**ALGORITHMIQUE PROGRAMMATION et LANGAGES**
**APL1**

**Ce module est encadré par Emilien Micard, Erwan Kerrien, Stéphane Dieudonné et Abdellatif Bourjij.**

**Volume horaire : 60h**

**APL1 :** 10h cours soit 4 séances (2h) et 2 colles (1h).
22h TD soit 11 séances de 2h.
28h TP soit 7 séances de 4h ou 14 séances de 2h.

**Plan du cours :**

I_ Introduction et Définitions importantes
II_ Notion de mémorisation et de type
III_ Structures algorithmiques fondamentales
IV_ Structures de données génériques
V_ Notion de sous-programme et fonction
VI_ Complexité, tests unitaires.

**Plan des TD :** voir Annexe 1

**Plan des TP :** voir Annexe 2

**Evaluation :** 2 colles comptant pour 3/5 + note de TD comptant pour 1/5+ note de TP comptant pour 1/5.
La note de TD est donnée par l'enseignant en fonction de l'assiduité, la participation et les colles « rapides ».
La note de TP est une moyenne des comptes rendus de TP.

Le langage utilisé en TP est le Python

**Bibliographie**


| **Titre** | **Auteur** | **Editeur** |
|---|---|---|
| Introduction à l'Agorithimique | T. Cormen & co | Dunod |
| Initiation à l'Algorithmique | J. Courtin & co | Dunod |
| Initiation à la Programmation | C. Delannoy | Eyrolles |
| www.visualgo.net | | |

**Déroulement des enseignements :**

| Semaine | Cours | Enseignant | TD | Enseignant | TP | Enseignant | colle |
|---|---|---|---|---|---|---|---|
| 1 | 1-2 | AB | | | | | |
| 2 | 3-4 | AB | 1-2 | AB | | | |
| 3 | | | 3-4 | EK | 1 | SD-EM | |
| 4 | | | 5-6 | AB | 2 | SD-AB | |
| 5 | | | 7-8 | EK | 3 | SD-EM | 1 |
| 6 | | | 9-10 | AB | 4 | SD-AB | |
| 7 | | | 11 | EK | 5 | SD-EM | |
| 8 | | | | | 6 | SD-AB | |
| 9 | | | | | 7 | SD-AB | |
| 10 | | | | | | | 1 |

EK : Erwan Kerrien
SD : Stéphane Dieudonné
EM : Emilien Micard
AB : Abdellatif Bourjij

**ANNEXE 1**
**ALGORITHMIQUE ET PROGRAMMATION**
***TRAVAUX DIRIGES APL1***

**TD N°1**
**Objectifs :** mémorisation, variables, expressions booléennes, affectations et premiers algorithmes.

**1/** Rappel : une expression est dite "booléenne" si sa valeur est dans l'ensemble {Vrai, Faux}. Les variables a, b et c désignent des entiers. Parmi les expressions suivantes, quelles sont les expressions booléennes ?

- 7=2
- 5>3
- (a+b)*3
- a+(b-c)*2
- (a < 2) et (a > 2)
- a*b
- a+b
- (a < -1) ou (a > 1)
- non(a>1)
- non a

**2/** Ecrire la négation des expressions suivantes :

- (a > 10)
- (b ≤ 2) et (b ≥ -2)
- (a < -5) ou (a > 5)

**3/** Réécrivez plus simplement les expressions suivantes :

- (a ≤ 0) = Faux
- non((b != 0) = Faux)
- non ((a ≤ 0) et (a = 0))

**4/** Ecrire les expressions suivantes en n'utilisant que les opérateurs >, =, non, et , ou :

- a != 0
- b ≥ 2
- c ≤ 1
- a < 0

**5/** Ecrire une expression booléenne pour chacune des propositions suivantes (les variables x, y ,z, a et b étant des entiers) :

- les valeurs de x, y et z sont égales
- La valeur de x est comprise entre celle de a et celle de b (avec a < b)
- la valeur de x est différente de celle de a et celle de y est différente de celle de b
- la valeur de x est strictement négative et celle de y est strictement positive
- la valeur de x est 1 ou 2 ou 3 ou 4

