

UE ARC1 L3 Informatique

Unités de contrôle

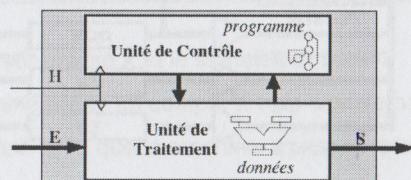
L3Info - ARC1

1

Principes généraux

- Principe : complexe = simple + simple
 - Décomposer un système complexe en deux parties

Unité de Traitement (UT)	Unité de Contrôle (UC)
<ul style="list-style-type: none"> • Traitements élémentaires sur les données • Opérateurs - Registres • Chemins de données 	<ul style="list-style-type: none"> • Séquencement des opérations



- Similaire à un programme qui agit sur des données

L3Info - ARC1

3

Plan du chapitre

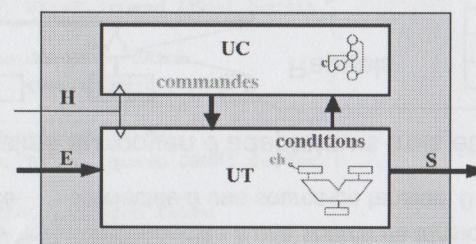
1. Décomposition d'un calculateur
 1. Principes généraux
 2. Méthode de décomposition
 3. Exemple
2. Réalisation des unités de contrôle
 1. Structure générale d'une unité de contrôle
 2. UC à bascules
 3. UC à compteur chargeable
 4. UC micro-programmée

L3Info - ARC1

2

Décomposition UT/UC

- Commandes et conditions

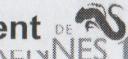


Commandes	Générées par l'UC Définissent les opérations effectuées dans le cycle
Conditions	Issues de l'UT et testées par l'UC Influencent le séquencement de l'UC

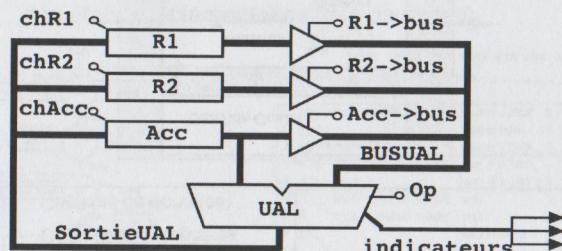
L3Info - ARC1

4

Décomposition UC/UT : l'Unité de Traitement



- UT = mémoires/registres, opérateurs, interconnexions



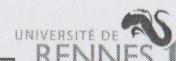
Registres	Opérateurs	Interconnexions	
		simple	bus*
R1, R2, Acc	UAL	SortieUAL	BUSUAL

* pas de panique, c'est expliqué au transparent suivant

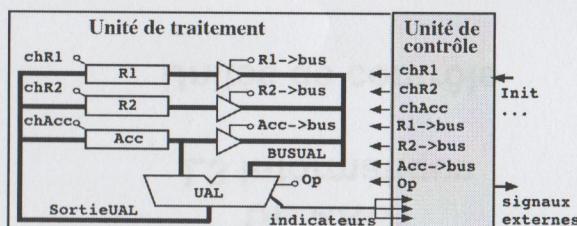
L3Info – ARC1

5

Unité de commande



- Commande le fonctionnement de l'UT
 - Sorties = commandes destinées à l'UT
 - Entrées = indicateurs/conditions calculés par l'UT



- Exemple : pour réaliser $acc := R1 + R2$

- Avec cette UT il faudra une séquence de 2 cycles :
 - 1 : R1->bus, op=transfert, chAcc (acc:=R1)
 - 2 : R2->bus, op=add, chAcc (acc:=acc+R2)

