movl    $0, %eax
    call    scanf
    movl    b(%rip), %edx
    movl    %edx, %eax
    addl    %eax, %eax
    addl    %eax, %edx
    movl    a(%rip), %eax
    addl    %edx, %eax
    movl    %eax, c(%rip)
    movl    c(%rip), %eax
    movl    %eax, %esi
    movl    $.LC1, %edi
    movl    $0, %eax
    call    printf
    ...
```

Listing du code machine

```
    ...
                    main:
    ...
0004 BA00000000     movl    $b, %edx
0009 BE00000000     movl    $a, %esi
000e BF00000000     movl    $.LC0, %edi
0013 B800000000     movl    $0, %eax
0018 E800000000     call    scanf
001d 8B1500000000   movl    b(%rip), %edx
0023 89D0           movl    %edx, %eax
0025 01C0           addl    %eax, %eax
0027 01C2           addl    %eax, %edx
0029 8B0500000000   movl    a(%rip), %eax
002f 01D0           addl    %edx, %eax
0031 890500000000   movl    %eax, c(%rip)
0037 8B0500000000   movl    c(%rip), %eax
003d 89C6           movl    %eax, %esi
003f BF00000000     movl    $.LC1, %edi
0044 B800000000     movl    $0, %eax
0049 E800000000     call    printf
    ...
```

Question 10 : l'adresse de la dernière case d'un tableau de 100 entiers de 32 bits

Si `tab` est un tableau de 100 entiers, la dernière case est désignée par `tab[99]`. Son adresse en assembleur est **tab+99*4** ou **tab+396** car la taille d'un entier sur 32 bits est de 4 octets

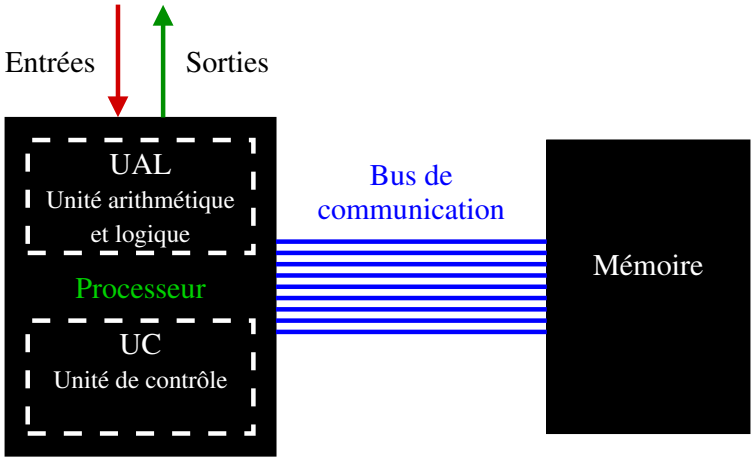
Programme C

```
#include <stdio.h>

int tab[100];

int main() {
    printf("%d %x\n",
           sizeof(int),
           &(tab[99]));
}
```

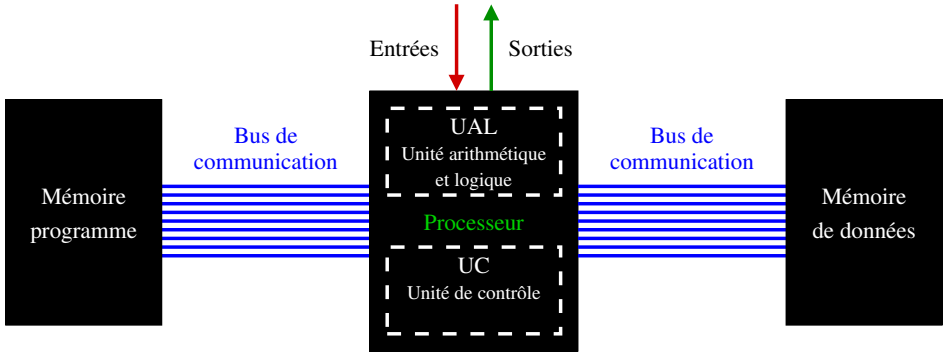
Programme en assembleur

```
        .file   "tab.c"
        .comm   tab,400,32
        .section        .rodata
.LC0:
        .string "%d %x\n"
        .text
        .globl  main
        .type   main, @function
main:
    ...
