L3 EEA

Architecture HAE504E

virazel@lirmm.fr

HAE504E

Architecture

Arnaud VIRAZEL
virazel@lirmm.fr

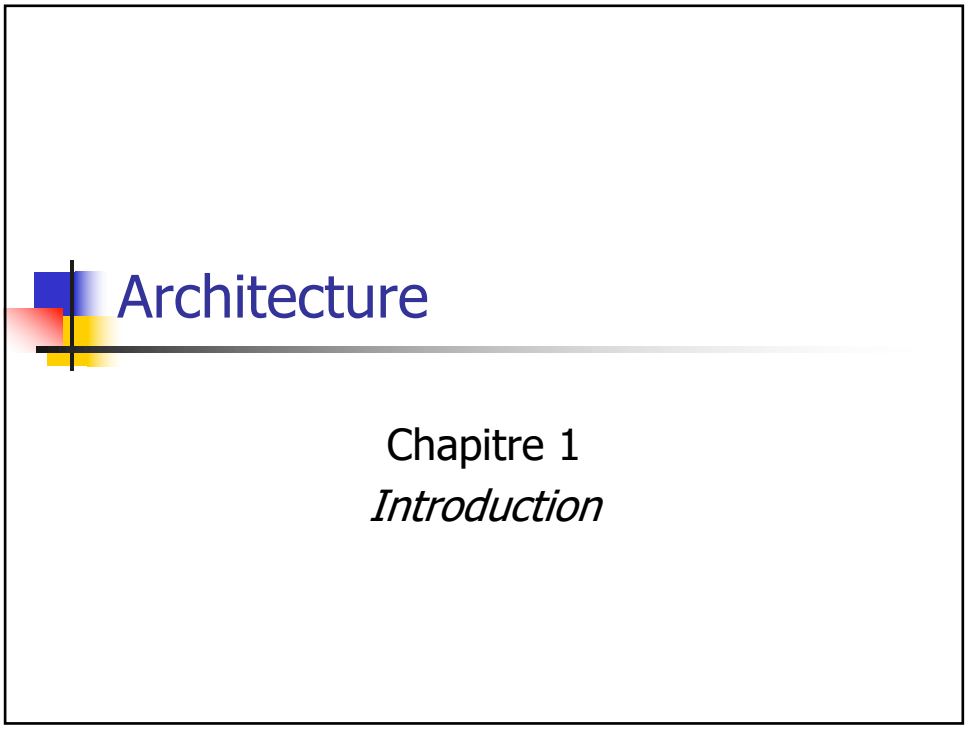
1

Plan du Cours

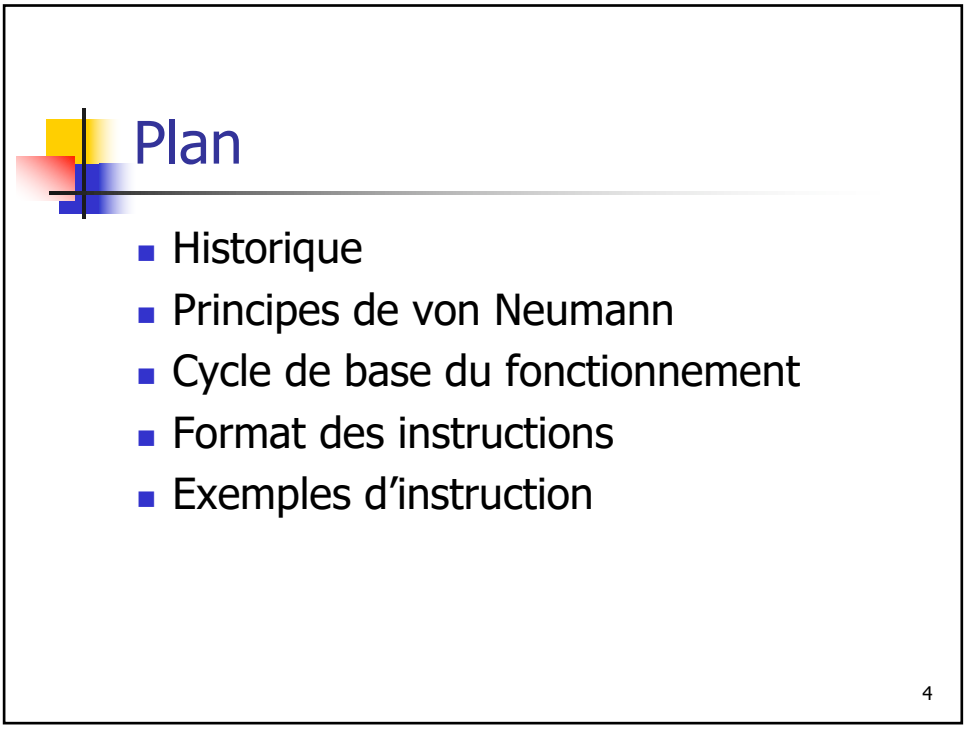
- Chapitre 1 : Introduction
- Chapitre 2 : Unité de Traitement
- Chapitre 3 : Décodage des Instructions
- Chapitre 4 : Interruptions
- Chapitre 5 : Entrées/Sorties
- Chapitre 6 : Circuits d'Interfaces