L3Info – ARC1

7

Notion de bus



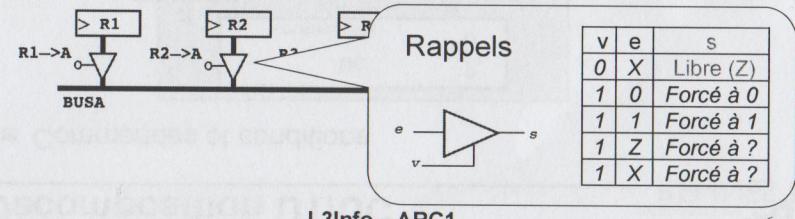
- Types d'interconnexions

Connexions simples	une seule source, plusieurs destinataires
Bus	plusieurs sources, plusieurs destinataires

- Réalisation modulaire des bus : sorties 3 états

- libre : non connectée à une source de tension
- forcé : connectée à une source de tension, 0 ou 1

- Bus réalisé au moyen d'adaptateurs trois états



L3Info – ARC1

6

Plan du chapitre



1. Décomposition d'un calculateur
 1. Principes généraux
 2. Méthode de décomposition
 3. Illustration sur un exemple
2. Réalisation des unités de contrôle
 1. Structure générale d'une unité de contrôle
 2. UC à bascules
 3. UC à compteur chargeable
 4. UC micro-programmée

L3Info – ARC1

8

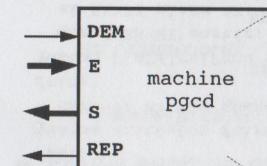
Méthode de décomposition

- ① Rédiger l'algorithme du fonctionnement
 - Exhiber le maximum d'actions parallèles/concurrentes
- ② Définir le schéma de l'Unité de Traitement
 - Variables de l'algorithme → Registres
 - Opérations → Opérateur(s)
 - Affectations → Chemins de données

Pour l'UT, choix entre une structure +/- spécialisée (performante) ou une structure plus générale (adaptable à d'autres algorithmes)
- ③ Dessiner le diagramme des états de l'Unité de Contrôle
 - Consiste à traduire l'algorithme en un diagramme d'états
 - Etat = paquet d'instructions de l'algorithme que l'on peut exécuter en même temps (i.e au même cycle)
 - Généralement mis en œuvre comme une machine de Moore

Un exemple : calcul des pgcd

- Machine à calculer le pgcd de 2 nombres



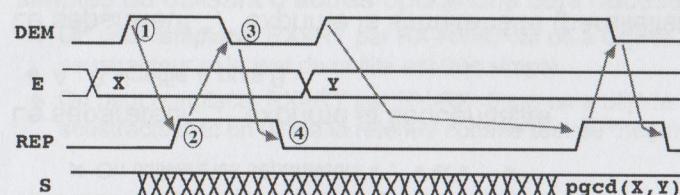
```

RX:= LireEntree();
RY:= LireEntree();
jusque FinCalculPgcd faire
    quand RX=RYS sortir,
    si RX>RY alors
        RX:= RX-RY
    sinon
        RY:= RY-RX
    finsi
fait;
    
```

- Opérandes X et Y fournis successivement sur l'entrée E
 - Il faut donc définir un protocole de communication pour X et Y

Un exemple : calcul des pgcd

- Protocole de demande-réponse entrelacées (*hand-shake*) :



- ① l'utilisateur envoie x et le signale par $DEM=1$
- ② la machine prend en compte x et le signale par $REP=1$
- ③ l'utilisateur a vu que la machine a pris en compte x et signale par $DEM=0$
- ④ la machine a vu que l'utilisateur avait vu que la machine avait pris en compte x et signale par $REP=0$

Algorithme de fonctionnement

```

faire
    jusque MontéeDem faire
        REP:=0, RX:=E, quand DEM=1 sortir
        fait;
    jusque RetombéeDem faire
        REP:=1, quand DEM=0 sortir
        fait;
    jusque MontéeDem faire
        REP:=0, RY:=E, quand DEM=1 sortir
        fait;
    jusque FinCalculPgcd faire
        quand RX=RYS sortir,
        si RX>RY alors RX:= RX-RY
        sinon RY:= RY-RX
        finsi
        fait;
    jusque RetombéeDem faire
        S:=RX, REP:=1, quand DEM=0 sortir
        fait
    
```