**6/** Traduire sous forme d'une phrase en français la plus simple possible, chacune des expressions suivantes :

- non ((a!=2) et (a<3))
- non ((x>y) ou ((x mod 2 = 0) et (x-y > 10)))
- (x mod 3 = 0) et (x mod 4 = 0)
- non (( x<3) ou (x >5))

**7/** Simuler l'exécution des instructions suivantes en donnant l'état des variables après chaque affectation (a, b et c sont des entiers, cvrai et cfaux sont des booléens, s1 et s2 sont des chaines de caractères)).

a ← 0
a ← a+1
a ← a+1
a ← 1+a
b ← 1
a ← a+b
b ← a+b

a ← a+b
b ← b-a
a ← a+b
b ← a+b
a ← b+a
b ← a+b
a ← 1
b ← a+1
c ← (a +3*b) mod 2
cvrai ← Vrai ou Faux
cfaux ← Vrai et (b >2)
cfaux ← non cfaux et non cvrai
s1 ← "Bon"
s2 ← "Jour"
s1 ← s1+s2
s2 ← s1+" et Bonsoir"

**8/** On constitue 3 tas de 16 allumettes. On fait passer la moitié du premier tas dans le second puis la moitié du second dans le troisième et enfin la moitié du troisième dans le premier. Ecrire un algorithme qui réalise ces opérations et affiche le contenu de chaque tas à la fin des transferts.

**9/** Soient deux variables entières de type entier a et b, ecrire un algorithme qui échange leurs valeurs sans utiliser de variable supplémentaire.

**TD N°2**

**Objectifs :** Tests simples, tests doubles et boucles.

**1/** Voici un algorithme inacceptable, dans lequel les variables a, b et c sont booléennes et la variable x est entière. Réécrire cet algorithme sous une forme correcte puis simplifiez-le (5 lignes).

**Si** a **Alors**
**Si** non (b) **Alors Si** non (a) **Alors** x ← 0
**Sinon Si** c **Alors** x ← 1
**Sinon**
x ← 0 **Finsi Finsi**
**Sinon Si** non (b) **Alors** x ← 1
**Sinon Si** non (c) **Alors** x ← 0
**Sinon** x ← 1
**Finsi Finsi Finsi Sinon Si** c **Alors** x ← 1
**Sinon** x ← 0 **Finsi Finsi**

**2/** Saisir 3 nombres calculer: la somme, le produit et la moyenne.

**3/** Saisir 2 nombre afficher le signe de leur produit (positif ou négatif).

**4/** Trois nombres positifs a,b et c peuvent représenter les trois cotés d'un triangle dans la mesure où chacun d'entre eux est inférieur à la somme des deux autres:

a < b+c b < a+c c < a+b

Saisir a, b et c afficher si oui ou non ils forment un triangle.

**5/** Écrire un algorithme qui demande de saisir deux réels et un caractère, puis affiche la valeur de l'expression ainsi définie.

Exemple:
Donnez un premier réel : 1.5
Donnez un deuxième réel : 4.0
Donnez un caractère parmi +, -, *, / : *
Le résultat de 1.5 * 4.0 est 6.0

**6/** Écrire un algorithme qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie :

- « Poussin » de 6 à 7 ans
- « Pupille » de 8 à 9 ans
- « Minime » de 10 à 11 ans
- « Cadet » après 12 ans

**7/** Écrire un algorithme qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître un message : « Plus petit ! », et inversement, « Plus grand ! » si le nombre est inférieur à 10.

**TD N°3**
**Objectifs :** Algorithmes.

**1/** Écrire un algorithme qui :
1. demande à l'utilisateur de donner un chiffre strictement compris entre 50 et 100,
2. si le nombre saisi n'est pas entre 50 et 100 affiche un message d'erreur et quitte,
3. sinon ajoute ce nombre à 62 et affiche le résultat,
4. supprime le premier chiffre (le plus à gauche) de ce résultat et l'ajoute au nombre restant (exemple : 123 - 1 = 23, 23 + 1 = 24) puis affiche le résultat;
5. enlève ce dernier nombre obtenu au premier nombre donné par l'utilisateur et affiche le résultat ;
6. vérifie que le résultat final vaut bien 37.

