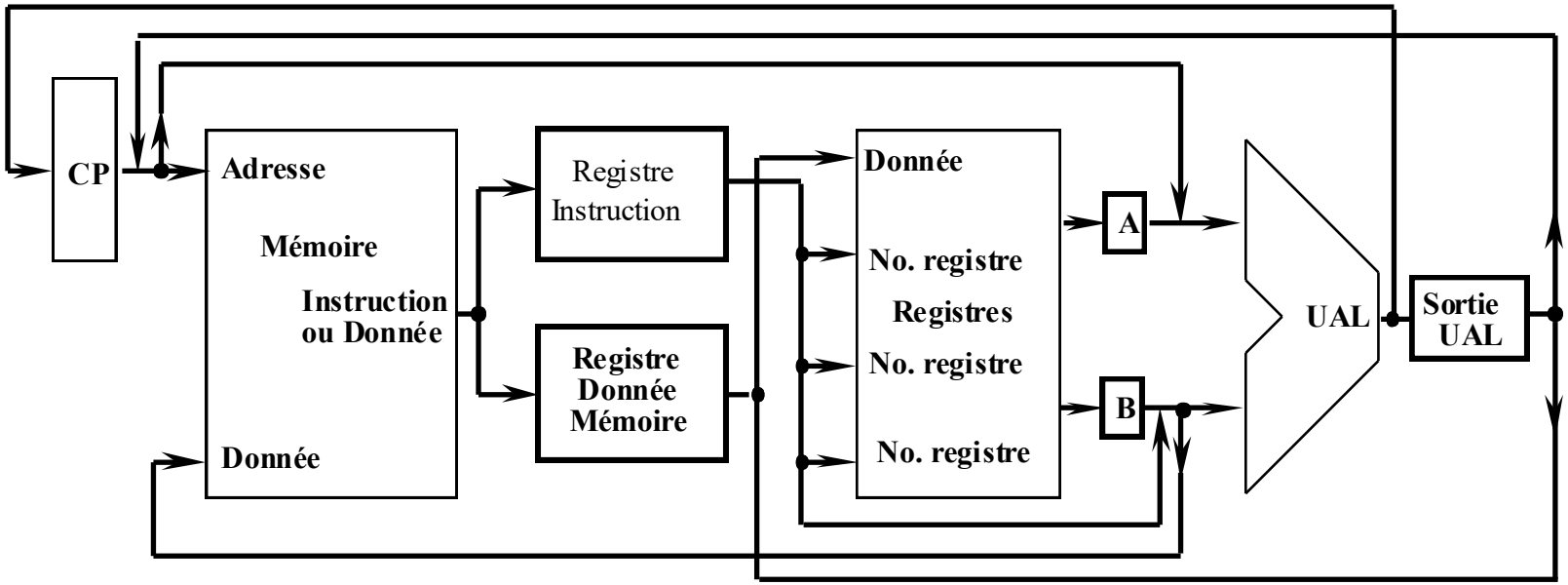
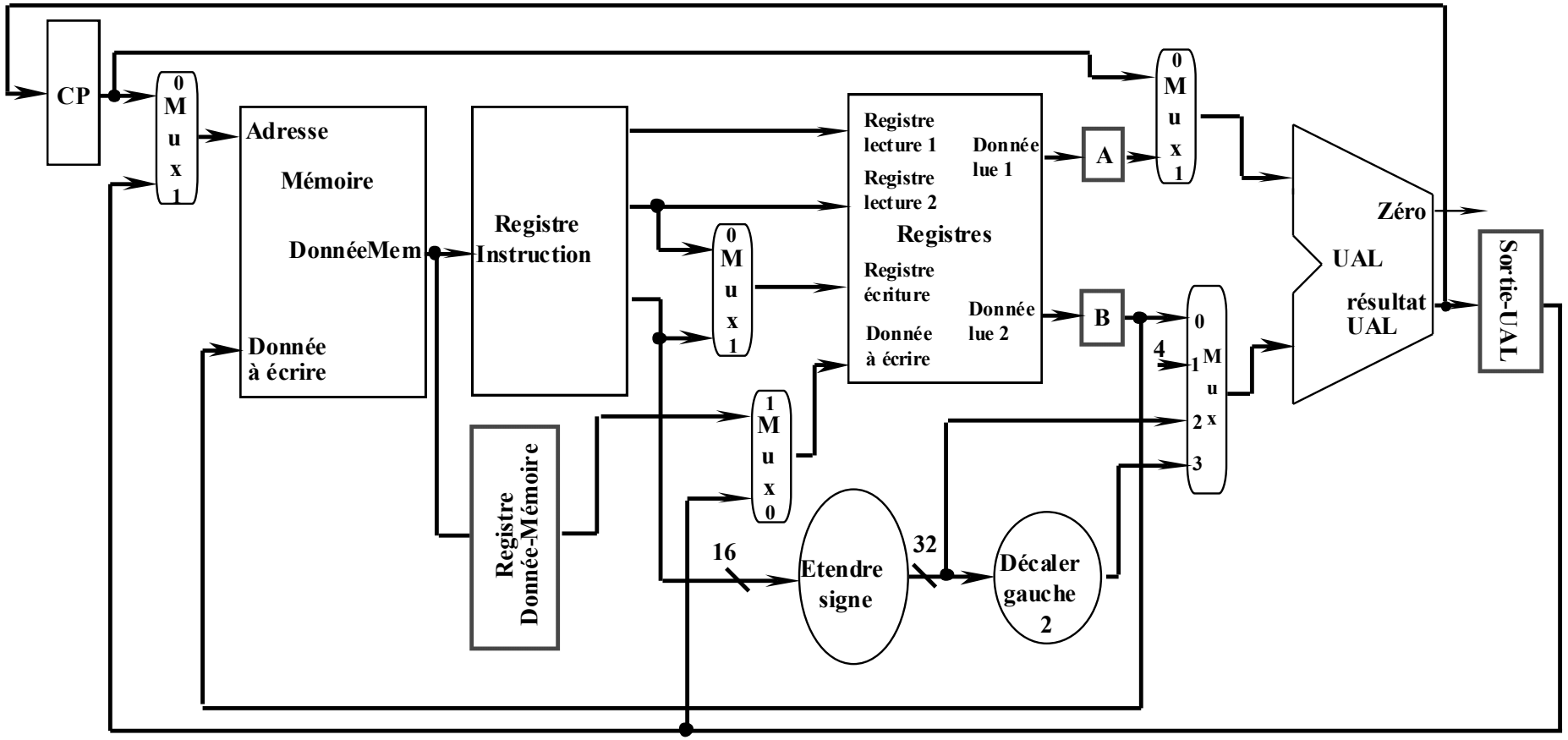
ETUDE ET CONCEPTION D'UN PROCESSEUR : CHEMIN DE DONNEES A PLUSIEURS CYCLES D'HORLOGE ET UNITE CENTRALE DE CONTROLE

VUE GLOBALE DU CHEMIN DE DONNEES MULTICYCLE

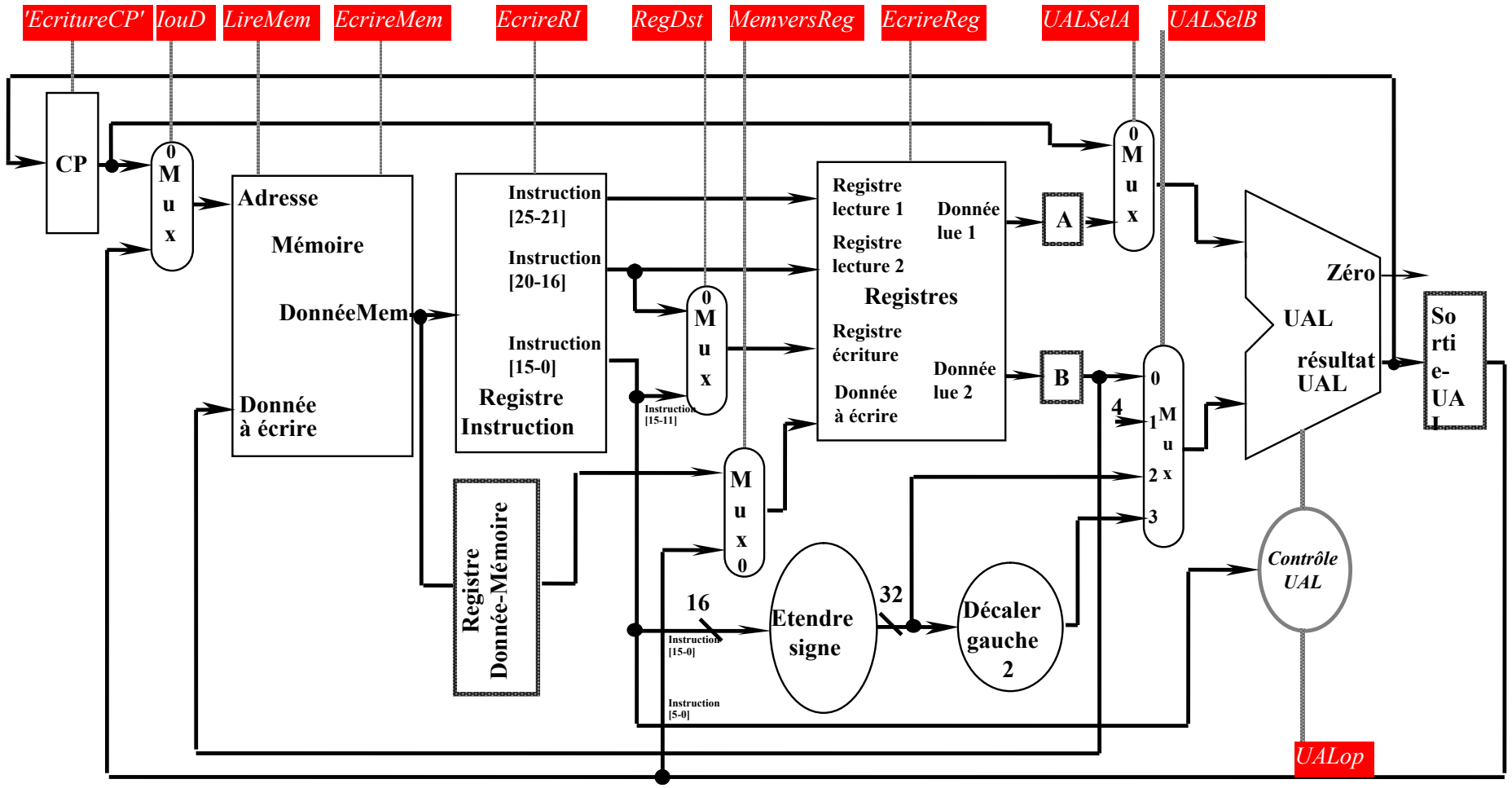


- ✓ **Une seule unité mémoire est utilisée pour les instructions et les données.**
- ✓ **Un registre est utilisé pour sauvegarder l'instruction après sa lecture. Ce registre instruction (RI) est nécessaire du fait que la mémoire peut être réutilisée plus tard dans l'exécution de l'instruction.**
- ✓ **Une UAL seulement, plutôt qu'une UAL et deux additionneurs.**

DETAIL DU CHEMIN DE DONNEES MULTICYCLE



SIGNAUX DE CONTROLE DU CHEMIN DE DONNEES MULTICYCLE



DESCRIPTION DES SIGNAUX DE CONTROLE ET LEUR EFFET

Nom du signal	Effet lorsque le signal est inactif	Effet lorsque le signal est actif
LireMem	Aucun	Contenu mémoire placé sur la sortie «Donnée Mem»
EcrireMem	Aucun	Contenu mémoire remplacé par la valeur à l'entrée «Donnée à écrire»
UALSelA	1er opérande UAL = CP	1er opérande UAL = Reg[rs]
RegDst	N° de registre à écrire = champ **rt**.	N° de registre à écrire = champ **rd**.
EcrireReg	Aucun	Ecriture de la valeur à l'entrée du banc de registres
MemversReg	Valeur à écrire dans le banc provient de l'UAL.	Valeur à écrire dans le banc provient de la mémoire.
IouD	Adresse mémoire provient du CP	Adresse mémoire provient de l'UAL.
EcrireRI	Aucun	Mémorisation de l'instruction dans (RI).
EcritureCP	Aucun	Chargement du CP avec une adresse choisie parmi plusieurs possibles

Nom du signal	valeur	Effet
UALSelB	**00**	2ème opérande UAL = Reg[rt].
	01	2ème opérande UAL = 4.
	10	2ème opérande UAL = RI[15-0] avec extension sur 32 bits.
	11	2ème opérande UAL = (RI[15-0] avec extension sur 32 bits <<2)
UALOp	**00**	Addition.
	01	Soustraction.
	10	Opération UAL déterminée par le code de fonction.

DECOUPAGE DE L'EXECUTION D'UNE INSTRUCTION EN CYCLES D'HORLOGE (1/7)

1. Etape d'extraction d'instruction :

Prendre l'instruction en mémoire et incrémenter le compteur de programme.