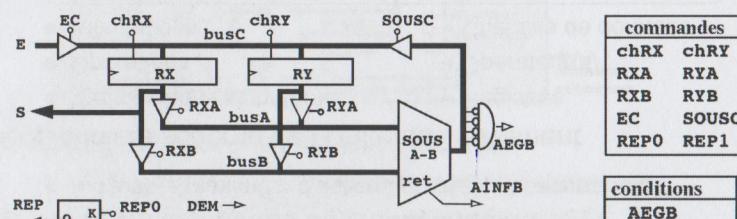
- Parallélisme → réaliser les traitements plus rapidement
 - Il faut faire apparaître du "parallélisme" dans l'algorithme.
- Il existe de nombreuses formes de parallélisme
 - Ici on se restreindra à un "parallélisme synchrone".
 - ✗ Calculs parallèles = calculs réalisés au même cycle horloge.
 - Il faut introduire des notations pour représenter ce parallélisme
 - ✗ On utilisera les séparateurs « ; » et « , »
- Le séparateur ";" exprime la séquentialité
 - A ; B signifie A puis B
- Le séparateur "," exprime la simultanéité (parallélisme)
 - A , B signifie A et B simultanément

- Intérêts : performance et moins d'états dans le contrôle
 - durée(A ; B) = durée(A) + durée(B)
 - durée(A , B) = max(durée(A),durée(B))
- Attention : sémantique différente, exemple :

<ul style="list-style-type: none"> ● X:=12 , Y:=56; ● X:=Y , Y:=X; ● X:=Y ; Y:=X; 	<ul style="list-style-type: none"> → X vaut 12 et Y vaut 56 → X vaut 56 et Y vaut 12 → X et Y valent 56
--	--

2. Conception de l'unité de traitement

- Mémoires : registres RX et RY, bascule REP
- Opérateurs : soustraction et comparaison
 - un soustracteur + test de nullité du résultat
- Chemins de données :
 - busA et busB pour réaliser X-Y ou Y-X
 - busC : valeur en entrée de RX et RY



Comment en est-on arrivé à ça plutôt qu'à autre chose ?

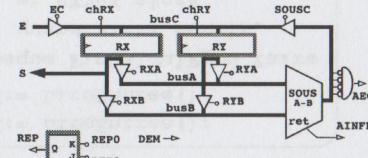
2. Comment identifier les registres ?

- On regarde les affectations de variables (opérateur :=)

```

faire
    jusque MontéeDem faire
        REP:=0, RX:=E, quand DEM=1 sortir
    fait;
    jusque RetombéeDem faire
        REP:=1, quand DEM=0 sortir
    fait;
    jusque MontéeDem faire
        REP:=0, RY:=E, quand DEM=1 sortir
    fait;
    jusque FinCalculPgcd faire
        quand RX=RY sortir,
        si RX>RY alors RX:= RX-RY
        sinon RY:= RY-RX
    finsi
    fait;
    jusque RetombéeDem faire
        S=RX, REP:=1, quand DEM=0 sortir
    fait
    fait

```



2. Comment trouver les opérateurs

■ On regarde les occurrences d'opérations

```

faire
jusque MontéeDem faire
    REP:=0, RX:=E, quand DEM=1 sortir
    fait;
jusque RetombéeDem faire
    REP:=1, quand DEM=0 sortir
    fait;
jusque MontéeDem faire
    REP:=0, RY:=E, quand DEM=1 sortir
    fait;
jusque FinCalculPgcd faire
    quand RX=RY sortir,
        si RX>RY alors RX:= RX-RY
        sinon RY:= RY-RX
    finsi
    * soustraction 1
test de supériorité
    * soustraction 2
finsi
    * soustraction 2
fais;
jusque RetombéeDem faire
    S=RX, REP:=1, quand DEM=0 sortir
    fait
fais

```

L3Info – ARC1

17

2. Comment trouver les opérateurs

■ Sur l'exemple on a inventorié les opérations

```

test d'égalité
jusque FinCalculPgcd faire
quand RX=RY sortir,
si RX>RY alors RX:= RX-RY
sinon RY:= RY-RX
finsi
    * soustraction 1
    * soustraction 2
test de supériorité
    * soustraction 2

```

- 2 soustractions, 1 test d'égalité et 1 test de supériorité

■ Faut-il avoir un opérateur pour chaque opération ?