**2/** Saisir n, saisir n nombres, trouver le plus petit et le plus grand.

**3/** Saisir N, calculez les sommes:

$S_1 = 1 + ½ + 1/3 + ¼ + \ldots + 1/N$

$S_2 = 1 - ½ + 1/3 - ¼ \ \ldots \ +-1/N$

**4/** Trouver un nombre inférieur à 100 qui satisfait les conditions suivantes :
- le reste de sa division par 2 est 1,
- le reste de sa division par 3 est 2,
- le reste de sa division par 4 est 3,
- le reste de sa division par 5 est 4.

**5/** Résoudre une équation du 2° degré dans R.

**6/** Soit y, la fonction de t:

$y = 2*t^3 - t^2 - 37*t + 36$

Calculez y pour chaque valeur de t comprise entre -5 et 5 par pas de 0.25 et déterminez le maximun et le minimun des valeurs de y ainsi calculées.

**TD N°4**
**Objectifs :** Algorithmes.

**1/** Ecrire un algorithme qui affiche la table de multiplication pour un nombre donné.

**2/** Trouver toutes les paires d'entiers positifs (m, n), telles que:

$m^2+2*n^2<100$

**3/** Affichez toutes les manières possibles d'obtenir un euro avec des pièces de 2 centimes, 5 centimes et 10 centimes. Dire combien de possibilités ont été trouvées. Vous afficherez les résultats sous la forme suivante:

1E = x*2c + y*5c + z*10c

**4/** Afficher un rectangle formé d'étoiles « * ». La largeur et la longueur sont données.

**5/** Affichez un triangle isocèle formé d'étoiles. La hauteur du triangle sera fournie en donnée. La dernière ligne du triangle sera affichée sur le bord gauche de l'écran.

**6/** Afficher les figures suivantes :

| | |
|---|---|
| x | 0000000 |
| xx | 111111 |
| xox | 22222 |
| xoox | 3333 |
| xooox | 444 |
| xoooox | 55 |
| xxxxxxx | 6 |

**7/** Calculer les racines carrées de nombre donnés jusqu'à la saisie du nombre 0. Les valeurs négatives seront refusées.

Calcul de la racine carrée par l'algorithme de Heron :

$a_{n+1} = 1/2 ( a_n + A/a_n )$ (initialisation avec $a_n=A$)

**TD N°5**
**Objectifs :** Algorithmes et tableaux.

**1/** Déterminez la $n^{ième}$ valeur $u_n$ (n étant une donnée) de la suite de Fibonacci définie comme suit:

$u_1 = 1$

$u_2 = 1$

$u_n = u_{n-1} + u_{n-2}$ pour $n>2$

**2/** Saisir un entier au clavier et afficher sa représentation en binaire.

**3/** Écrivez un algorithme qui a près avoir demandé un numéro de jour, de mois et d'année à l'utilisateur, renvoie s'il s'agit ou non d'une date valide.
Rappel : le mois de février compte 28 jours, sauf si l'année est bissextile, auquel cas il en compte 29. L'année est bissextile si elle est divisible par quatre. Toutefois, les années divisibles par 100 ne sont pas bissextiles, mais les années divisibles par 400 le sont.

**4/** Saisir un tableau de nombres afficher sa taille, le nombre d'occurrences d'un nombre donné. Insérer un nombre donné à une position donnée. Supprimer toutes les occurrences d'un nombre donné.

**5/** Écrire un algorithme qui teste si la chaîne passée en paramètre est un palindrome ou pas.

**6/** Écrivez un algorithme qui demande une phrase à l'utilisateur et affiche le nombre de mots de cette phrase. On suppose que les mots ne sont séparés que par un espace et un seul.

**7/** Écrire un algorithme qui demande une phrase à l'utilisateur et qui affiche le nombre de voyelles contenues dans cette phrase.