RI = Mémoire [CP];
CP = CP + 4;

Opérations	*Signaux de contrôle activés*
Envoyer le CP à la mémoire comme adresse.	***IouD = 0*** (choisir CP comme source d'adresse)
Effectuer une lecture mémoire.	***LireMem = 1***
Placer l'instruction dans le (RI), où elle sera sauvegardée.	***EcrireRI = 1***
Incrémenter CP de 4.	***UALSelA = 0*** (choix de CP) ***UALSelB = 01*** (choix de 4) ***UALOp = 00*** (addition UAL)
* Stocker de nouveau dans le CP la nouvelle adresse d'instruction.	***Activer écriture CP**

*** Chemin et contrôle du CP seront ajoutés quand le contrôle complet pour CP (branchement et saut inclus).**

Remarque : **l'incrémentation du CP et l'accès à la mémoire peuvent avoir lieu en parallèle.**

DECOUPAGE DE L'EXECUTION D'UNE INSTRUCTION EN CYCLES D'HORLOGE (2/7)

2. Etape de décodage de l'instruction et extraction de registres :

Type instruction encore indéterminé => on ne peut mener que des actions qui sont :

- **soit communes à toutes les instructions (telle que l'extraction de l'instruction),**

- **soit sans danger, dans le cas où l'instruction ne serait pas celle imaginée.**

  - ✓ **Lire 2 registres repérés rs et rt et les mémoriser dans les registres temporaires A et B.**

  - ✓ **Calculer avec l'UAL l'adresse de destination de branchement et la sauvegarder dans le registre temporaire SortieUAL.**

DECOUPAGE DE L'EXECUTION D'UNE INSTRUCTION EN CYCLES D'HORLOGE (3/7)

2. Etape de décodage de l'instruction et d'extraction de registre : (suite)

A = Registre[RI[25-21]] ;
B = Registre[RI[20-16]] ;
SortieUAL = CP + (extension-signe(RI[15-0]) << 2) ;

Opérations	*Signaux de contrôle activés*
Lire 2 registres en utilisant les champs rs et rt et les mémoriser dans A et B.	Aucun positionnement de ligne de contrôle n'est nécessaire.
Calculer l'adresse de branchement et la sauvegarder dans SortieUAL.	***UALSelA = 0*** (choix de CP) ***UALSelB = 11*** (choix déplacement étendu et décalé) ***UALOp = 00*** (addition UAL)

Remarques :

- **Les accès aux registres et le calcul de l'adresse de destination de branchement ont lieu en parallèle.**
- **Ce cycle d'horloge achevé, l'action à mener peut dépendre du contenu de l'instruction.**

DECOUPAGE DE L'EXECUTION D'UNE INSTRUCTION EN CYCLES D'HORLOGE (4/7)

3. Exécution, calcul d'adresse mémoire, terminaison de branchement ou terminaison de saut :

Instruction arithmétique et logique (type R) :

SortieUAL = A op B;

Opérations	*Signaux de contrôle activés*
Effectuer sur A et B lus au cycle précédent l'opération définie par le code fonction.	***UALSelA = 1*** (choix A) ***UALSelB = 00*** (choix B) ***UALOp = 01*** (choix du code fonction pour définir l'opération UAL)

Référence mémoire :

SortieUAL = A + extension-signe (RI[15-0]);

Opérations	*Signaux de contrôle activés*
Effectuer par l'UAL (A + déplacement) pour former l'adresse effective de la mémoire.	***UALSelA = 1*** (choix A) ***UALSelB = 10*** (choix de la sortie de l'unité d'extension de signe) ***UALOp = 00*** (addition UAL).

DECOUPAGE DE L'EXECUTION D'UNE INSTRUCTION EN CYCLES D'HORLOGE (5/7)

3. Exécution, calcul d'adresse mémoire, terminaison de branchement ou terminaison de saut : (suite)

Branchement si égal :

Si (A == B) alors CP = SortieUAL;

Opérations	*Signaux de contrôle activés*
Effectuer par l'UAL le test d'égalité entre A et B lus au cycle précédent.	***UALSelA = 1*** (choix A), ***UALSelB = 00*** (choix B), ***UALOp = 10*** (force une soustraction UAL).

⇒ **La sortie *Zéro* de l'UAL détermine s'il faut ou non effectuer un branchement.**
⇒ **Activer le signal de mise à jour du CP si le signal *Zéro* est actif.**
⇒ **On le verra plus loin comment le contrôle du CP sera ajouté.**

Instruction de saut :

CP = CP[31-28] || (RI[25-0] << 2)

Opérations	*Signaux de contrôle activés*
CP est remplacé par l'adresse de saut	***SourceCP = 10*** (choix de l'adresse de saut), ***EcritureCP = 1***

DECOUPAGE DE L'EXECUTION D'UNE INSTRUCTION EN CYCLES D'HORLOGE (6/7)

4. Accès mémoire ou étape de terminaison d'une instruction de type R :

Durant cette étape :

- ✓ **soit accès mémoire effectué par un chargement ou un rangement**
- ✓ **soit écriture du résultat d'une opération arithmétique ou logique**

Référence mémoire : **Donnée-Mémoire = Mémoire [SortieUAL]; ou**
Mémoire [SortieUAL] = B;

Opérations	*Signaux de contrôle activés*
Envoyer l'adresse à la mémoire calculée à l'étape précédente.	***IouD = 1*** (choix SortieUAL plutôt que CP).
Si l'instruction est un chargement, une donnée revient de la mémoire	
Lecture de la mémoire, et mémorisation dans Donnée-Mémoire**.**	***LireMem = 1***
Si l'instruction un rangement, alors la donnée est écrite en mémoire	
L'opérande dans B, lu lors de l'étape survenue deux cycles plus tôt.	***EcrireMem = 1***

Instruction arithmétique ou logique (type R) : **Reg[RI[15-11]] = SortieUAL;**

Opérations	*Signaux de contrôle activés*
Placer SortieUAL à l'entrée du banc.	***MemversReg = 0*** (choix de SortieUAL).
Placer le champ rd (RI [15-11]) à l'entrée du banc.	***RegDst = 1*** (choix du champ rd comme adresse d'écriture).
Ecrire le contenu SortieUAL dans le banc**.**	***EcrireReg = 1***

DECOUPAGE DE L'EXECUTION D'UNE INSTRUCTION EN CYCLES D'HORLOGE (7/7)

5. Etape de terminaison de la lecture mémoire :

Ecriture du banc de registres :

Reg[RI[20-16]] = Donnée-Mémoire;

Opérations	*Signaux de contrôle activés*
Placer le contenu de Donnée-Mémoire à l'entrée donnée du banc .	***MemversReg = 1*** (choix de la donnée mémoire).
Placer le champ rt (RI[20-16]) à l'entrée adresse du banc.	***RegDst = 0*** (choix du champ rt comme adresse d'écriture).
Ecrire le contenu de Donnée-Mémoire dans le banc de registre.	***EcrireReg = 1***

RECAPITULATIF DU DECOUPAGE L'EXECUTION D'UNE INSTRUCTION ET DES ACTIONS A MENER EN FONCTION DE SON TYPE

Nom de l'étape	*Action pour les instructions de type R*	*Action pour les instructions de référence mémoire*	*Action pour les branchements*	*Action pour les sauts*
Extraction de l'instruction	RI = Mémoire[CP] CP = CP + 4			
Décodage de l'instruction et extraction de registre	A = Registre[RI[25-21]] B = Registre[RI[20-16]] SortieUAL = CP + (extension-signe (RI[15-0]) << 2)			
Exécution, calcul d'adresse, terminaison de branchement ou de saut	SortieUAL = A op B	SortieUAL = A + extension signe (RI[15-0])	Si (A == B) Alors CP = SortieUAL	CP= CP[31-28] ||(RI[25-0]<<2)
Accès mémoire ou terminaison d'une instruction de type R	Reg[RI[15-11]] = SortieUAL	Donnée-Mémoire = Mémoire[SortieUAL] ou Mémoire [SortieUAL] = B		
Ecriture de mise à jour		Reg[RI[20-16]] = Donnée-Mémoire		
Nombre de cycles (étapes)	4	5	3	3

MISE EN ŒUVRE DU CONTROLE DU CP

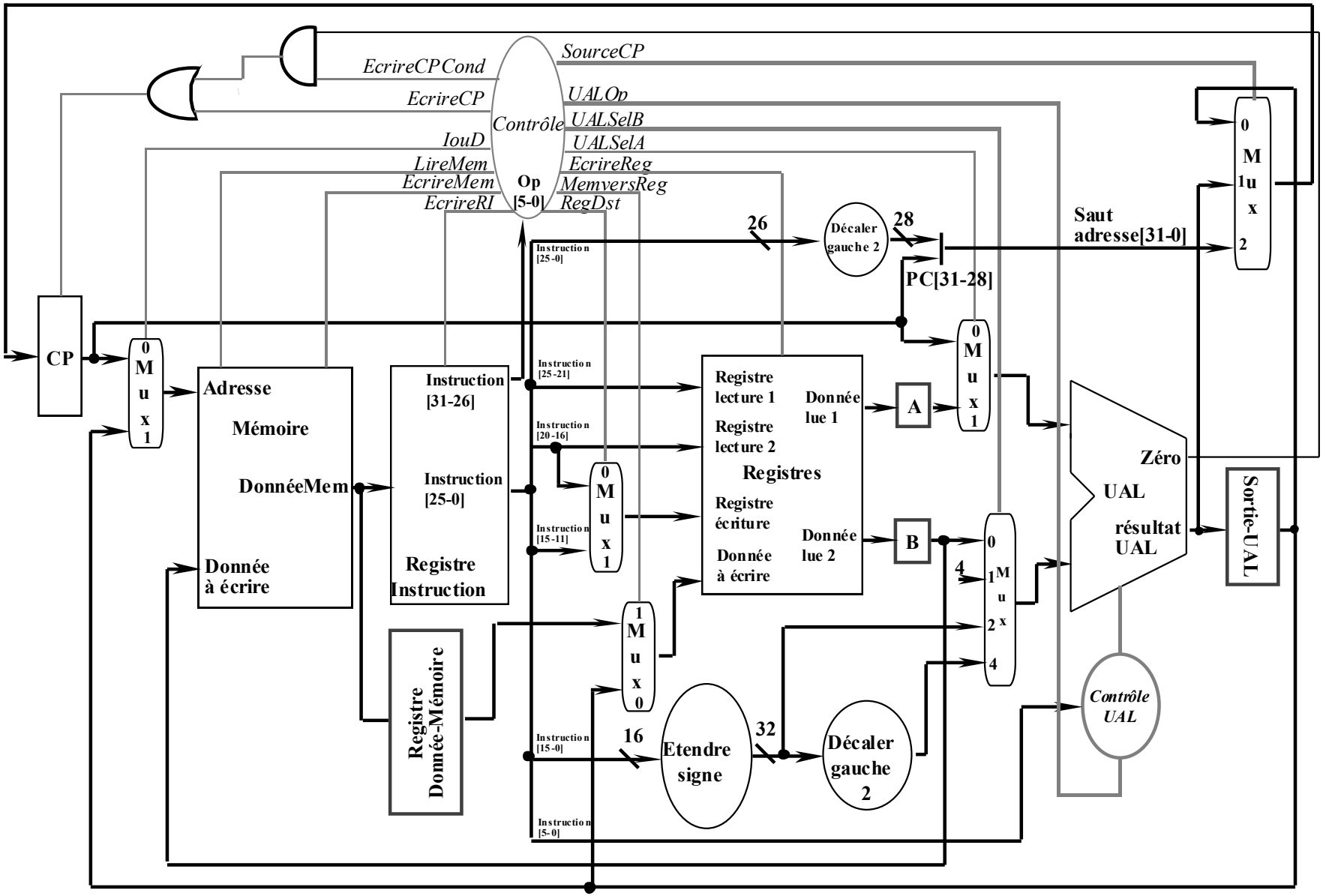
- **3 sources possibles pour le CP :**
  - ✓ sortie Résultat UAL (cas de l'instruction suivante)
  - ✓ registre SortieUAL (cas de branchement conditionnel effectué)
  - ✓ adresse de saut (cas de branchement inconditionnel)
- **Un multiplexeur 3 voies contrôlé par un signal 2 bits (*SourceCP*) permet de sélectionner l'une des 3 sources.**
- **Le CP est écrit de deux façons différentes :**
  - ✓ Conditionnelle (cas branchements)
  - ✓ Inconditionnelle (pour les autres cas)
- **Il faut donc 2 signaux d'écriture du CP :**
  - ✓ un signal *EcrireCPCond* conditionné avec le signal *Zero* de l'UAL par une porte ET
  - ✓ un signal *EcrireCP*
- **Le signal final d'écriture du CP** $EcritureCP = EcrireCPCond \cdot Zero + EcrireCP$

EFFET DES SIGANUX DE CONTROLE DETERMINANT L'ECRITURE DU CP

Nom du signal	Effet lorsque le signal est inactif	Effet lorsque le signal est actif
EcrireCP	Aucun	Ecriture CP, la source est contrôlée par SourceCP.
EcrireCPCond	Aucun	Ecriture CP si Zero = 1.

Nom du signal	valeur	Effet
SourceCP	00	Source = registre SortieUAL.
	01	Source = sortie résultat de l'UAL
	10	Source = Adresse du saut ((CP[31-28])||RI[25-0], <<2).