- Critère systématique : les opérations de même type sont elles utilisées simultanément ? sinon, un seul opérateur suffit.

■ Exemple

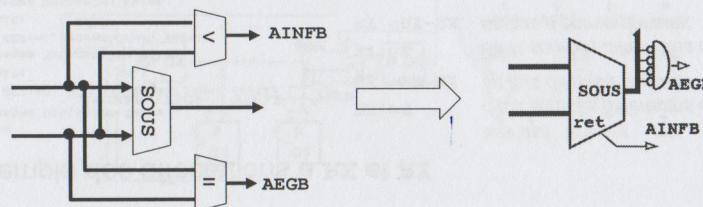
- *soustraction₁* et *soustraction₂* ne sont pas utilisées en même temps (branches d'une conditionnelle) : un seul soustracteur *sous* suffit

L3Info – ARC1

18

2. Comment trouver les opérateurs

- Peut-on remplacer des opérations par des opérations plus simples ou utilisant d'autres opérations déjà nécessaires ?
 - On peut remplacer RX=RY par RX-RY=0 car on a déjà le soustracteur et le test de nullité est très simple.
 - On peut remplacer RX>RY par RY-RX<0 car on a déjà le soustracteur et on utilise la retenue comme test de "négativité"



L3Info – ARC1

19

2. Dériver les interconnexions (1/3)

■ Les interconnexions se déduisent

- Des parties droites des affectations de variables (opérateur :=)
- Des opérandes utilisées dans les opérateurs ou sorties

Exemple de la variable REP

```

jusque MontéeDem faire
    REP:=0, RX:=E, qd DEM=1 sortir
    fait;
jusque RetombéeDem faire
    REP:=1, qd DEM=0 sortir
    fait;
jusque MontéeDem faire
    REP:=0, RY:=E, qd DEM=1 sortir
    fait;
jusque FinCalculPgcd faire
    quand RX=RY sortir,
        si RX>RY alors RX:= RX-RY
        sinon RY:= RY-RX
    finsi
    * soustraction 1
    * soustraction 2
finsi
    * soustraction 2

```

deux sources différentes pour REP (0 ou 1)

version "systématique" / version "ad-hoc"

REP = ChRep REP0

un bus avec 0 et 1 comme sources

usage d'une bascule JK

L3Info – ARC1

20

2. Dériver les interconnexions (2/3)

Exemple des affectations à RX et RY

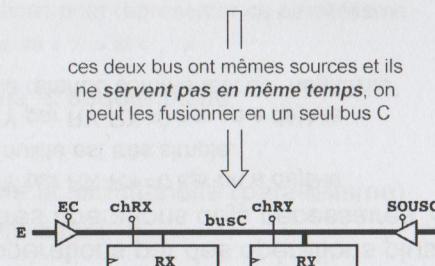
```

faire
  jusque MontéeDem faire
    REP:=0, RX:=E, quand DEM=1 sortir
  fait;
  jusque RetombéeDem faire
    REP:=1, quand DEM=0 sortir
  fait;
  jusque MontéeDem faire
    REP:=0, RX:=E, quand DEM=1 sortir
  fait;
  jusque FinCalculPgcd faire
    quand RX=RY sortir,
      si RX>RY alors RX:= RX-RY
      sinon RY:= RY-RX
    finsi
  fait;
  jusque RetombéeDem faire
    S=RX, REP:=1, quand DEM=0 sortir
  fait
fait

```

RX := E deux sources différentes donc un bus d'entrée dans RX
 RX := RX - RY deux sources différentes donc un bus d'entrée dans RY
 RY := E deux sources différentes donc un bus d'entrée dans RY
 RY := RY - RX deux sources différentes donc un bus d'entrée dans RY

ces deux bus ont mêmes sources et ils ne servent pas en même temps, on peut les fusionner en un seul bus C