**8/** Un des plus anciens systèmes de cryptographie (aisément déchiffrable) consiste à décaler les lettres d'un message pour le rendre illisible. Ainsi, les A deviennent des B, les B des C, etc. Ecrivez un algorithme qui demande une phrase à l'utilisateur et qui la code selon ce principe.

**9/** Une technique ultérieure de cryptographie consista à opérer non avec un décalage systématique, mais par une substitution aléatoire. Pour cela, on utilise un alphabet-clé, dans lequel les lettres se succèdent de manière désordonnée, par exemple : XKGUJPVREAYBNDOFSQZCWMLITH
C'est cette clé qui va servir ensuite à coder le message. Selon notre exemple, les A deviendront des X, les B des K, les C des G, etc.
Ecrire un algorithme qui effectue ce cryptage (l'alphabet-clé sera stocké dans une constante).

**TD N°6**
**Objectifs :** Algorithmes et tableaux.

**1/** Écrire un algorithme qui teste si un tableau est trié.
**2/** Soit un tableau T1[N] donné. Écrire un algorithme permettant de renverser l'ordre de ses éléments, sans utiliser d'autre tableau.
**3/** Un tableau T1[m+n] est la concaténation de deux segments T1[1..m] et T1[m+1..m+n]. Mettre le deuxième segment au début du tableau, sans utiliser d'autres tableaux.
**4/** Trouver le nombre d'éléments différents dans un tableau donné et mémorisez les dans un second tableau.
**5/** Le crible d'Ératosthène permet de déterminer les nombres premiers inférieurs à une certaine valeur N donnée. On place dans un tableau unidimensionnel T les nombres entiers compris entre 1 et N. L'algorithme consiste, pour chaque élément T[i], à rechercher parmi tous les suivants (indice i+1 à N) ceux qui sont des multiples et les éliminer (par exemple les remplacer par des 0 ou -1). Lorsque tout le tableau a subi ce traitement, seul les nombres premiers n'ont pas été éliminés du tableau. Ecrire l'algorithme de ce crible.

**TD N°7**
**Objectifs :** étude des fonctions.
Présentation des fonctions sous python.
Étude de fonctions simples, transformation des algorithmes précédents en fonctions ou procédures. Étude des fonctions récursives (exemples: factorielle, exposant, pgcd, syracuse, reverse, tri-récursif etc).

**TD N°8**
**Objectifs :** Les tests unitaires.
Présentation des tests unitaires sous python, application aux fonctions vues précédemment.

**TD N°9 & 10**
**Objectifs :** Étude et comparaison de différentes méthodes de tri.

- tri à bulle,
- tri par insertion,
- tri par sélection,
- tri radix
- tri compteur

(voir visualgo.net, écrire les différentes étapes des tris sur des exemples)
Recherche dichotomique.

**TD N°11**
**Objectifs :** Préparation aux projets Mips, Voyageur de commerce, Codage, Démineur, Simulateur robot tondeuse, jeu au choix.

**ANNEXE 2**
**ALGORITHMIQUE ET PROGRAMMATION**
***TRAVAUX PRATIQUES AP1***

**TP N°1-3**

**Objectifs:** Initiation à la programmation et étude des différents environnements de développement (notamment sous Windows et Linux).

Saisie et test sur un jeu de données significatif des différents algorithmes réalisés en TD.

**TP N°4**

**Objectifs**: étude des fonctions et des fonctions récursives (vues en TD)

Ecrire la fonction permettant le calcul du factorielle (n!=n*(n-1)*(n-2)* ...*1)

Ecrire la fonction puissance permettant le calcul de $x^n$

Ecrire la fonction reverse permettant de retourner une chaîne de caractère inverse de celle reçu en paramètre (BONJOUR retournera RUOJNOB)

Ecrire la fonction pgcd(a,b) permettant de calculer le plus grand commun diviseur de deux nombres

Ecrire la fonction fibo(n) permettant de calculer la suite de fibonacci (rappel : $u_1=1$, $u_2=1$, $u_n=u_{n-1}+u_{n-2}$)

**TP N°5**

**Objectifs**: étude des tableaux, complexité d'un algorithme.

Traduisez les algorithmes de tris vus en TD puis testez-les et comparez leur rapidité sur de grands tableaux aléatoires (tri à bulle, tri par insertion, tri par sélection, tri radix, tri compteur).