CHEMIN DE DONNEES ET DE CONTROLE COMPLET



DEFINITION DU CONTROLE

- Le contrôle doit à la fois positionner les signaux de l'étape courante et spécifier l'étape suivante.
- Deux techniques différentes permettent de définir le contrôle d'un chemin de données multicycle :
  - ✓ Les machines à états finis, représentées graphiquement sous forme d'automates à états finis.
  - ✓ La microprogrammation : le contrôle est représenté sous forme de programme.

Ces deux techniques permettent à un système de CAO de synthétiser la mise en œuvre en utilisant des portes logiques, des ROM ou des RLP.

IMPLEMENTAION DU CONTROLE PAR UNE MACHINE A ETATS FINIS (1/8)

Définition : Machine à Etats Finis (MEF) = ensemble d'états + indications sur la façon de changer d'états.

Indications : définies par une *fonction* **Etat-Suivant** qui fait correspondre à **l'état courant** un nouvel état en fonction des entrées.

Ensemble d'états : définis par une *fonction* **Sortie** qui fait correspondre à chaque état l'ensemble des signaux (sorties) actifs lorsque la machine est dans cet état.

Pour implémenter le contrôle comme une MEF :

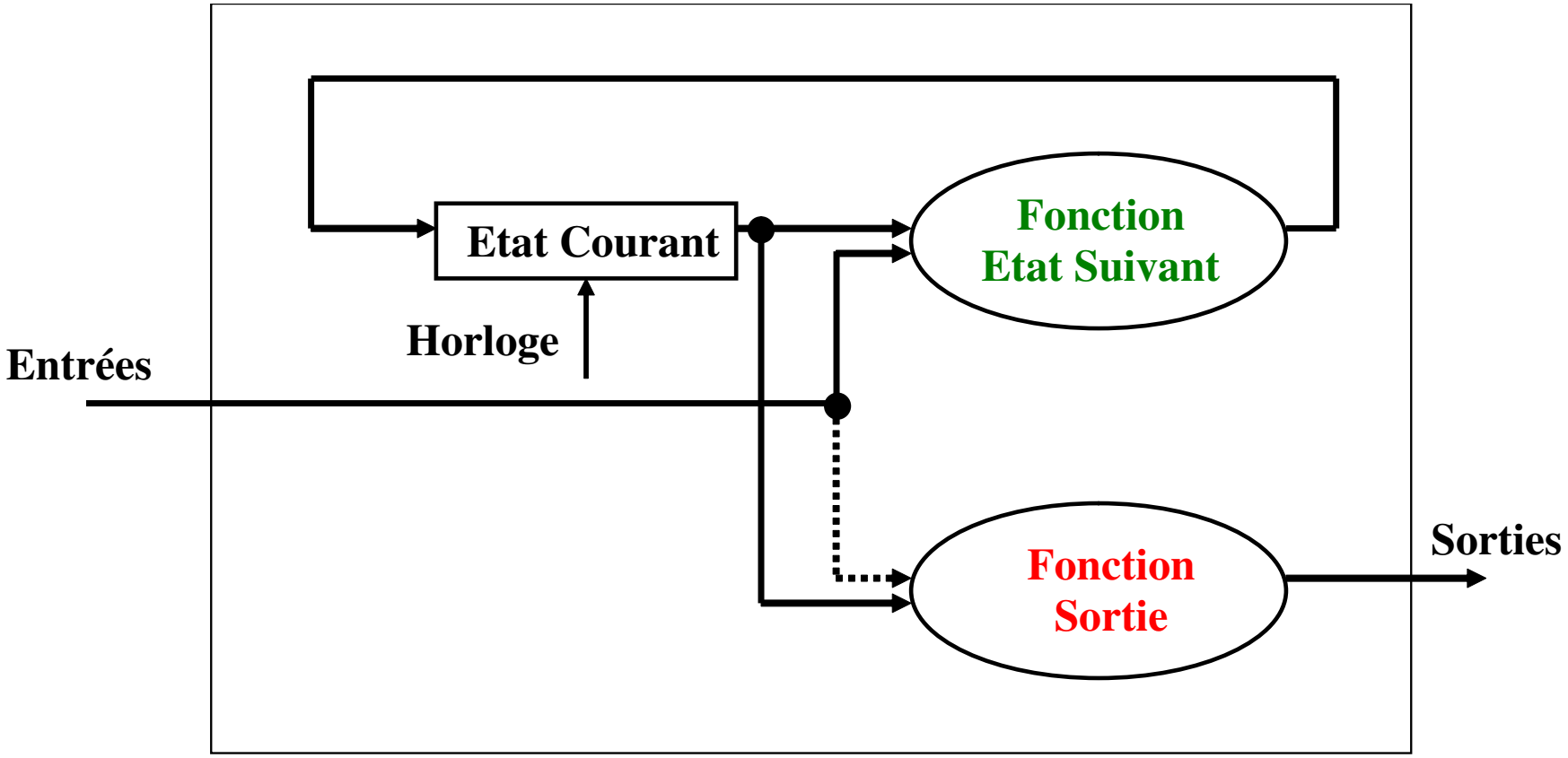
- On doit assigner un numéro à chacun des états.
- On doit fixer le nombre de bits nécessaires pour numéroter ces états.
- Le numéro de l'état courant doit être mémorisé dans un **registre d'état**.

L'état **i** est encodé en utilisant les bits d'états comme le nombre binaire **i**.

L'unité de contrôle a des **sorties** qui spécifient **l'état suivant** en plus des signaux de contrôles.

L'état suivant est mémorisé dans **le registre d'état** sur le front actif de l'horloge et devient le nouvel état au début du cycle d'horloge suivant (toujours sur le front actif de l'horloge).

IMPLEMENTAION DU CONTROLE PAR UNE MACHINE A ETATS FINIS (2/8)

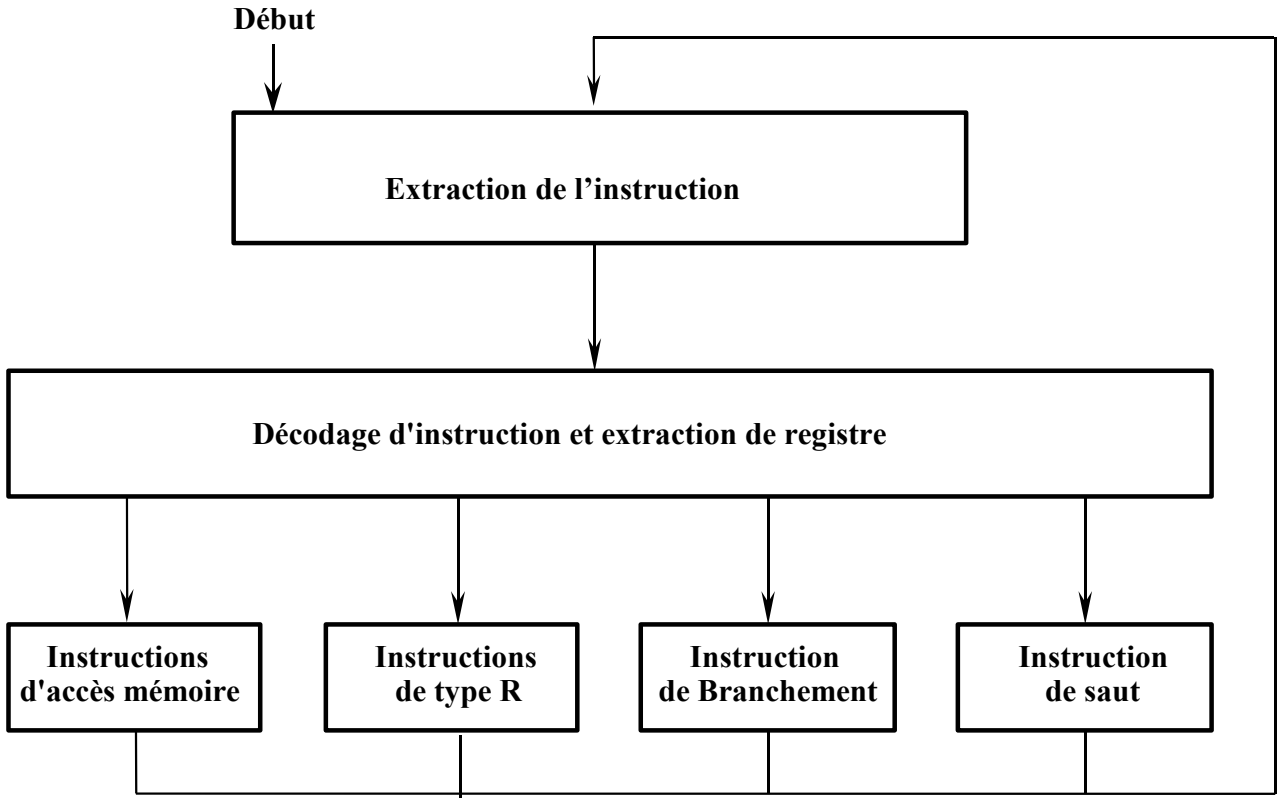


Les deux fonctions (fonction Sortie et fonction Etat Suivant) sont totalement combinatoires.

Souvent la fonction Sortie ne dépend que l'état courant comme entrées.

IMPLEMENTAION DU CONTROLE PAR UNE MACHINE A ETATS FINIS (3/8)

➢ Vue abstraite du contrôle de la machine à état finis

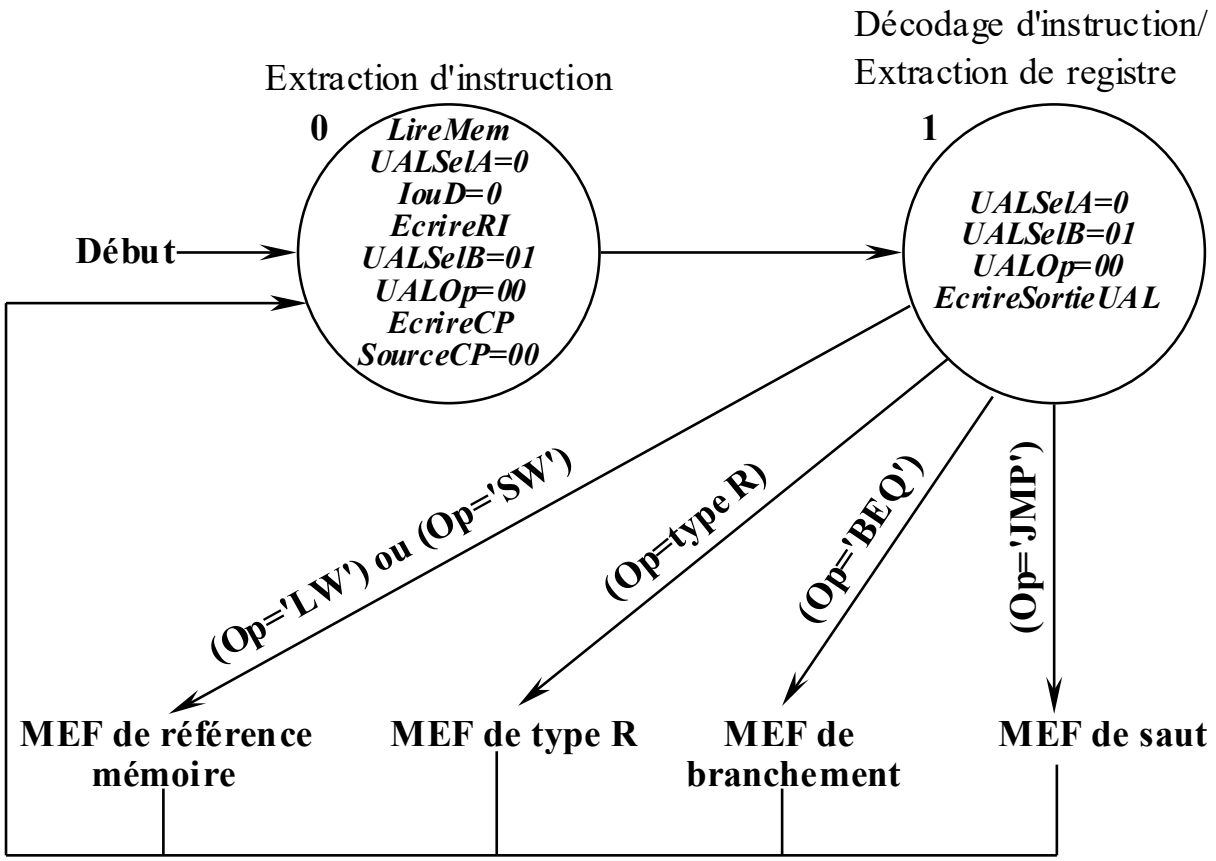


➢ Les 2 premières étapes sont indépendantes du type de l'instruction.

➢ Les séquences suivantes dépendent du code opération.