        movl    $tab+396, %edx
        movl    $4, %esi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
```

Modèle de von Neumann vs modèle de type Harvard

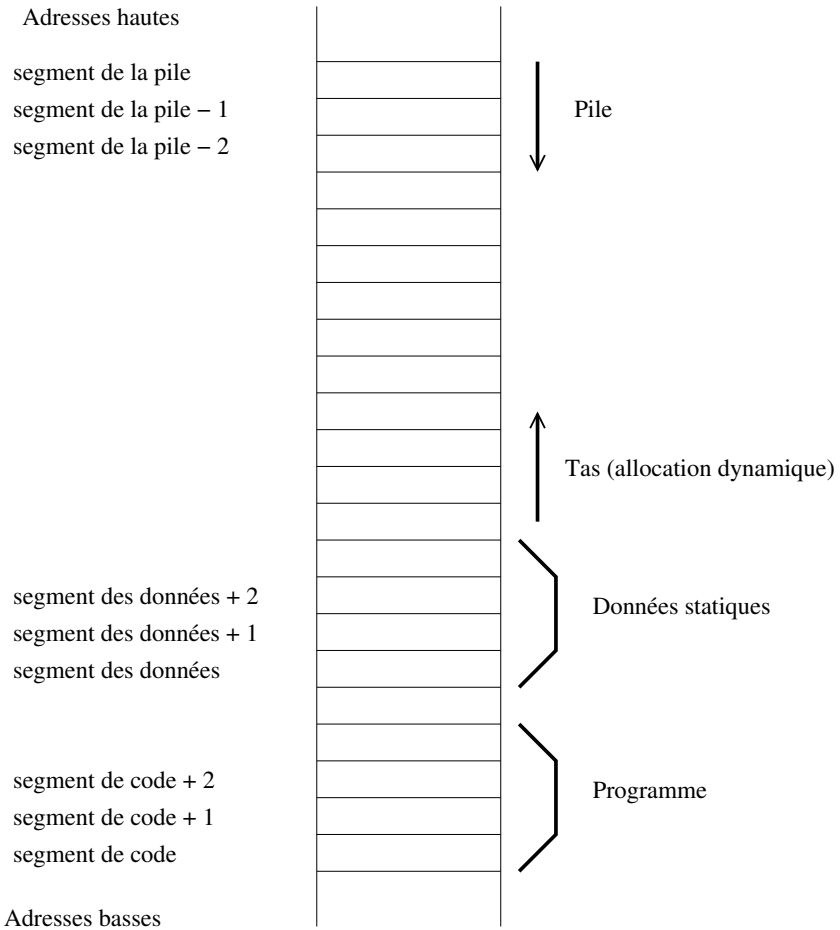


Les données et les instructions des programmes sont stockées au même endroit (dans la mémoire) dans le modèle de **von Neumann**



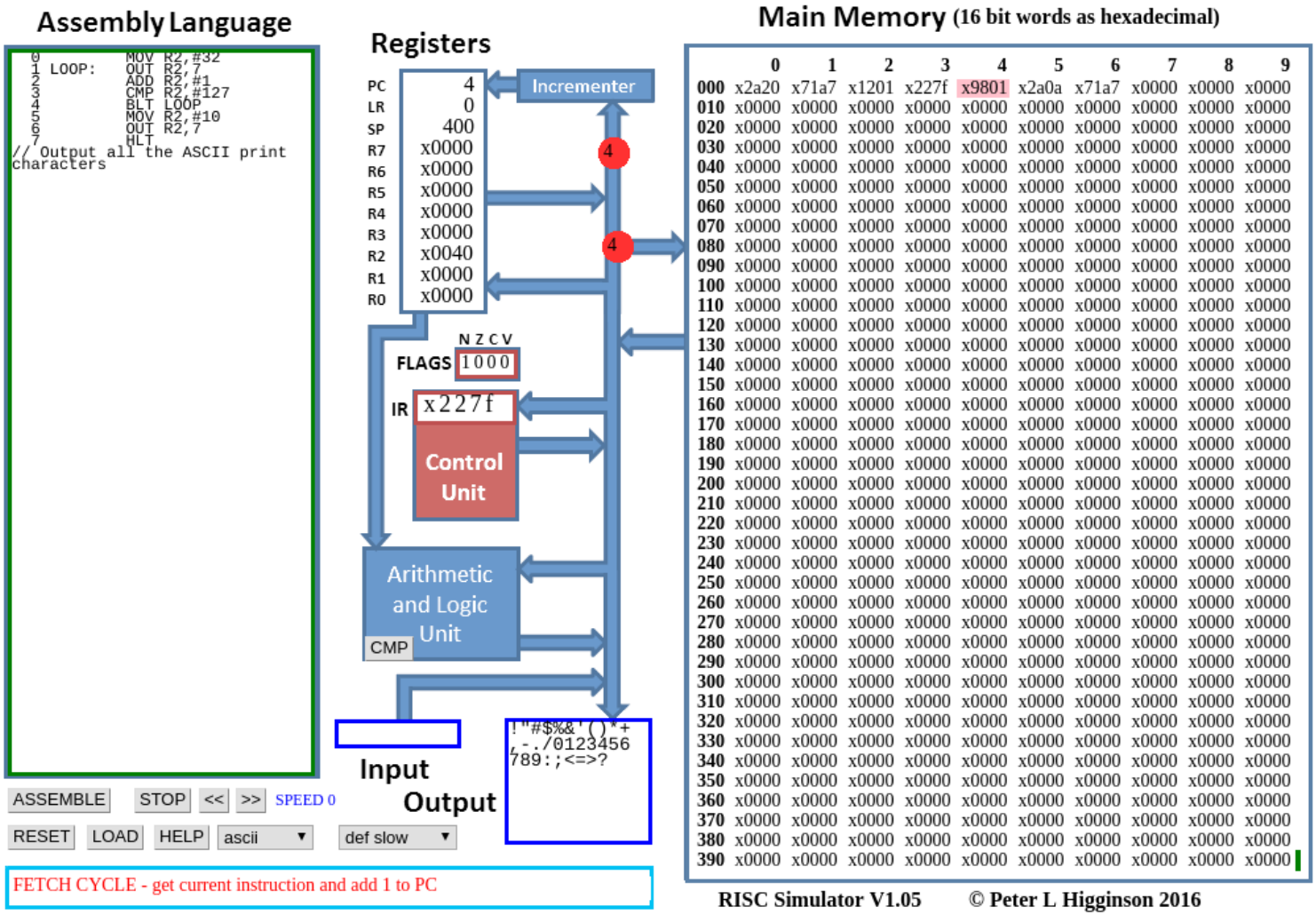
Architecture de type **Harvard** : 2 mémoires et 2 bus distincts pour les instructions et les données

Organisation de la mémoire