**TP N°6**

**Objectifs**: Appréhender les fonctions de manipulation de fichier, utilisation des tests unitaires chaque fonction doit être accompagnée d'une fonction de test.

Écrire une fonction permettant d'ouvrir un fichier texte donné par l'utilisateur puis de le coder en remplaçant chaque caractère par un caractère obtenu en réalisant une translation des codes ACSII. Une translation de x s'obtient en ajoutant x au code ASCII de chaque caractère par exemple en prenant x=1, 'A' devient 'B' et X devient 'Y'.

Utilisez cette fonction pour réaliser les fonctionnalités suivantes ::

coder un fichier source et générer un fichier destination nommé source.cod,

décoder un fichier codé et générer le fichier source,

à partir d'un fichier texte générez un fichier destination ne contenant que des majuscules,

et à partir d'un fichier texte générez un fichier destination ne contenant que des minuscules.

(Utiliser la documentation et l'aide en ligne, vous aurez besoin des fonctions open(), read(), write(), ord () et chr())

**TP N°7**

**Objectifs** : développer un mini logiciel en autonomie.

Sujets au choix :

**Simulateur Robot Tondeuse**

Utilisez le module nommé **turtle** afin de réaliser des dessins simples. On disposera d'un curseur que l'on peut faire avancer, reculer, tourner, sauter à une nouvelle position etc.

La commande from turtle import * permet de charger toutes les fonctions du module souhaité.

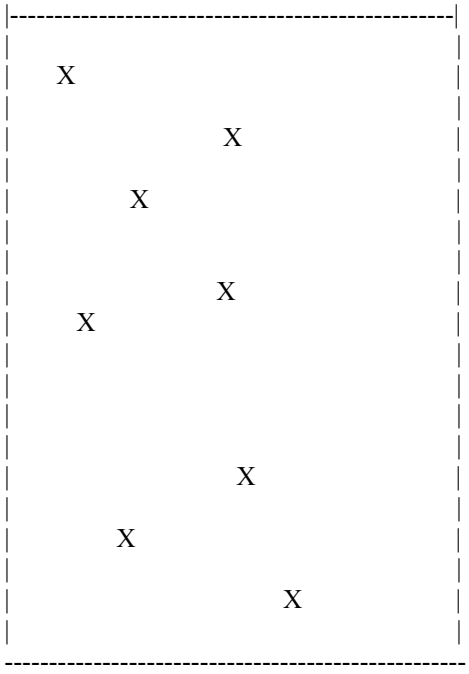
A l'aide de la documentation disponible http://docs.python.org/3/library/turtle.html, dessinez une surface à tondre et posez des obstacles dedans (leurs coordonnées doivent être répertoriées). Puis faites se déplacer le curseur aléatoirement dans cet surface en s'arrêtant face aux obstacles et en restant dans la surface à tondre. Quand la tondeuse rencontre un obstacle elle s'arrête, recule et dévie d'un angle aléatoire avant de reprendre sa course. Ajoutez la possibilité de choisir l'algorithme de tonte : aléatoire, en spirale etc

Prévoir une fonction de calcul du temps de tonte pour chaque algorithme.

**Le voyageur de commerce**

1/ Générer un tableau de 8 Coordonnées aléatoires. Une coordonnée est un tableau de 2 valeurs x (compris entre 0 et 49) et y  (compris entre 0 et 19).