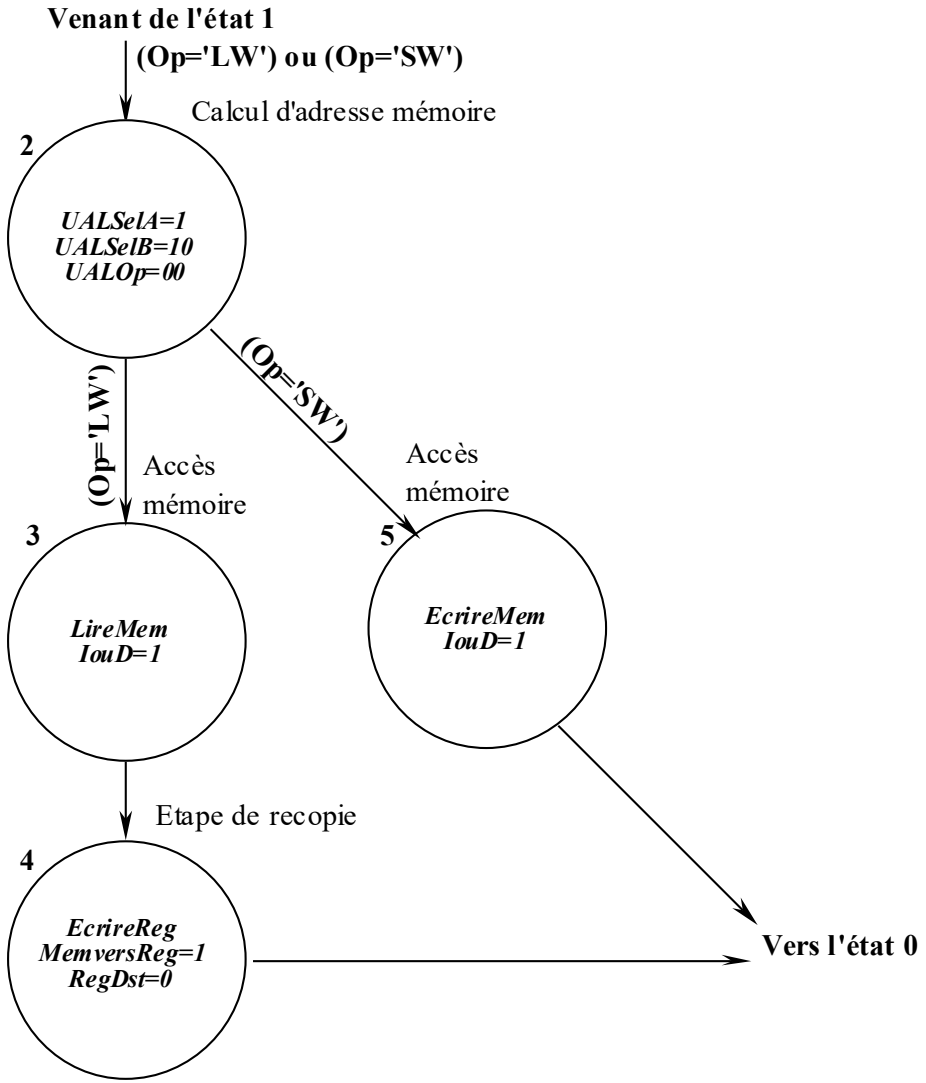
IMPLEMENTAION DU CONTROLE PAR UNE MACHINE A ETATS FINIS (4/8)

☞ Automate fini des deux premières étapes de la MEF



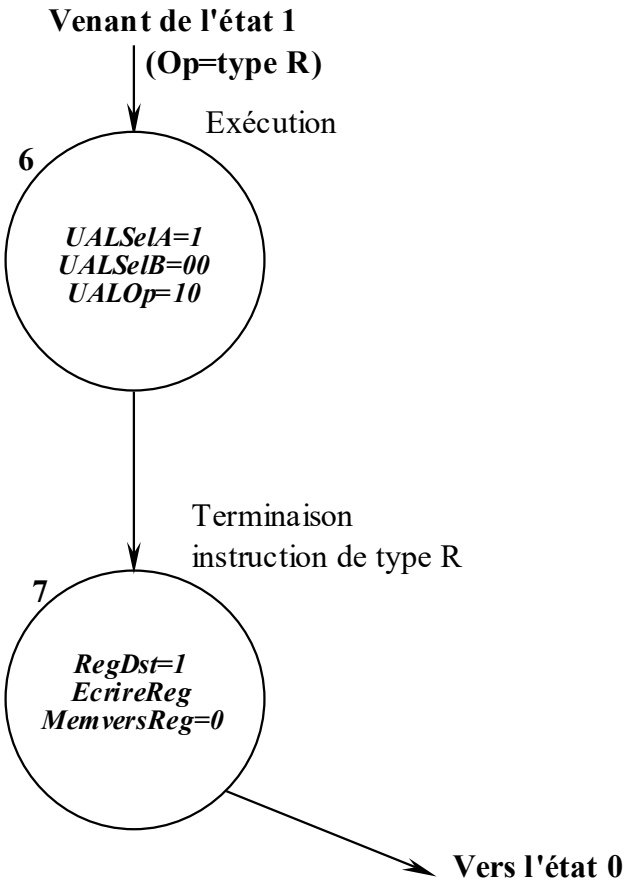
IMPLEMENTAION DU CONTROLE PAR UNE MACHINE A ETATS FINIS (5/8)

☞ Automate fini pour les références mémoires



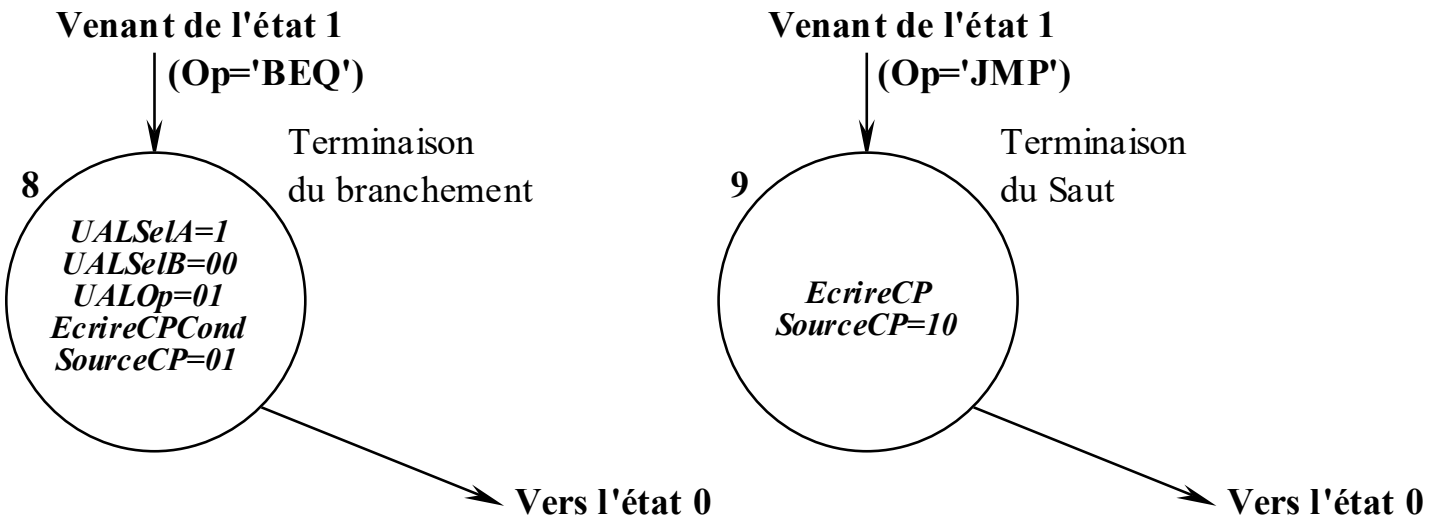
IMPLEMENTAION DU CONTROLE PAR UNE MACHINE A ETATS FINIS (6/8)

☞ Automate fini pour les instructions de type R



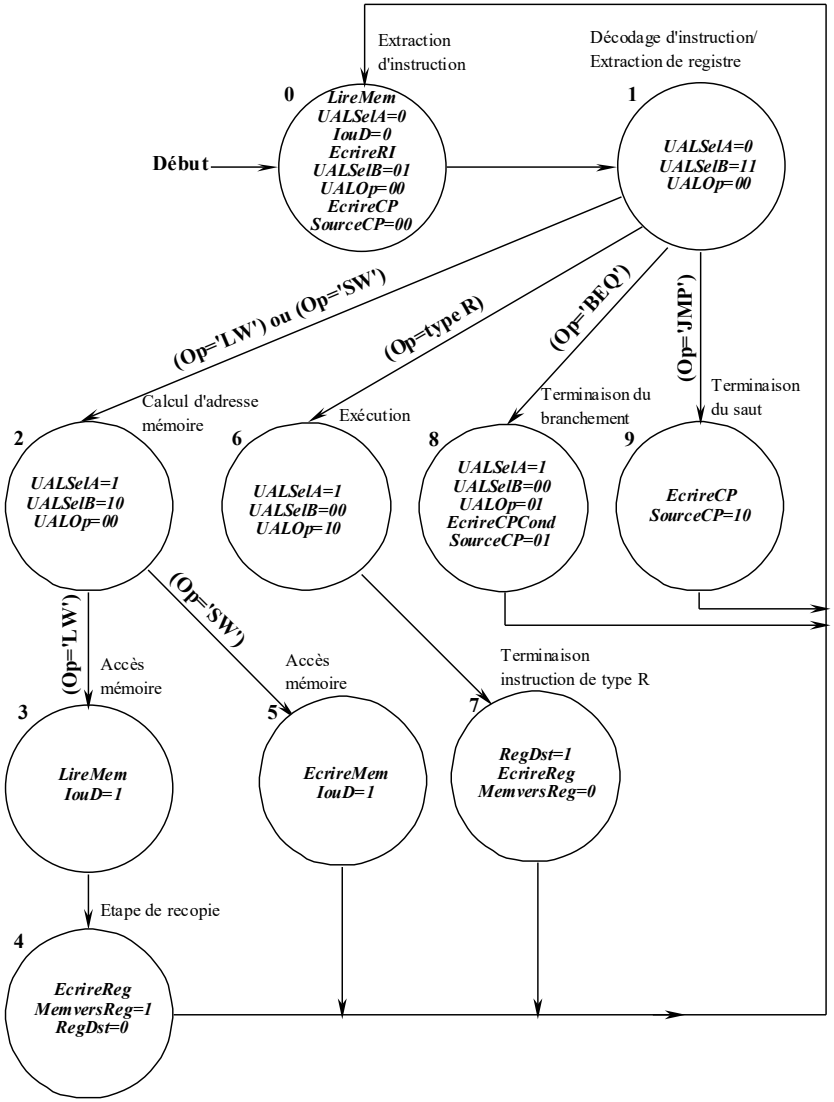
IMPLEMENTAION DU CONTROLE PAR UNE MACHINE A ETATS FINIS (7/8)

☞ Automate fini pour les instructions de branchement et de saut

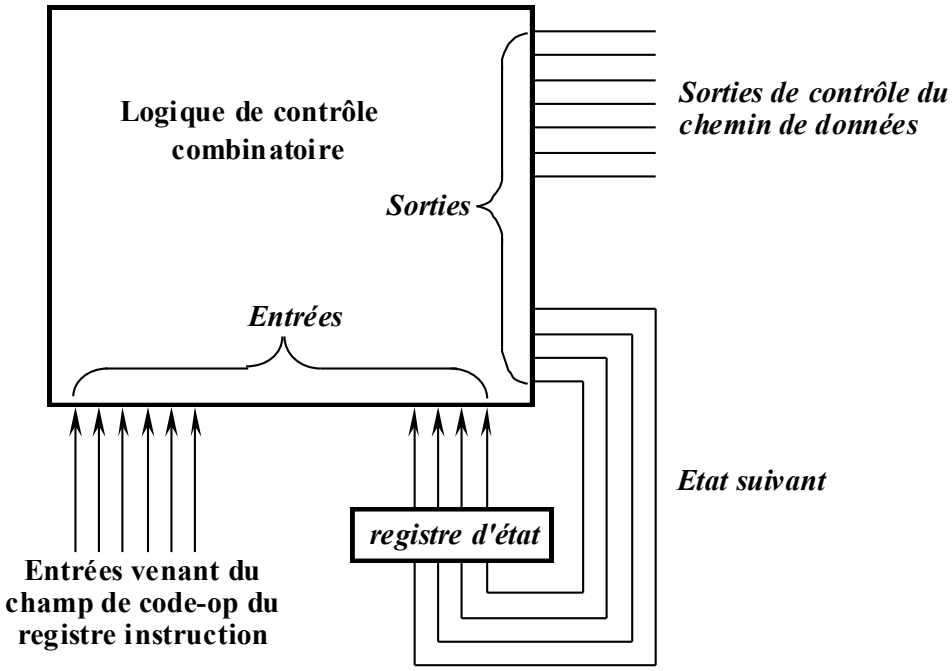


IMPLEMENTAION DU CONTROLE PAR UNE MACHINE A ETATS FINIS (8/8)

☞ MEF de l'organe de contrôle complet



ALLURE DE L'ORGANE DE CONTROLE



Le bloc de contrôle est totalement combinatoire. Il peut être défini comme une grande table (ROM ou RLP) fournissant la valeur des sorties en fonctions des entrées.

En fait la logique de contrôle implémente les deux parties de la MEF :

- ✓ Une partie définissant la fonction **Sortie** responsable du positionnement des signaux de contrôle du chemin de données qui dépendent seulement des bits de l'état courant.