2. Dériver les interconnexions (3/3)

Exemple des opérandes en entrées d'opérateurs

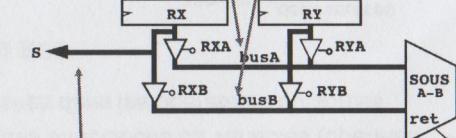
```

faire
  ...
  jusque FinCalculPgcd faire
    quand RX=RY sortir,
      si RX>RY alors RX:= RX-RY
      sinon RY:= RY-RX
    finsi
  fait;
  jusque RetombéeDem faire
    S=RX, REP:=1, quand DEM=0 sortir
  fait
fait

```

deux sources != différentes pour l'opérande A du soustracteur
 RX - RY deux sources != différentes pour l'opérande B du soustracteur

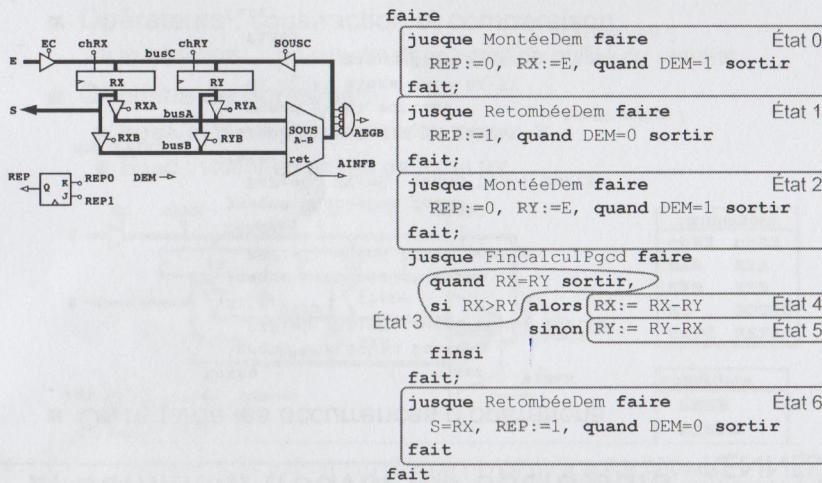
→ ajout de busA → ajout de busB



S=RX une seule source donc connexion directe de RX sur S, pas besoin de bus

3. Diagramme des états de UC

■ Etat = paquet d'instructions réalisables simultanément



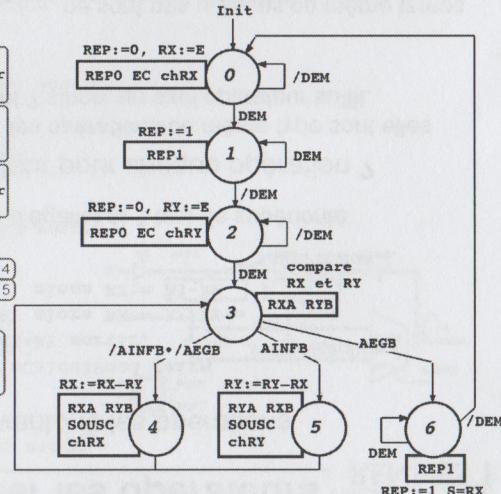
3. Diagramme des états de UC

■ Il faut bien sûr traduire les instructions en commandes :

```

faire
  jusque MontéeDem faire      État 0
    REP:=0, RX:=E, quand DEM=1 sortir
  fait;
  jusque RetombéeDem faire      État 1
    REP:=1, quand DEM=0 sortir
  fait;
  jusque MontéeDem faire      État 2
    REP:=0, RY:=E, quand DEM=1 sortir
  fait;
  jusque FinCalculPgcd faire
    quand RX=RY sortir,
      si RX>RY alors RX:= RX-RY      État 4
      sinon RY:= RY-RX      État 5
    finsi
  fait;
  jusque RetombéeDem faire      État 6
    S=RX, REP:=1, quand DEM=0 sortir
  fait
fait