2/ Générer un affichage basique de la forme suivante, en y insérant les point générés

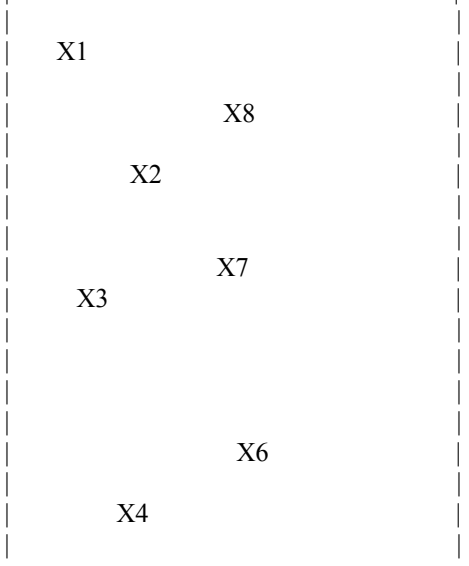


3/ Calculer toutes les distances entre tous les points

4/ A l'aide de plusieurs boucles imbriquées, construisez le tableau de tous les chemins possibles
12345678
12345687
12345768
12345786
...
87654321

5/ Calculer tous les possibilités de voyage et trouvez le voyage le plus court.
Nous considéreront que le point de départ sera le point de coordonnées [0,0]
Affichez le résultat

6/ Régénérer l'affichage en notant l'ordre de passage a coté des points



| X5 |
| |
---

6/ Sachant que le véhicule possède un réservoir de 10L et que la consommation moyenne est de 12L aux 100 km, définissez chacune des villes dans lesquelles le plein d'essence devra être effectué.

**ou**

**Le jeu du Démineur**

Le démineur est un jeu dont le but est de localiser des mines cachées dans un champ virtuel avec pour seule indication le nombre de mines dans les zones adjacentes.
Écrire une fonction générateur() dont le but sera d'initialiser la grille (matrice carrée de taille au choix 30x30 ou 40x40 ). Puis cacher dans cette grille 30 ou 40 bombes de manière aléatoire.
La fonction jouer() doit permettre à l'utilisateur de saisir la taille de la grille de jeu désirée, fixer le nombre de bombes et enfin jouer jusqu'au déminage total du champ ou explosion d'une bombe. Le joueur pourra saisir les positions de son choix en indiquant les numéros de ligne et de colonne. Si cette cellule contient une bombe le jeu est fini et vous afficherez le nombre de coups joués et le temps passé sinon vous devrez indiquer le nombre de bombes dans toutes les cellules autour du choix de l'utilisateur.
La fonction Affichage() générera un écran basique de la forme suivante :

---
| |
| |
| |
| |
...
---

Une fonction gestionScore() permettra aux différents joueurs de mémoriser leurs scores et d'afficher les scores par ordre de mérite et les dates de réalisations.

**ou**

**Simulateur pour le contrôle d'un processeur ---MIPS**

Il s'agit de simuler les opérations réalisées par le système de contrôle d'un processeur ---MIPS lors de l'exécution des lignes d'un programme source assembleur.
Le modèle (Machine à états finis) de la logique de contrôle à simuler est fourni en annexe.

Les commandes (sorties) à envoyer aux différents circuits (mémoires, registres, multiplexeurs, UAL,...) seront simplement intégrées à différents messages pour affichage.

En annexe un exemple de code et un exemple de message

1_Lecture du fichier source .asm
Il présente une instruction par ligne.
Une fonction Lecture_Mnemo retourne dans un tableau Code_a_Executer les lignes lues.

2_Extraction et Segmentation de l'instruction
Une fonction Seg_Mnemo extrait la x ième instruction de Code_a_Executer et isole chaque champ dans un tableau Champs_Code.

3_Dictionnaire des mnémoniques : Dico_Mnemo
Il fournit les éléments d'interprétation du mnémonique à exécuter.
Il donne le code opération (Op) et associe à chaque état les champs du mnémonique qui seront utilisés.
Soit une structure de données Dico_Mnemo=['Op',dict({'etat' :list(champ)})]

(1) lw $2,100($1) dont l'interprétation est
Op : lw
Etat 1 : lecture de $1 et $2
Etat 2 : addition de 100 et de $1
Etat 3 : lecture mémoire

Etat 4 : Ecriture de $2

En rouge les éléments que doit fournir le dictionnaire pour cet exemple

4_Table d'état du modèle : Etat_Suivant
Elle représente la structure du modèle.
Elle renseigne l'état suivant à partir du code Op et de l'état courant et ce pour l'ensemble du modèle.