2

2


Architecture
Chapitre 1
Introduction

3


Plan
Historique
Principes de von Neumann
Cycle de base du fonctionnement
Format des instructions
Exemples d'instruction
4

4

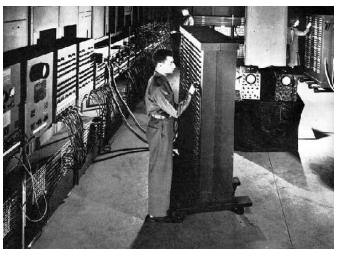
John von Neumann

- Mathématicien et physicien américano-hongrois
- Il a établi les bases d'un calculateur électronique et construit l'une des premières machines en 1945.

5

5

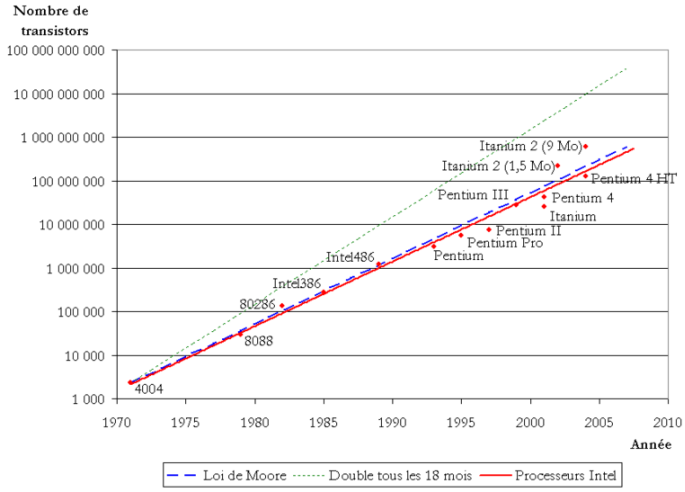
Evolution Technologique



6

6

Loi de Moore



7

7

Principes de von Neumann

- L'architecture de von Neumann décompose l'ordinateur en 4 parties distinctes :
  - L'unité arithmétique et logique (UAL ou ALU en anglais) ou unité de traitement :
    - son rôle est d'effectuer les opérations
  - L'unité de contrôle :
    - chargée du séquençage des opérations