- ✓ L'autre définissant la fonction **EtatSuivant** responsable du positionnement de la valeur de l'état suivant qui dépend des bits d'état courant et du code opération.

DETERMINATION DES FONCTIONS DE LA MEF Sortie & EtatSuivant

☞ Table des équations des fonctions Sortie & EtatSuivant

Sorties	Etats courants	Opération
EcrireCP	Etat0 + Etat9	
EcrireCPCond	Etat8	
IouD	Etat3 + Etat5	
LireMem	Etat0 + Etat3	
EcrireMem	Etat5	
EcrireRI	Etat0	
MemVersReg	Etat4	
SourceCP1	Etat9	
SourceCP0	Etat8	
UALOp1	Etat6	
UALOp0	Etat8	
UALSelB1	Etat1 + Etat2	
UALSelB0	Etat0 + Etat1	
UALSelA	Etat2 + Etat6 + Etat8	
EcrireReg	Etat4 + Etat7	
RegDst	Etat7	
EtatSuivant0	Etat4 + Etat5 + Etat7 + Etat8 + Etat9	
EtatSuivant1	Etat0	
EtatSuivant2	Etat1	(op= "lw") + (op= "sw")
EtatSuivant3	Etat2	(op= "lw")
EtatSuivant4	Etat3	
EtatSuivant5	Etat2	(op= "sw")
EtatSuivant6	Etat1	(op= "typeR")
EtatSuivant7	Etat6	
EtatSuivant8	Etat1	(op= "beq")
EtatSuivant9	Etat1	(op= "jmp")

MINTERMS DES SIGNAUX DE CONTROLE

E3 E2 E1 E0	Signaux de contrôle
0 0 0 0	Ecrire CP, **LireMem, EcrireRI, UALSelB0**
1 0 0 1	Ecrire CP, **SourceCP1**
1 0 0 0	EcrireCPCond, **SourceCP0, UALOp0, UALselA**
0 0 1 1	IouD, **LireMem**
0 1 0 1	IouD, **EcrireMem**
~~0 0 0 0~~	~~LireMem~~
~~0 0 1 1~~	
~~0 1 0 1~~	~~EcrireMem~~
~~0 0 0 0~~	~~EcrireRI~~
0 1 0 0	MemversReg, **EcrireReg**
~~1 0 0 1~~	~~SourceCP1~~
~~1 0 0 0~~	~~SourceCP0~~
0 1 1 0	UALOp1, **UALselA**
~~1 0 0 0~~	~~UALOp0~~
0 0 0 1	UALSelB1, **UALSelB0**
0 0 1 0	UALSelB1, **UALselA**
~~0 0 0 0~~	~~UALSelB0~~
~~0 0 0 1~~	
~~0 0 1 0~~	
~~0 1 1 0~~	~~UALselA~~
~~1 0 0 0~~	
~~0 1 0 0~~	~~EcrireReg~~
0 1 1 1	EcrireReg, **RegDst**
~~0 1 1 1~~	~~RegDst~~

TABLES DE VERITE DES SIGNAUX EtatSuivant

Table de vérité de ES3

	Op5	Op4	Op3	Op2	Op1	Op0	E3	E2	E1	E0	ES3	
beq	0	0	0	1	0	0	0	0	0	1	1	EtatSuivant 8
jmp	0	0	0	0	1	0	0	0	0	1	1	EtatSuivant 9

Table de vérité de ES2

	Op5	Op4	Op3	Op2	Op1	Op0	E3	E2	E1	E0	ES2	
typeR	0	0	0	0	0	0	0	0	0	1	1	EtatSuivant 6
sw	1	0	1	0	1	1	0	0	1	0	1	EtatSuivant 5
-	x	x	x	x	x	x	0	0	1	1	1	EtatSuivant 4
-	x	x	x	x	x	x	0	1	1	0	1	EtatSuivant 7

Table de vérité de ES1

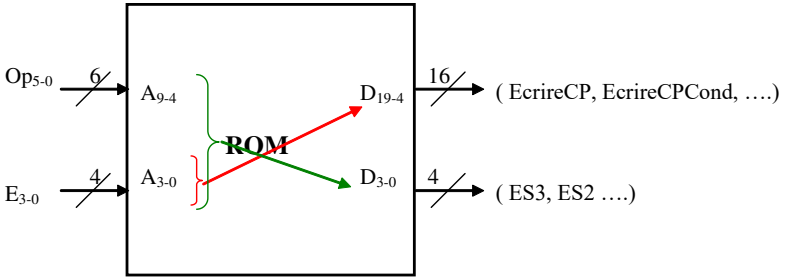
	Op5	Op4	Op3	Op2	Op1	Op0	E3	E2	E1	E0	ES1	
~~typeR~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	~~1~~	~~EtatSuivant 6~~
lw	1	0	0	0	1	1	0	0	0	1	1	EtatSuivant 2
sw	1	0	1	0	1	1	0	0	0	1	1	EtatSuivant 2
lw	1	0	0	0	1	1	0	0	1	0	1	EtatSuivant 3
-	~~x~~	~~x~~	~~x~~	~~x~~	~~x~~	~~x~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~EtatSuivant 7~~

Table de vérité de ES0

	Op5	Op4	Op3	Op2	Op1	Op0	E3	E2	E1	E0	ES0	
-	x	x	x	x	x	x	0	0	0	0	1	EtatSuivant 1
~~jmp~~	~~1~~	~~0~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	~~1~~	~~EtatSuivant 9~~
~~lw~~	~~1~~	~~0~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1~~	~~EtatSuivant 3~~
~~sw~~	~~1~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1~~	~~EtatSuivant 5~~
-	~~x~~	~~x~~	~~x~~	~~x~~	~~x~~	~~x~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~EtatSuivant 7~~

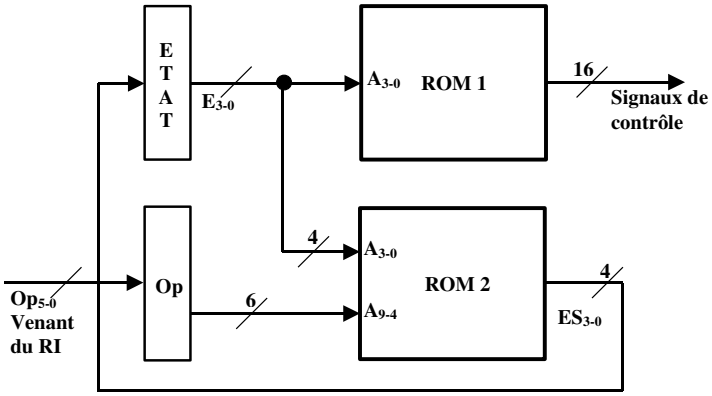
MISE EN ŒUVRE EN ROM (1/3)

☞ Cas d'une seule ROM



✓ Bus adresse = 10 bits, bus de données = 20 bits => Taille ROM = 20 Kbits
✓ ***Inconvénients*** : beaucoup de combinaisons inutiles, représentation du contenu difficile.

☞ Cas de deux ROMs : une ROM pour chaque fonction de la MEF



✓ Taille ROM1 = 256 bits, Taille ROM2 = 4 Kbits => ~ 4,3 Kbits
✓ ***Avantage*** **:** simplicité de représentation
✓ ***Inconvénient*** **:** encore beaucoup de pertes dues à la fonction **EtatSuivant**

MISE EN ŒUVRE EN ROM (2/3)

☞ Table de vérité des signaux de contrôle :

Sorties	Valeur des entrées E[3-0]									
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
EcrireCP	1	0	0	0	0	0	0	0	0	1
EcrireCPCond	0	0	0	0	0	0	0	0	1	0
IouD	0	0	0	1	0	1	0	0	0	0
LireMem	1	0	0	1	0	0	0	0	0	0
EcrireMem	0	0	0	0	0	1	0	0	0	0
EcrireRI	1	0	0	0	0	0	0	0	0	0
MemVersReg	0	0	0	0	1	0	0	0	0	0
SourceCP1	0	0	0	0	0	0	0	0	0	1
SourceCP0	0	0	0	0	0	0	0	0	1	0
UALOp1	0	0	0	0	0	0	1	0	0	0
UALOp0	0	0	0	0	0	0	0	0	1	0
UALSelB1	0	1	1	0	0	0	0	0	0	0
UALSelB0	1	1	0	0	0	0	0	0	0	0
UALSelA	0	0	1	0	0	0	1	0	1	0
EcrireReg	0	0	0	0	1	0	0	1	0	0
RegDst	0	0	0	0	0	0	0	1	0	0

☞ Cette table de vérité représente le contenu de la ROM 1.

4 bits de poids faible de l'adresse E[3-0]	Bits 19-4 du mots
0000	1001010000001000
0001	0000000000011000
:	:
:	:
1001	1000000100000000

MISE EN ŒUVRE EN ROM (3/3)

☞ Table de vérité des signaux EtatSuivant :

Code-op / Etat courant E[3-0]	type R	Jmp	Beq	Lw	Sw	
	000000	000010	000100	100011	101011	Autres valeurs
0000	*0001*	*0001*	*0001*	*0001*	*0001*	*0001*
0001	*0110*	*1001*	*1000*	*0010*	*0010*	*Illégale*
0010	xxxx	xxxx	xxxx	*0011*	*0101*	*Illégale*
0011	0100	0100	0100	*0100*	0100	*Illégale*
0100	0000	0000	0000	*0000*	0000	*Illégale*
0101	0000	0000	0000	0000	*0000*	*Illégale*
0110	*0111*	0111	0111	0111	0111	*Illégale*
0111	*0000*	0000	0000	0000	0000	*Illégale*
1000	0000	0000	*0000*	0000	0000	*Illégale*
1001	0000	*0000*	0000	0000	0000	*Illégale*

☞ Cette table de vérité représente le contenu de la ROM 2 (50 combinaisons de 4 bits utiles !!!)

MISE EN ŒUVRE EN RLP TYPE PLA (1/2)

☞ Réduit la quantité de logiques au stricte nécessaire => seules les combinaisons d'entrées nécessaires seront encodées.

☞ Chaque sortie est un OR d'un ou de plusieurs minterms, un minterm étant le AND d'une ou de plusieurs entrées.

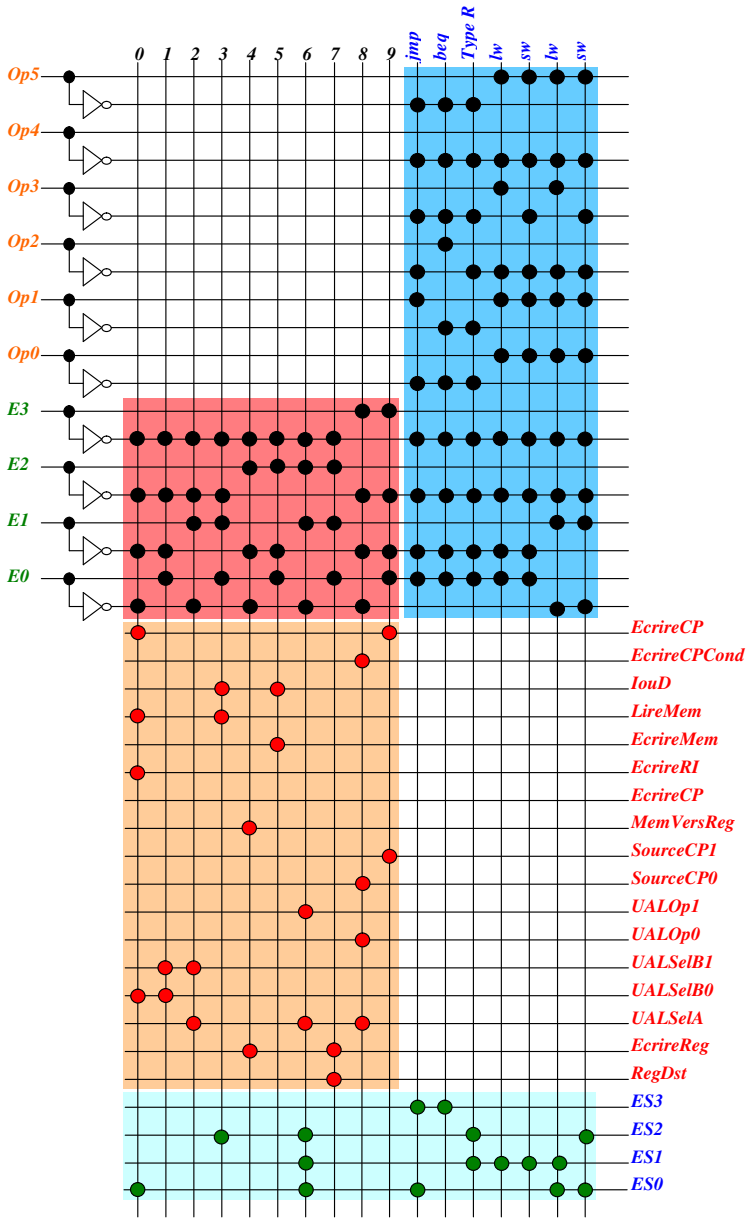
☞ Au contraire d'une ROM seules les entrées des tables de vérité qui produisent une sortie active sont nécessaires et chaque minterm n'apparaît qu'une seule fois.