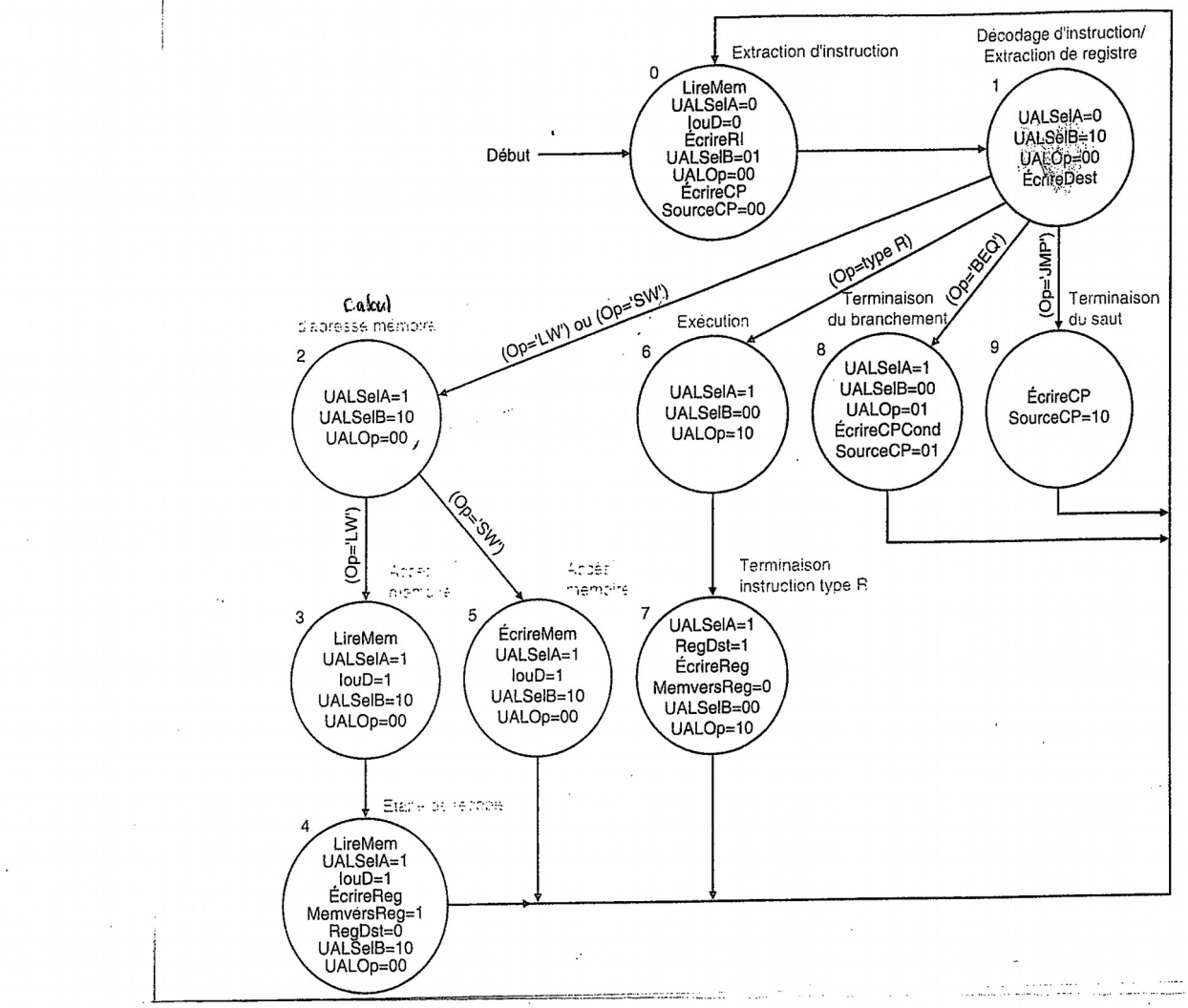
Soit : Etat_Suivant={'Etat_courant','Op':'Etat_suivant'}

5_Messages de sortie
Une fonction affiche le message qui correspond à l'état courant et aux valeurs du dictionnaire pour cet état.