```



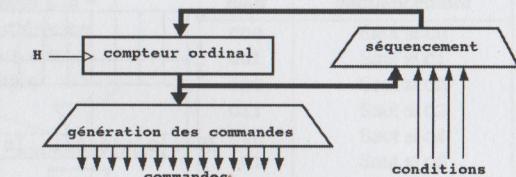
Plan du chapitre

1. Décomposition d'un calculateur
 1. Principes généraux
 2. Méthode de décomposition
 3. Exemple
2. Réalisation des unités de contrôle
 1. Structure générale d'une unité de contrôle
 2. UC à bascules
 3. UC à compteur chargeable
 4. UC micro-programmée

Structure générale d'une UC

- UC = système séquentiel synchrone standard
 - On y associe cependant une terminologie particulière :

Etat	Compteur ordinal
Fonction de sortie	Génération de commandes
Fonction de transition	Séquencement

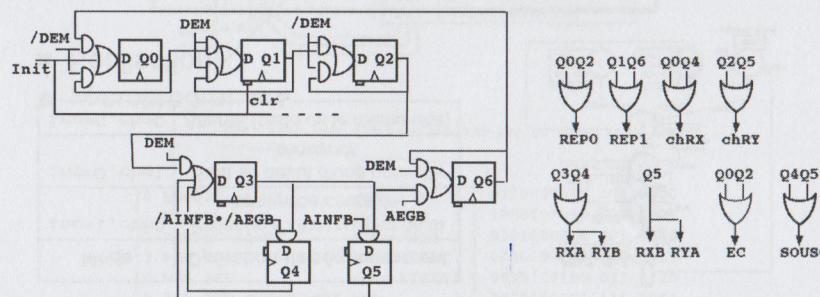


- Il existe trois principales techniques de mise en œuvre

- ① UC à bascules
- ② UC à compteur chargeable
- ③ UC microprogrammée

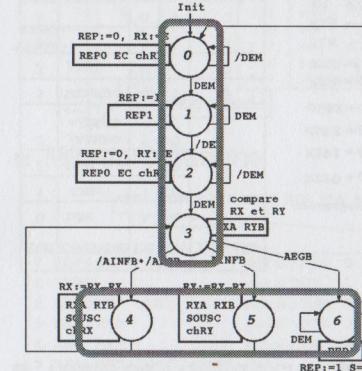
Unité de contrôle à bascules

- Déjà vu au chapitre précédent !!
 - Codage de l'état : à priori sans importance
 - Mais on choisit souvent un codage "machine à jeton"
 - ✗ structure du circuit similaire au diagramme des états



UC à compteur chargeable

- Observation : la plupart des diagrammes d'état contient de nombreuses (mais courtes) séquences d'états



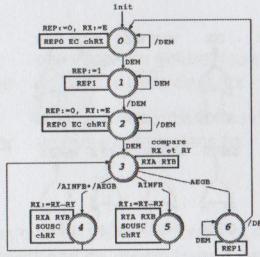
Le séquencement de ces 4 états pourrait être réalisé à l'aide d'un simple compteur binaire sur 2 bits !

Il resterait à gérer le cas des états 4,5 et 6

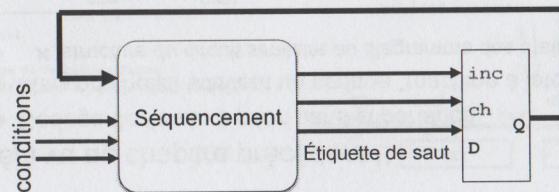
UC à compteur chargeable

- Principe : le séquencement est assuré par un compteur

Mode	Opération de séquencement
inc=1, ch=0	Passage à l'état suivant de la séquence courante
inc=0, ch=1	Saut au début d'une nouvelle séquence
inc=0, ch=0	Attente (reste ds le même état)



Réalisation



L3Info – ARC1

29

Plan du chapitre

1. Décomposition d'un calculateur
 1. Principes généraux
 2. Méthode de décomposition
 3. Exemple
2. Réalisation des unités de contrôle
 1. Structure générale d'une unité de contrôle
 2. UC à bascules
 3. UC à compteur chargeable
 4. UC micro-programmée