☞ Taille PLA = (# Entrées x # termes produit) + (# Sorties x # termes produit).

☞ Dans notre exemple on a 17 minterms (termes produit)

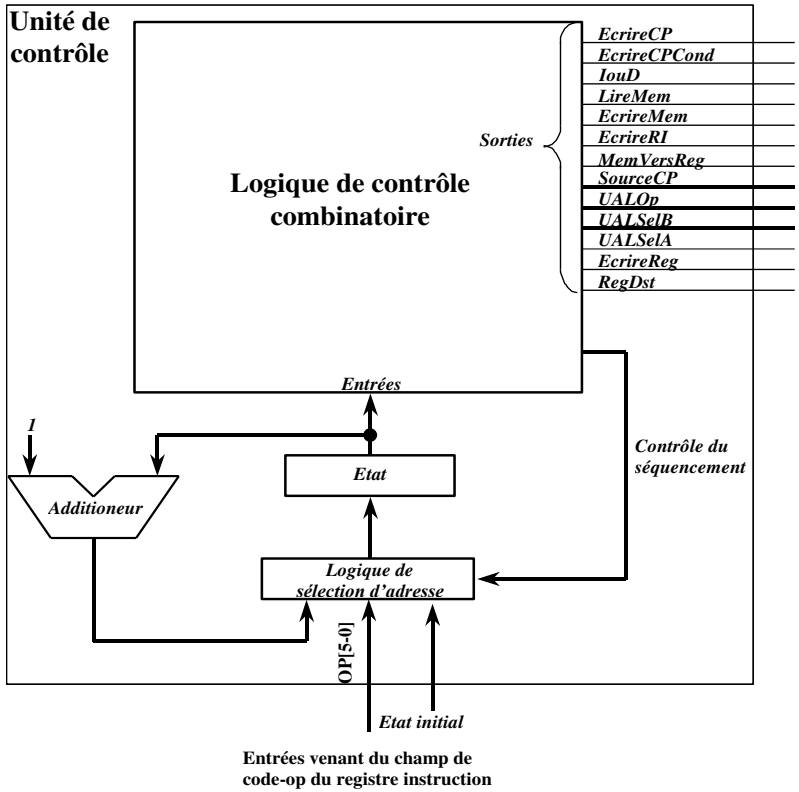
- ✓ **10 dépendent des bits de l'état courant et**
- ✓ **7 dépendent des bits code-op + les bits de l'état courant.**
- ✓ **Cas d'un seul PLA ➔ taille = (10 x 17) + (20 x 17) = 510 cellules.**
- ✓ **Cas de deux PLA :**
  - taille PLA 1 (4 entrées, 16 sorties et 10 minterms) = (4 x 10) + (16 x 10) = 200 cellules
  - taille PLA 2 (10 entrées, 4 sorties et 7 minterms) = (10 x 7) + (4 x 7) = 98 cellules

MISE EN ŒUVRE EN RLP TYPE PLA (2/2)



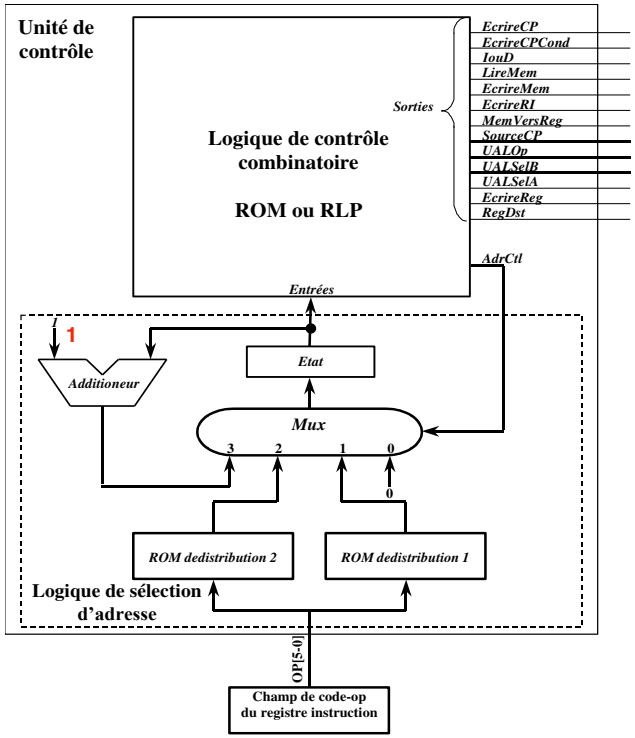
MISE EN ŒUVRE DE LA FONCTION ETAT SUIVANT AVEC UN SEQUENCEUR (1/4)

- Encoder la fonction **EtatSuivant** de manière explicite par un compteur qui fournira l'état immédiatement suivant.
- Un additionneur est utilisé, transformé en compteur en incrémentant l'état courant toujours dans l'ordre numérique.
- ***Problème*** *:* la MEF effectue parfois des **« branchements ».**
- ***Solution*** *:* des lignes de contrôles associées avec chaque mot de contrôle détermineront comment l'état suivant est choisi. Cette opération est dite **« distribution ».**



MISE EN ŒUVRE DE LA FONCTION ETAT SUIVANT AVEC UN SEQUENCEUR (2/4)

☞ Mise en œuvre de l'opération de distribution : utiliser une ROM ou un PLA à chaque fois qu'un état la MEF peut avoir plusieurs états successeurs possible.



Valeur AdrCtl	Action
0	Choisir l'état 0
1	Distribuer avec ROM 1
2	Distribuer avec ROM 2
3	Choisir l'état incrémenté

MISE EN ŒUVRE DE LA FONCTION ETAT SUIVANT AVEC UN SEQUENCEUR (3/4)

☞ Contenu de la ROM de distribution 1

Code-Op (adresse)	Nom du code-Op	Contenu de la ROM
000000	formatR	**0110**
000010	jmp	**1001**
000100	Beq	**1000**
100011	Lw	**0010**
101011	Sw	**0010**

☞ Contenu de la ROM de distribution 2

Code-Op (adresse)	Nom du code-Op	Contenu de la ROM
100011	Lw	**0011**
101011	Sw	**0101**

☞ Taille des ROM = de 64 x 4 bits

☞ Valeurs des lignes de contrôle d'adresse dans le mot de contrôle correspondant à chaque état.

# de l'état	Action de contrôle d'adresse	Valeur de AdrCtl
0	Utiliser l'état incrémenté	**3**
1	Utiliser ROM de distribution 1	**1**
2	Utiliser ROM de distribution 2	**2**
3	Utiliser l'état incrémenté	**3**
4	Utiliser l'état initial (état 0)	**0**
5	Utiliser l'état initial (état 0)	**0**
6	Utiliser l'état incrémenté	**3**
7	Utiliser l'état initial (état 0)	**0**
8	Utiliser l'état initial (état 0)	**0**
9	Utiliser l'état initial (état 0)	**0**

MISE EN ŒUVRE DE LA FONCTION ETAT SUIVANT AVEC UN SEQUENCEUR (4/4)

☞ Contenu de la ROM de contrôle avec un compteur explicite

# d'état (adresse)	Bits de 17 à 2 du mot de contrôle	Bits 1 et 0 du mot de contrôle
0	1001010000001000	11
1	0000000000011000	01
2		10
3		11
4		00
5		00
6		11
7		00
8		00
9	1000000100000000	00

☞ La taille globale de la logique de contrôle dans la mise en œuvre avec un séquenceur est la suivante :

- ✓ Taille ROM de contrôle 16 x 18 = 288 bits ➔ 180 bits utiles
- ✓ Taille ROM 1 de distribution 64 x 4 = 256 bits Total = 800 bits ➔ 20 bits utiles Total =188 bits
- ✓ Taille ROM 2 de distribution 64 x 4 = 256 bits ➔ 8 bits utiles

IMPLEMENTAION DU CONTROLE PAR MICROPROGRAMME

➢ **La conception du contrôle par microprogrammation nécessite :**

- La définition de micro-instructions symboliques ;
- La conception d'un micro-assembleur ;
- L'écriture du microprogramme complet ;
- La traduction par le micro-assembleur du microprogramme complet en une logique de contrôle.

➢ **Le format d'une micro-instruction doit être choisi afin :**

- de simplifier la représentation, et la lisibilité du microprogramme ;
- de rendre impossible l'écriture de micro-instructions incohérentes.

Une micro-instruction est incohérente si elle exige qu'un signal soit positionné à 2 valeurs différentes.

➢ **Définition d'une micro-instruction :**

- définir le nombre de champs dont doit disposer une micro-instruction ;
- définir les signaux de contrôle affectés à chaque champ.

DEFINITION D'UN FORMAT DE MICRO-INSTRUCTION

Un format fixe et un découpage possible de la micro-instruction est le suivant :

Etiquette	Contrôle UAL	SRC1	SCR2	Contrôle registres	Mémoire	Contrôle EcrireCP	Séquencement

La signification de chaque champ de la micro-instruction est donnée dans le tableau suivant :

Nom du champ	Fonction du champ
Contrôle UAL	Spécifie l'opération effectuée par l'UAL pendant cette période d'horloge.
SRC1	Spécifie la source du premier opérande de l'UAL.
SRC2	Spécifie la source du deuxième opérande de l'UAL
Contrôle registres	Spécifie lecture /écriture dans le banc de registres, la source et la valeur en cas d'une écriture.
Mémoire	Spécifie lecture ou écriture et l'adresse source.
Contrôle EcrireCP	Spécifie l'écriture du CP.
Séquencement	Spécifie comment choisir la prochaine micro-instruction à exécuter.

Le champ étiquette permet de repérer chaque micro-instruction, il représente son mnémonique.

METHODES DE SEQUENCEMENT

- **Ext** : pour l'extraction
- **Déc** : pour le décodage
- **FormatR1** : pour les instructions de type R
- **Mem1** : pour les instructions de référence mémoire
- **BEQ1** : pour l'instruction de branchement si égal
- **JUMP1** : pour l'instruction de saut

On dispose de trois méthodes différentes pour le choix de la prochaine micro-instruction à exécuter :

1. Aller l'adresse de la micro-instruction suivante : ceci est repéré dans le champ séquencement par ***Séq***.

2. Aller à l'exécution de **l'instruction suivante** : ceci est repéré dans le champ séquencement par ***Ext***.

3. Aller la prochaine micro-instruction en fonction du code opération : ceci est repéré dans le champ séquencement par ***Distribuer i*** (où ***i*** est le numéro de la table de distribution).

REPRESENTATION SYMBOLIQUE DES VALEURS DES DIFFERENTS CHAMPS DE LA MICRO-INSTRUCTION

Nom du champ	Valeurs du champ	Fonction du champ avec valeur spécifique
Contrôle UAL	***Add***	Faire additionner l'UAL
	Code Fonction	Utiliser le code de fonction de l'UAL pour déterminer le contrôle de l'UAL.
	Soust	Faire soustraire l'UAL
SRC1	***CP***	Utiliser le CP comme première entrée de l'UAL.
	A	Le registre A (obtenu en utilisant le champ rs) est la première entrée de l'UAL.
SRC2	***4***	Utiliser la valeur 4 comme seconde entrée de l'UAL.
	Extension	Utiliser la sortie de l'unité d'extension de signe comme seconde entrée de l'UAL.
	ExtenDec	Utiliser la sortie de l'unité de décalage de 2 comme seconde entrée de l'UAL.
	B	Le registre B (obtenu en utilisant le champ rs) est la seconde entrée de l'UAL.
Contrôle Registres	***Lire***	Lire 2 registres A et B en utilisant les champs rs et rt du RI.
	Ecrire UAL	La sortie l'UAL est écrite dans le banc de registres en utilisant le champ rd du RI.
	Ecrire Mémoire	La donnée-mémoire est écrite dans le banc en utilisant le champ rt du RI.
	RI	La donnée lue en mémoire est écrite dans le registre instruction.
Mémoire	***Lire CP***	Lire en mémoire en utilisant le CP comme adresse.
	Lire UAL	Lire en mémoire en utilisant le contenu du registre SortieUAL comme adresse.
	Ecrire UAL	Ecrire en mémoire en utilisant le contenu du registre SortieUAL comme adresse.
Contrôle EcrireCP	***UAL***	Ecrire la sortie de l'UAL dans le CP.
	SortieUAL-cond	Si la sortie Z de l'UAL est à 1, écrire dans le CP le contenu du registre SortieUAL.
	Adresse de saut	Ecrire dans le CP l'adresse de saut contenue dans l'instruction.
Séquencement	***Séq***	Choisir la prochaine micro-instruction séquentiellement.
	Ext	Aller à la première micro-instruction pour démarrer une nouvelle instruction.
	Distribuer i	Distribuer en utilisant la ROM spécifiée par i (1 ou 2).