- Une case contient un **mot mémoire** constitué de 8 à 64 bits (ou plus)
- Une **variable** (ou une cellule d'une tableau) correspond à un ou plusieurs mots
- Une **instruction** (langage machine) correspond à un ou plusieurs mots
- Le processeur contient aussi des cellules mémoires appelées **registres** (largeur variable)

Un ordinateur (virtuel)



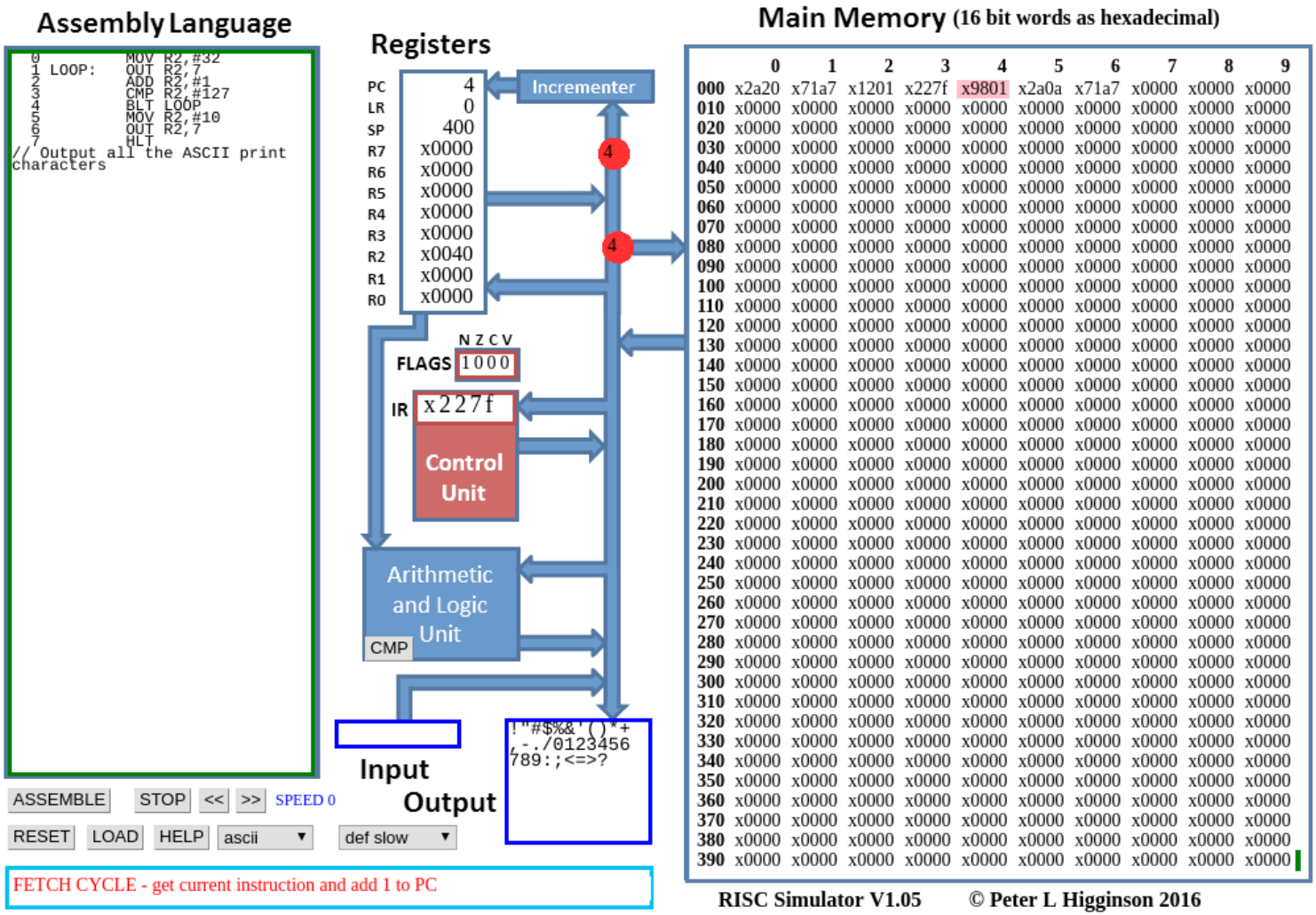
Instructions et cycle d'exécution du processur

Le registre **PC** *Program Counter* (appelé aussi **IP** pour *Instruction Pointer*) contient l'adresse mémoire de la prochaine instruction à exécuter

1. Récupération du code de l'instruction en utilisant le registre PC
2. Décodage de l'instruction et incrémentation (de 1) du PC
3. Exécution de l'instruction (plus ou moins complexe)
4. Retour à l'étape 1 (pour la prochaine instruction)

La liste des instructions possibles est spécifique à chaque processeur
Leur codage (langage machine) est souvent complexe

Simulation d'un ordinateur



Voir http://www.peterhigginson.co.uk/RISC/