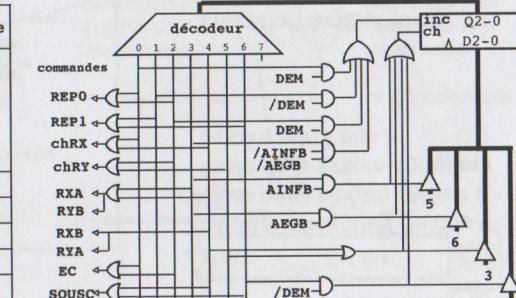
L3Info – ARC1

31

UC à compteur chargeable

- Table des cas de chargement et d'incrémentation

Etat	Condition	Inc	Ch	Etiquette
0	DEM	1	0	
1	/DEM	1	0	
2	DEM	1	0	
3	/AINFB • /AEGB	1	0	
3	AINFB	0	1	5
3	AEGB	0	1	6
4		0	1	3
5		0	1	3
6	/DEM	0	1	0

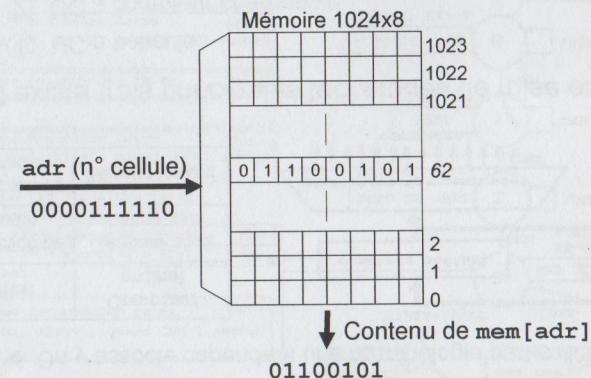


L3Info – ARC1

30

Notion de mémoire adressable

- Mémoire adressable = version matérielle du type tableau
 - Ici mémoire combinatoire, mais il existe des versions synchrones



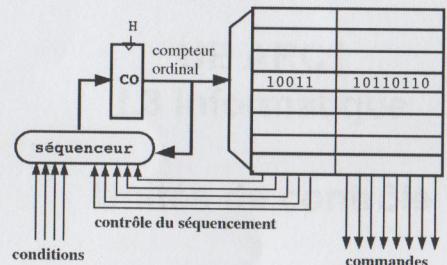
- Pas de panique : sera étudié en détail au chapitre suivant

L3Info – ARC1

32

UC microprogrammée

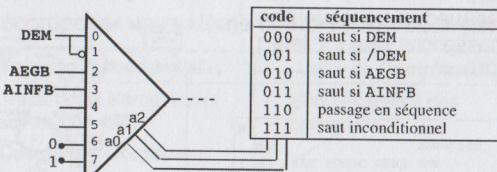
- Séquencement et génération des commandes par une mémoire adressable adressée par un compteur ordinal



- Séquencement assuré par un composant séquenceur
 - Le composant séquence est commandé par des codes
 - Ces codes sont issus du champ contrôle du séquencement