6_Exécution de la simulation
L'ensemble du code de code_à_exécuter doit être simulé.
Le point 2 correspond à l'état 0 du modèle.
Pour chaque ligne de code_à_exécuter, on souhaite dérouler pas à pas les états du modèle.
-----Distinguer le cas de l'état 0 des autres ----

Le premier objectif est de réaliser un test sur l'instruction lw $2,100($1). (1)
Vous pouvez commencer au point 3 avec Champ_code pré-rempli, et limiter la définition du dictionnaire et de la table d'état à (1).Ensuite, vous faites les points 1 et 2 pour (1).Enfin vous développez l'ensemble de la solution.

**Annexes**



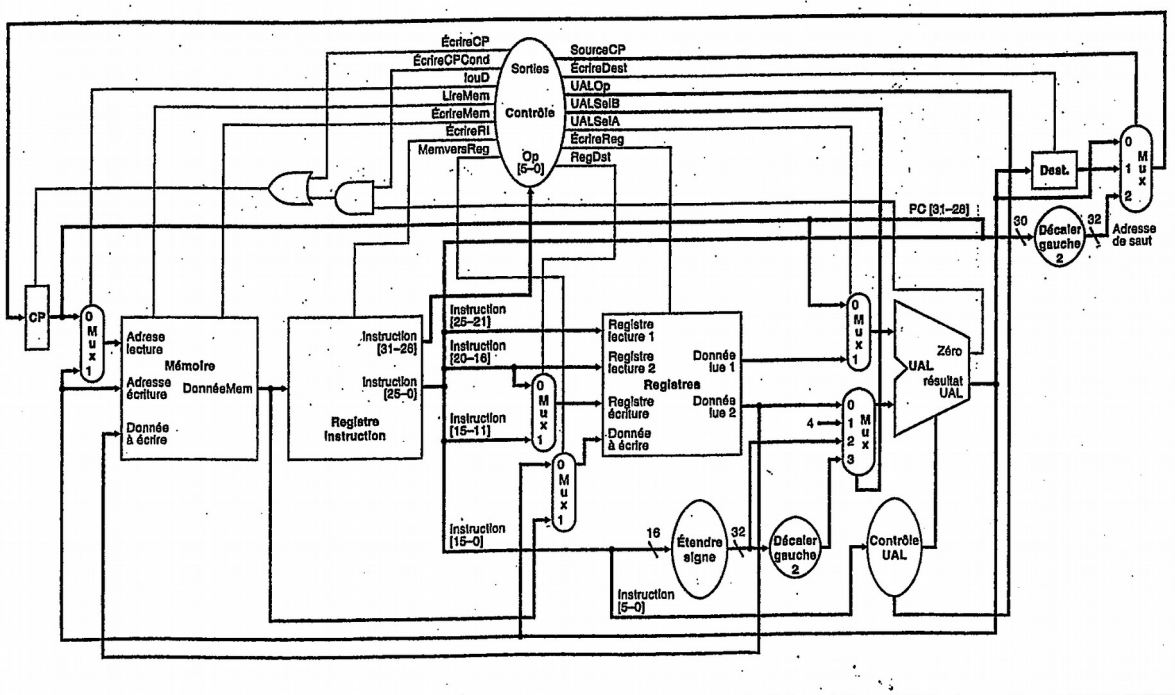




**FIGURE 5.39 Voici le chemin de données complet pour la mise en œuvre multicycle**

Exemple de code

```
Lw $8,100($19)
Bne $8,$20,Exit
Add $19,$19,4
j 0
sw $2,120($3)
```

Messages pour lw $2,100($3)

Etat 0 : Extraction d'instruction
Etat 1 : lecture de $1 et $2
Etat 2 : addition de 100 et de $1
Etat 3 : lecture mémoire
Etat 4 : Ecriture de $2