CREATION DES MICRO-INSTRUCTIONS (1/2)

Micro-instruction d'extraction

Etiquette	Contrôle UAL	SRC1	SCR2	Contrôle registres	Mémoire	Contrôle EcrireCP	Séquencement
Ext	***Add***	***CP***	***4***	***RI***	***Lire CP***	***UAL***	***Séq***

Micro-instruction de décodage

Etiquette	Contrôle UAL	SRC1	SCR2	Contrôle registres	Mémoire	Contrôle EcrireCP	Séquencement
Déc	***Add***	***CP***	***ExtenDec***	***Lire***			***Distribuer 1***

Micro-instructions pour les instructions de référence mémoire

Etiquette	Contrôle UAL	SRC1	SCR2	Contrôle registres	Mémoire	Contrôle EcrireCP	Séquencement
Mem1	***Add***	***A***	***Extension***				***Distribuer 2***
LW2					***Lire UAL***		***Séq***
				Ecrire Mémoire			***Ext***
SW2					***Ecrire UAL***		***Ext***

CREATION DES MICRO-INSTRUCTIONS (2/2)

Micro-instructions pour les instructions de type R

Etiquette	Contrôle UAL	SRC1	SCR2	Contrôle registres	Mémoire	Contrôle EcrireCP	Séquencement
FormatR1	***Code Fonction***	***A***	***B***				***Séq***
				Ecrire UAL			***Ext***

Micro-instructions pour l'instruction de branchement si égal

Etiquette	Contrôle UAL	SRC1	SCR2	Contrôle registres	Mémoire	Contrôle EcrireCP	Séquencement
BEQ1	***Soust***	***A***	***B***			***SortieUAL-cond***	***Ext***

Micro-instructions pour l'instruction de saut

Etiquette	Contrôle UAL	SRC1	SCR2	Contrôle registres	Mémoire	Contrôle EcrireCP	Séquencement
JUMP1						***Adresse de saut***	***Ext***

MICROPROGRAMME COMPLET

Etiquette	Contrôle UAL	SRC1	SCR2	Contrôle registres	Mémoire	Contrôle EcrireCP	Séquencement
Ext	*Add*	CP	4	RI	Lire CP	UAL	*Séq*
Déc	*Add*	*CP*	*ExtenDec*	*Lire*			*Distribuer 1*
Mem1	*Add*	*A*	*Extension*				*Distribuer 2*
LW2					*Lire UAL*		*Séq*
-				*Ecrire Mémoire*			*Ext*
SW2					*Ecrire UAL*		*Ext*
FormatR1	*Code Fonction*	*A*	*B*				*Séq*
-				*Ecrire UAL*			*Ext*
BEQ1	*Soust*	*A*	*B*			*SortieUAL-cond*	*Ext*
JUMP1						*Adresse de saut*	*Ext*

TABLE DES SYMBOLES : VALEURS SYMBOLIQUES ⟷ VALEURS BINAIRES

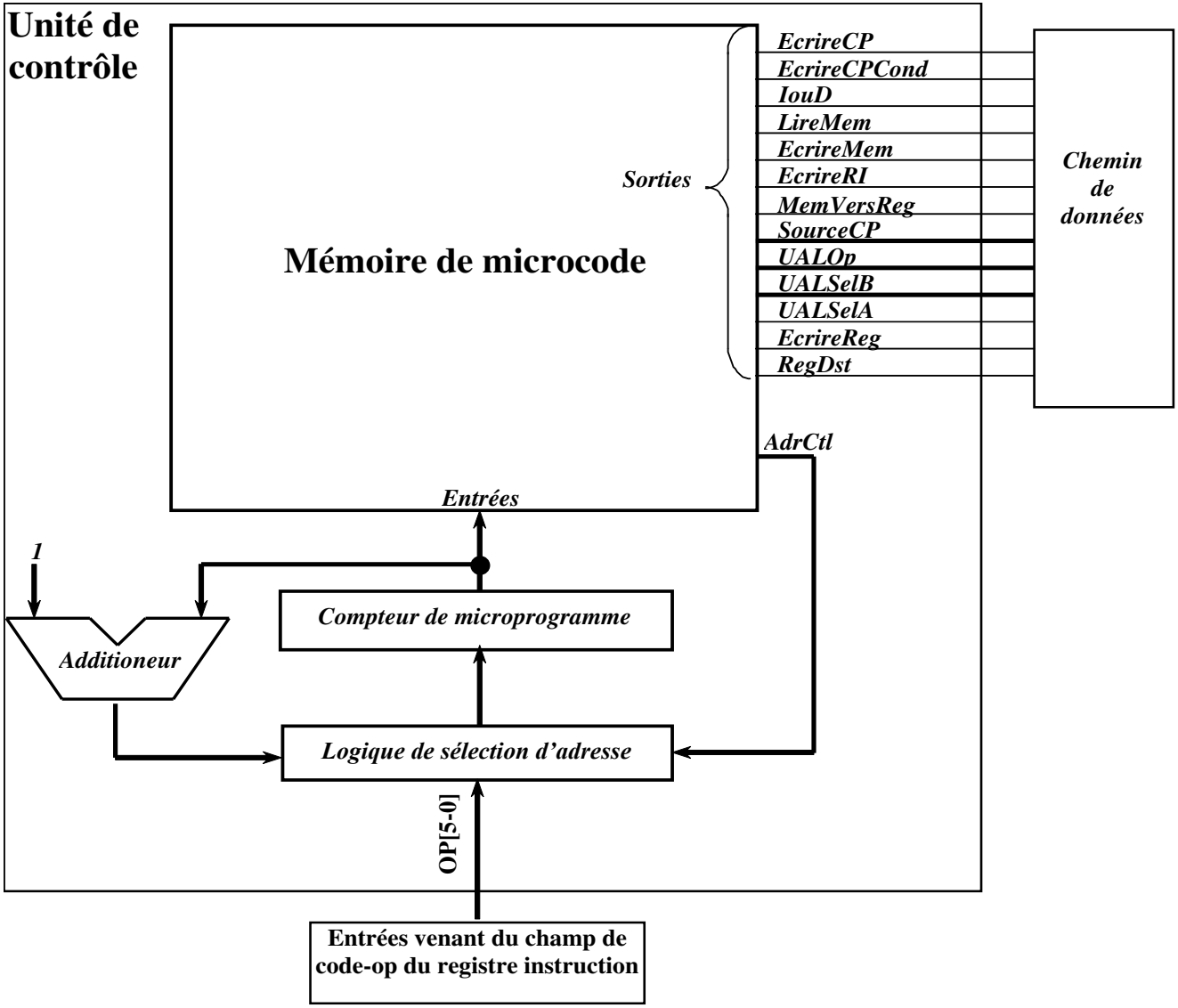
Nom du champ	Valeur	Signaux actifs	Commentaire
Contrôle UAL	Add	UALOp = 00	L'UAL additionne
	Soust	UALOp = 01	L'UAL soustrait
	Code Fonction	UALOp = 10	L'UAL utilise le code fonction
SRC1	CP	UALSelA = 0	Le CP est le premier opérande de l'UAL
	A	UALSelA = 1	Le registre A (rs) est le premier opérande de l'UAL
SRC2	B	UALselB = 00	Le registre B (rt) est le second opérande de l'UAL
	4	UALselB = 01	4 est le second opérande de l'UAL
	Extension	UALselB = 10	Extension-signe (RI[15-0]) est le second opérande de l'UAL
	ExtenDéc	UALselB = 11	Extension-signe (RI[15-0])<<2 est le second opérande de l'UAL
Contrôle Registres	Lecture		Lecture de 2 registres A et B pointés par rs et rt
	Ecrire UAL	EcrireReg RegDst = 1 MemVersReg = 0	Ecrire la sortie de l'UAL dans le registre spécifié par rd
	Ecrire Mémoire	EcrireReg RegDst = 0 MemVersReg = 1	Ecrire la donnée-mémoire dans le registre spécifié par rt
	RI	EcrireRI	Provoque l'écriture de RI à partir de la mémoire
Mémoire	Lire CP	LireMem IouD = 0	Lire la mémoire ; l'adresse est dans le CP
	Lire UAL	LireMem IouD = 1	Lire la mémoire ; l'adresse est dans SortieUAL
	Ecrire UAL	EcrireMem IouD = 1	Ecrire la mémoire ; l'adresse est dans SortieUAL
Contrôle EcrireCP	UAL	SourceCP = 00 EcrireCP	Ecrire SortieUAL dans le CP
	SortieUAL-cond	SourceCP = 01 EcrireCPCond	Si sortie Zéro de l'UAL est active, alors écrire dans le CP la valeur contenu dans SortieUAL
	Adresse de saut	SourceCP = 10 EcrireCP	Ecrire l'adresse de destination du saut dans le CP
Séquencement	Séq	AdrCtl = 11	La prochaine instruction suit séquentiellement
	Ext	AdrCtl = 00	La prochaine instruction est celle de l'étiquette ExtDéc
	Distribuer 1	AdrCtl = 01	ROM de distribution 1
	Distribuer 2	AdrCtl = 10	ROM de distribution 2

TRADUCTION DU MICROPROGRAMME EN SIGNAUX (VALEURS BINAIRES)

Etiquette	Contrôle UAL	SRC1	SCR2	Contrôle registres	Mémoire	Contrôle EcrireCP	Séquencement
	UALOp1 UALOp0	UALselA	UALselB1 UALSelB0	EcrireRI EcrireReg RegDst MemversReg	LireMem EcrireMem IouD	SourceCP1 SourceCP0 EcrireCPCond EcrireCP	AdrCtl1 AdrCtl0
ExtDéc	00	0	01	1 0 x x	1 0 0	00 0 1	11
	00	0	11	0 0 x x	0 0 x	xx 0 0	01
Mem1	00	1	10	0 0 x x	0 0 x	xx 0 0	10
LW2	xx	x	xx	0 0 x x	1 0 1	xx 0 0	11
	xx	x	xx	0 1 0 1	0 0 x	xx 0 0	00
SW2	xx	x	xx	0 0 x x	0 1 ~~0~~ 1	xx 0 0	00
FormatR1	10	1	00	0 0 x x	0 0 x	xx 0 0	11
	xx	x	xx	0 1 1 0	0 0 x	xx 0 0	00
BEQ1	01	1	00	0 0 x x	0 0 x	01 1 0	00
JUMP1	xx	x	xx	0 0 x x	0 0 x	10 0 1	00

X : valeur indifférente

IMPLEMENTATION DU MICROCODE EN ROM ET ETAT SUIVANT AVEC UN SEQUENCEUR



LES EXCEPTIONS

☞ Critères des exceptions :

- ✓ **synchrone ou asynchrone**
- ✓ **requête utilisateur ou imposée**
- ✓ **Masquable ou non par l'utilisateur**
- ✓ **Pendant ou entre les instructions**
- ✓ **Reprise ou fin**

☞ Principaux types d'exception :

- ✓ Requêtes d'E/S
- ✓ Appel système
- ✓ Trace et point d'arrêt
- ✓ Débordement ou anomalie arithmétique
- ✓ Défaut de page
- ✓ Accès mémoire non aligné, violation de protection, instruction non définie
- ✓ Panne matérielle

GESTION DES EXCEPTIONS

☞ **Dès l'arrivée d'une exception les actions menées sont :**

- ✓ **Sauvegarde de l'adresse de l'instruction qui a provoquée l'exception dans un registre (CEP)**
- ✓ **Transfert du contrôle à l'OS en spécifiant une adresse particulière.**
- ✓ **Détermination de la cause de l'exception**

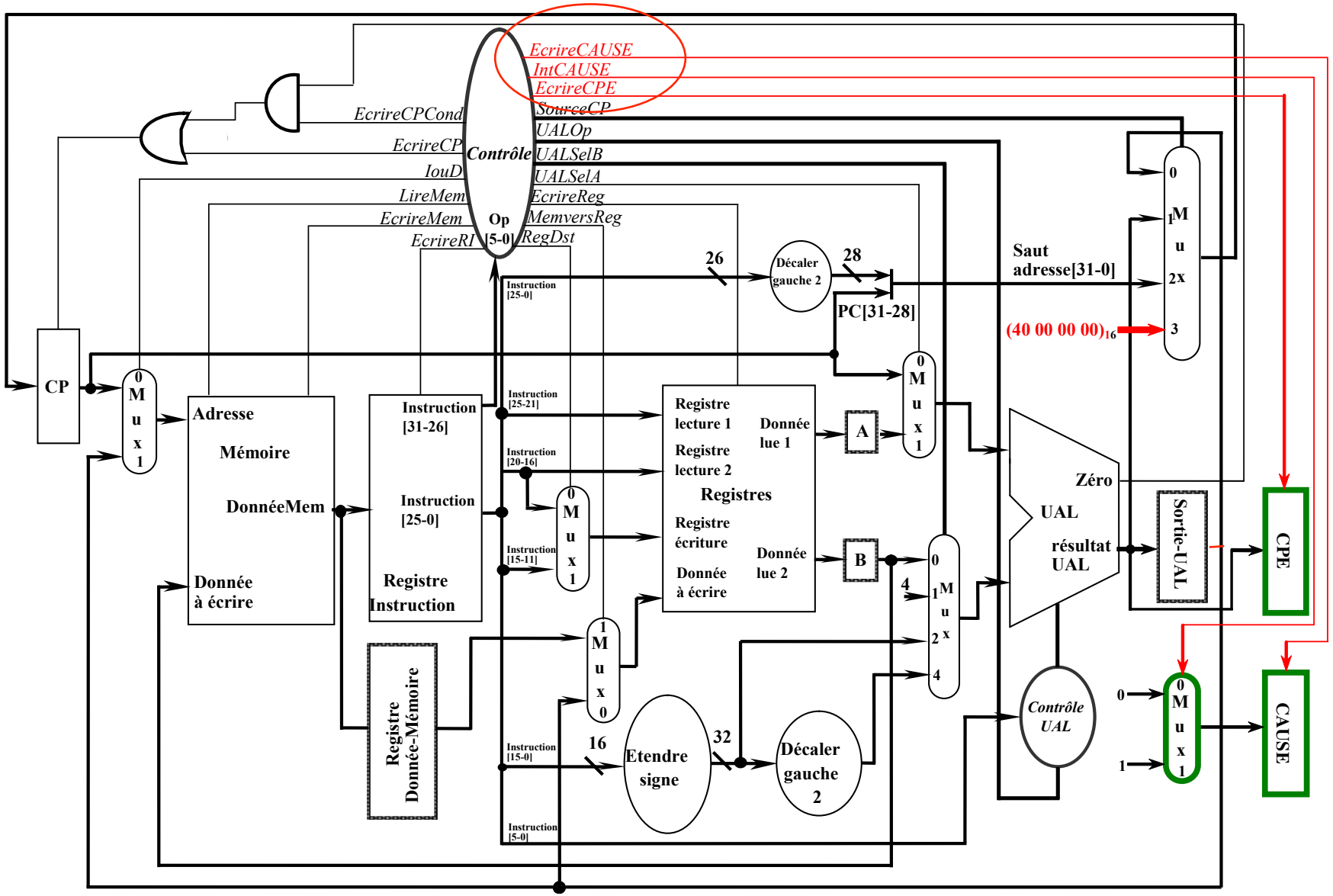
2 techniques possibles :