Exemple : pour la machine pgcd

Séquenceur

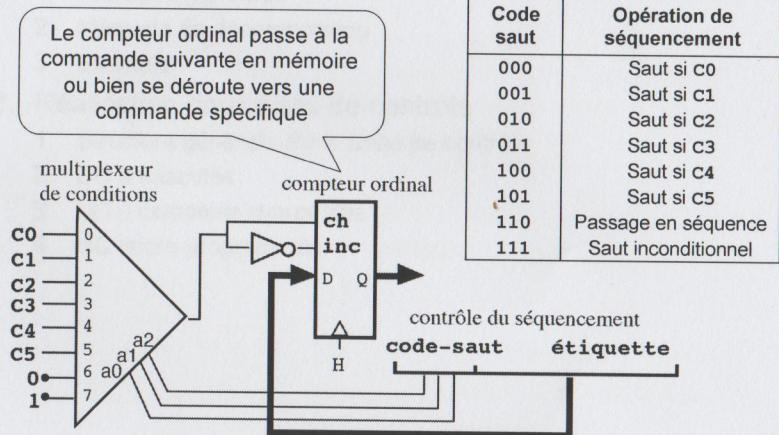


Microprogramme

	REP1 REP0 EC chRX	chRY RYA RXA RXB RYB EC SOUSC	Code	adr.
0	REP0 EC chRX	saut si /DEM	0	0110000010 001 0000
1	REP1	saut si DEM	1	1000000000 000 0001
2	REP0 EC chRY	saut si /DEM	2	0101000010 001 0010
3	RXA RYB	saut si AEGB	7	0000100100 010 0111
4	RXA RYB	saut si AINFBA	6	0000100100 011 0110
5	RXA RYB SOUSC chRX	saut	3	0010100101 111 0011
6	RXB RYA SOUSC chRY	saut	3	0001011001 111 0011
7	REP1	saut si DEM	7	1000000000 000 0111
8		saut	0	1000000000 111 0000

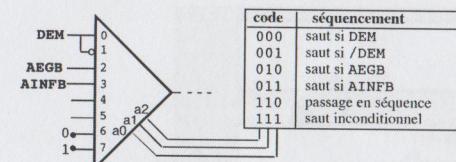
Séquenceur

- Séquenceur simple : sauts conditionnels



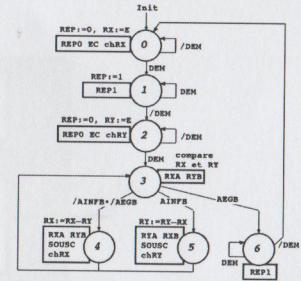
Exemple : pour la machine pgcd

Séquenceur



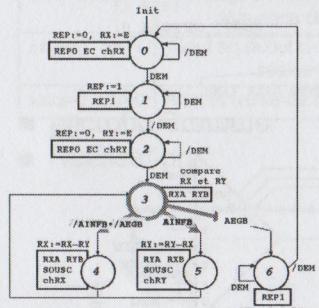
Microprogramme

	REP1 REP0 EC chRX	chRY RYA RXA RXB RYB EC SOUSC	Code	adr.
0	REP0 EC chRX	saut si /DEM	0	0110000010 001 0000
1	REP1	saut si DEM	1	1000000000 000 0001
2	REP0 EC chRY	saut si /DEM	2	0101000010 001 0010
3	RXA RYB	saut si AEGB	7	0000100100 010 0111
4	RXA RYB	saut si AINFBA	6	0000100100 011 0110
5	RXA RYB SOUSC chRX	saut	3	0010100101 111 0011
6	RXB RYA SOUSC chRY	saut	3	0001011001 111 0011
7	REP1	saut si DEM	7	1000000000 000 0111
8		saut	0	1000000000 111 0000



■ Remarque :

- La manque d'opérations de séquencement oblige à introduire des états supplémentaires (dits parasites)



	REP1	REP0	chRX	chRY	RXA	RYB	EC	SOUSC	Code	adr.
0	REP0	EC	chRX		saut si /DEM	0	01100000010	001	0000	
1	REP1				saut si DEM	1	10000000000	000	0001	
2	REP0	EC	chRY		saut si /DEM	2	01010000010	001	0010	
3	RXA	RYB			saut si AEGB	7	00001001000	010	0111	
4	RXA	RYB			saut si AINFB	6	00001001000	011	0110	
5	RYA	RYB	SOUSC	chRX	saut	3	00101001011	111	0011	
6	RYA	RYB	SOUSC	chRY	saut	3	00001011001	111	0011	
7	REP1				saut si DEM	7	10000000000	000	0111	
8					saut	0	10000000000	111	0000	

L'état 3 à trois successeurs alors que ce séquenceur n'en permet que deux, ce qui oblige à décomposer l'état 3 en deux lignes de microprogramme (lignes 3 et 4)