# Architecture des Ordinateurs
# TD 7 - CPU

Halim Djerroud (hdd@ai.univ-paris8.fr)

## Exercice 1 :

1. On considère le programme suivant, écrit en pseudo-code (X, Y, P et Q sont des variables) :

   ```
   V = 4
   X = P + Q
   Si X == V
   Alors Y = V
   Sinon Y = V + V
   ```

   On considère le processeur à accumulateur vu en cours, dont le schéma est redonné ci-dessous. On note @X, @Y, @P, @Q, @V les adresses mémoire auxquelles sont rangées respectivement les variables X, Y, P, Q et V du programme en pseudo-code. Par exemple : @X=20, @Y=21, @P=22, @Q=23, @V=24. Vous pouvez utiliser aussi la variable tmp stockée à @tmp=25.

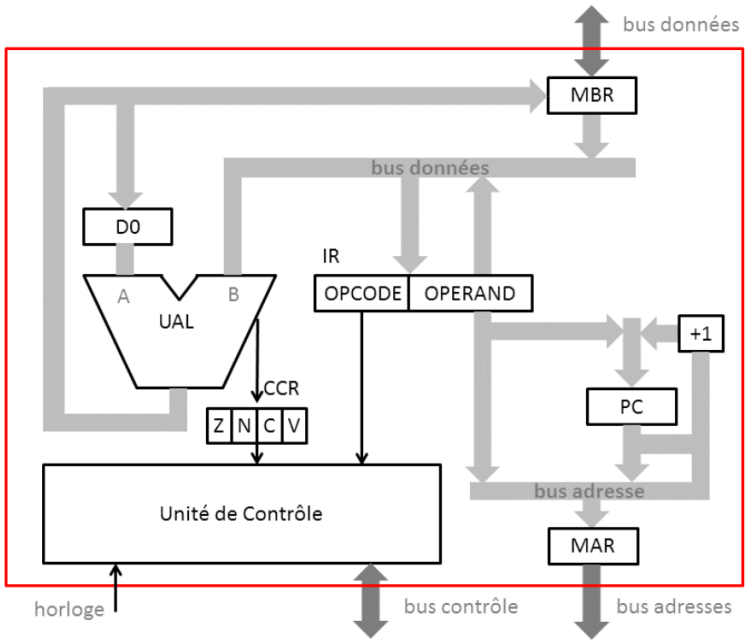
   (a) A l'aide des instructions du processeur Johnny (voir annexe), traduisez la première instruction du programme ci-dessus

   (b) Traduisez maintenant l'ensemble de ce programme. Discutez les différentes solutions possibles, comparez-les en termes de nombre d'instructions totales et exécutées.

## Exercice 2 :

On considère le programme en assembleur écrit à l'exercice 1.

1. Rappelez les trois phases d'exécution d'une instruction, ainsi que leur rôle.
2. Détaillez le chemin de données parcouru lors de la phase fetch. Est-il nécessaire pour toutes les instructions de la machine ? Est-il le même pour toutes les instructions de la machine ?
3. Détaillez le chemin de données pour l'ensemble des instructions processeur correspondant aux lignes suivantes du programme en pseudo-code : `X = P + Q ;`




## Exercice 3 : Simulateur Johnny

1. En utilisant le simulateur Johnny, donnez la suite d'instructions (voir le jeu d'instruction en annexe ou se référer à la documentation) qui permet de connaître la taille d'un tableau à partir de la plage d'indices valides (indice_max - indice_min + 1). Les variables sont stockées aux adresses suivantes :

   ```
   020 résultat
   021 indice_min
   022 indice_max
   ```

2. Écrire un autre programme qui calcule la taille d'une chaîne de caractère. L'adresse de début de la chaîne est stockée à l'adresse 20.
   Rappel : Une chaîne de caractère se termine par un 0. Stockez le résultat à l'adresse @21. Afin de ne pas perdre le pointeur sur le début de la chaîne, vous pour le dupliqueronse à @22.
3. Donner la suite d'instructions qui permet de faire la somme des 10 premiers entiers.

   ```
   020 valeur
   021 résultat
   ```

4. En vous basant sur un algorithme itératif de la suite de Fibonacci, donner la suite d'instructions qui permet de calculer $F_6$ .

5. Écrire un programme qui permet de trouver la valeur maximale parmi toutes les valeurs en mémoire situées entre l'adresse 100 et l'adresse 200.

## Exercice 4 :

1. Soit le programme en C suivant. Traduisez ce programme en assembleur. On suppose que la variable **s** est stockée à l'adresse 100 et que la variable i est stockée à l'adresse 200.

```
int s = 0 ;
for (int i = 0 ; i < 100 ; i++) {
s += i;
}
```

# 1 Annexe : jeu d'instruction Johnny

— **TAKE** : La valeur de l'emplacement (donnée par l'adresse absolue) est transférée vers l'accumulateur.
— **SAVE** : La valeur de l'accumulateur est transférée vers l'emplacement donné par l'adresse absolue.
— **ADD** : La valeur de l'emplacement (donnée par l'adresse absolue) est ajoutée (+) à la valeur dans l'accumulateur.
— **SUB** : La valeur de l'emplacement (donnée par l'adresse absolue) est soustraite (-) de la valeur dans l'accumulateur.
— **INC** : La valeur de l'emplacement (donnée par l'adresse absolue) est incrémentée (+1).
— **DEC** : La valeur de l'emplacement (donnée par l'adresse absolue) est décrémentée (-1).
— **NULL** : La valeur de l'emplacement (donnée par l'adresse absolue) est mise à zéro.
— **TST** : Si et seulement si l'emplacement (donné par l'adresse absolue) a une valeur nulle, alors l'instruction suivante est ignorée.
— **JMP** : Le programme se poursuit à l'emplacement indiqué (goto).
— **HLT** : Le simulateur affiche un message indiquant que le programme est terminé.