- **utilisation d'un registre d'état indiquant la ou les causes de l'exception avec une seule adresse pour toutes les exceptions.**
- **utilisation d'exceptions vectorisées : une adresse spécifique pour chaque cause.**

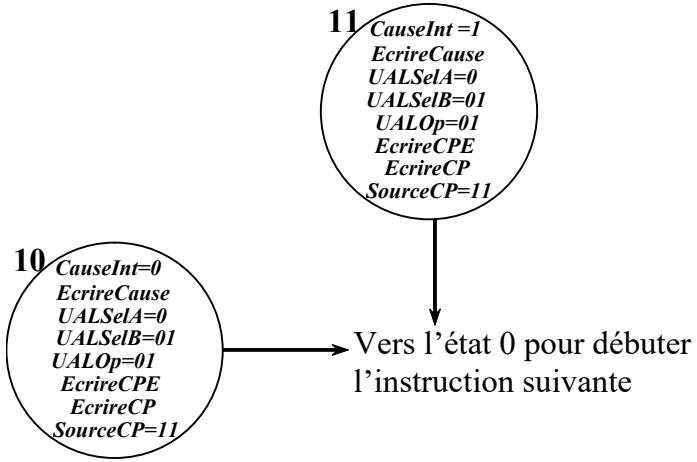
☞ **Actions que peut mener l'OS :**

- ✓ **fournir un service particulier au programme utilisateur, répondre de façon prédéfinie à un débordement.**
- ✓ **ou stopper l'exécution du programme et signaler une erreur.**
- ✓ **Le traitement de l'exception terminé, l'OS peut terminer le programme ou poursuivre son exécution en utilisant le CEP.**

MATERIEL POUR LA PRISE EN COMPTE DE L'EXCEPTION DEBORDEMENT ET INSTRUCTION INDEFINIE

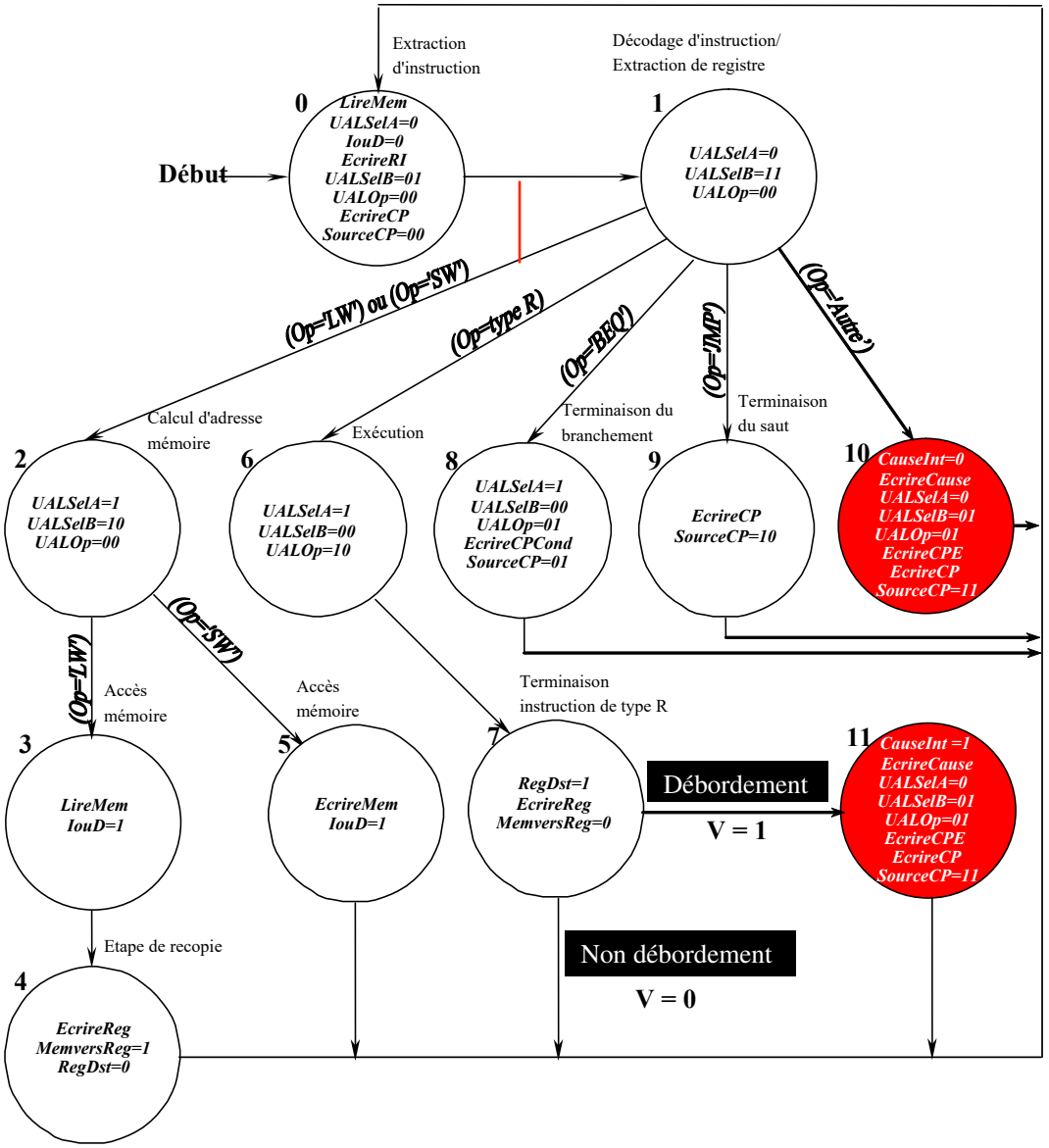


AUTOMATE FINI POUR LES DEBORDEMENTS ET LES INSTRUCTIONS INDEFINIES

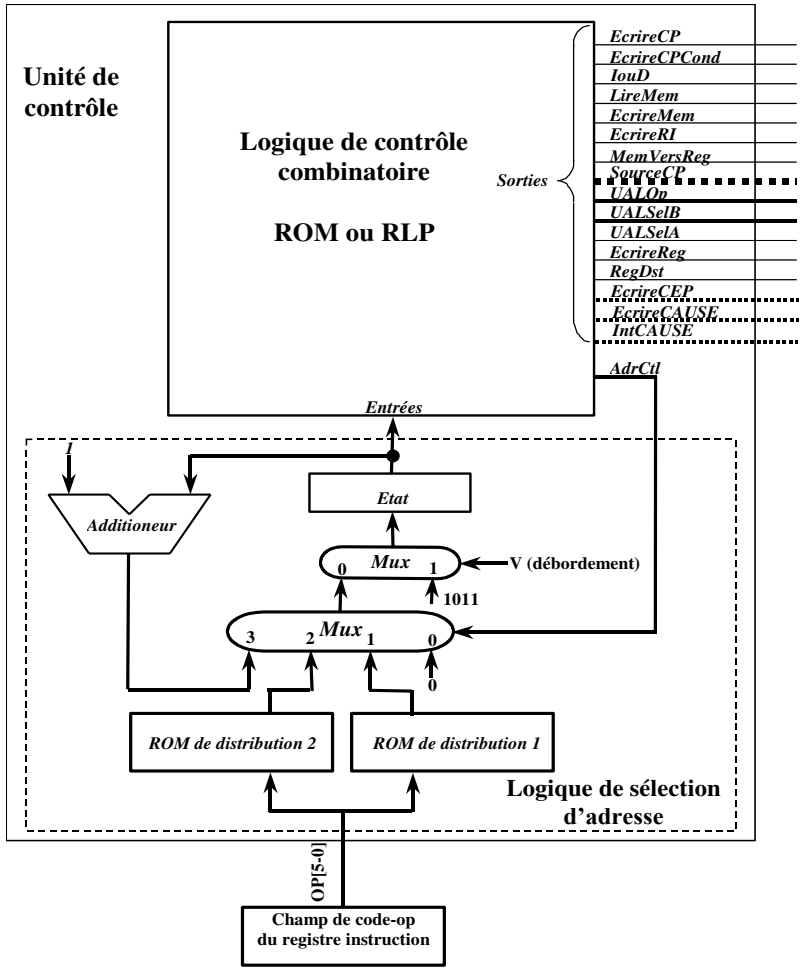


Opérations	*Signaux de contrôle activés*
Calculer l'adresse de l'instruction qui a provoquée l'exception CEP = CP-4	***UALSelA = 0*** (choix de CP) ***UALSelB = 01*** (choix de 4) ***UALOp = 01*** (force soustraction UAL)
Stocker SortieUAL dans le CPE.	***EcrireCPE = 1***
Si débordement V = 1 alors	***IntCAUSE = 1*** ***EcrireCAUSE = 1***
Si instruction indéfinie (déterminée par le contrôle) alors	***IntCAUSE = 0*** ***EcrireCAUSE = 1***
Charger le CP avec l'adresse d'exception $(40\ 00\ 00\ 00)_{16}$	***SourceCP = 11***

LES EXCEPTIONS : SPECIFICATION COMPLETE DE LA MEF



LOGIQUE DE CONTROLE INCLUANT LES EXCEPTIONS



Valeur AdrCtl	Action
0	Choisir l'état 0
1	Distribuer avec ROM 1
2	Distribuer avec ROM 2
3	Choisir l'état incrémenté

ROM DE DISTRIBUTION INCLUANT LES EXCEPTIONS

☞ Contenu de la ROM de distribution 1

Code-Op (adresse)	Nom du code-Op	Contenu de la ROM
000000	formatR	0110
000010	jmp	1001
000100	Beq	1000
100011	Lw	0010
101011	Sw	0010
xxxxxx	**autre**	**1010**

☞ Contenu de la ROM de distribution 2 ne change pas

☞ Valeurs des lignes de contrôle d'adresse dans le mot de contrôle correspondant à chaque état.

# de l'état	Action de contrôle d'adresse	Valeur de AdrCtl
0	Utiliser l'état incrémenté	3
1	Utiliser ROM de distribution 1	1
2	Utiliser ROM de distribution 2	2
3	Utiliser l'état incrémenté	3
4	Utiliser l'état initial (état 0)	0
5	Utiliser l'état initial (état 0)	0
6	Utiliser l'état incrémenté	3
7	Utiliser l'état initial (état 0)	0
8	Utiliser l'état initial (état 0)	0
9	Utiliser l'état initial (état 0)	0
10	**Utiliser l'état initial (état 0)**	**0**
11	**Utiliser l'état initial (état 0)**	**0**

ROM DES SIGNAUX DE CONTROLE INCLUANT LES EXCEPTIONS

☞ **Contenu de la ROM de contrôle avec un compteur explicite**

# d'état (adresse)	Bits de 17 à 2 du mot de contrôle	Bits 1 et 0 du mot de contrôle
0	10010100000010000*00*	11
1	00000000000110000*00*	01
2		10
3		11
4		00
5		00
6		11
7		00
8		00
9	10000001000000000*00*	00
10	**1000000110101000*10***	**00**
11	**1000000110101000*11***	**00**

☞ **La taille globale de la logique de contrôle dans la mise en œuvre avec un séquenceur est la suivante :**

- ✓ Taille ROM de contrôle = 16 x 21 = 336 bits ➔ 210 bits utiles
- ✓ Taille ROM 1 de distribution 64 x 4 = 256 bits Total = 848 bits ➔ 256 bits utiles Total = 474 bits
- ✓ Taille ROM 2 de distribution 64 x 4 = 256 bits ➔ 8 bits utiles

LOGIQUE DE DISTRIBUTION DES PHASES