8

8

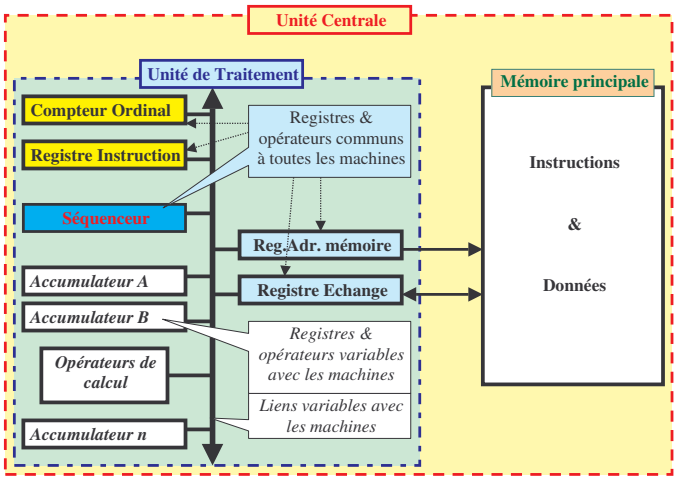
Principes de von Neumann

- La mémoire :
  - qui contient à la fois les données et le programme qui dira à l'unité de contrôle quels calculs faire sur ces données. La mémoire se divise entre mémoire volatile (programmes et données en cours de fonctionnement) et mémoire permanente (programmes et données de base de la machine).
- Les dispositifs d'entrée-sortie :
  - qui permettent de communiquer avec le monde extérieur.

9

9

Architecture de von Neumann

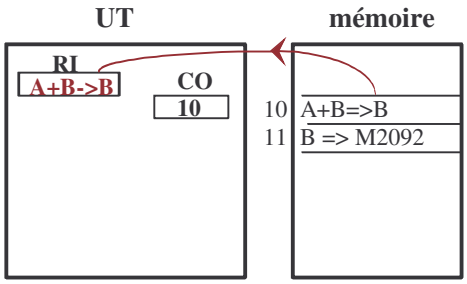


10

10

Cycle de base du Fonctionnement

- Recherche de l'instruction ou cycle Fetch

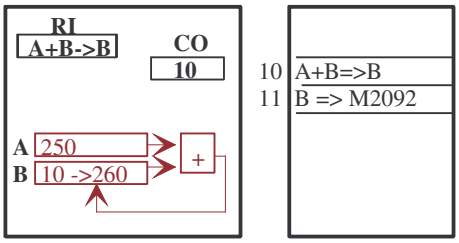


11

11

Cycle de base du Fonctionnement

- Exécution de l'instruction

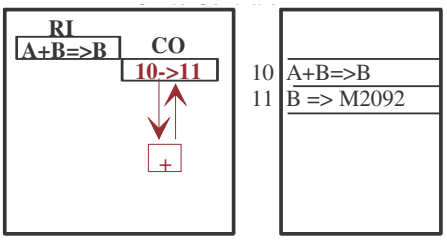


12

12

Cycle de base du Fonctionnement

- Progression du compteur ordinal pour pointer l'instruction suivante



13

13

Format des Instructions

- Chaque instruction est représentée en mémoire par une combinaison de bits (un ou plusieurs mots selon sa complexité)
- La combinaison se compose d'une partie Code Opération et d'autant de (champs) de bits qu'il faut pour définir les autres paramètres de l'instruction.

COP / MA / RA

14

14

Le Code Opération - COP

- Il définit le(s) objectif(s) de l'instruction
- Exemples :
  - LOAD A – charger une valeur dans le registre A
  - ADD B – Additionner une valeur au registre B et placer le résultats dans B.

15

15

Le Mode d'Adressage - MA

- Il définit le type d'accès à la mémoire
- Exemples :
  - **Immédiat** – passage par valeur

    COP / MA / RA l'Opérande = RA
  - **Direct** – passage par adresse (pointeur en C)

    COP / MA / RA AdresseOpérande = RA

16

16

Le Mode d'Adressage - MA

- **Indirect** – passage par adresse d'adresse (pointeur de pointeur en C)

COP / MA / RA

AdresseOpérande = (RA)
( ) *signifie CONTENU de la mémoire d'adresse RA*

- **Relatif** – passage par adresse relatif à la position de l'instruction

α : COP / MA / RA

AdresseOpérande = α +RA

17

17

Le Mode d'Adressage - MA

- **Indexé** – passage par adresse (adressage des tableaux en C)

COP / MA / RA

AdresseOpérande = RIndex+RA

18

18

La Référence Adresse - RA

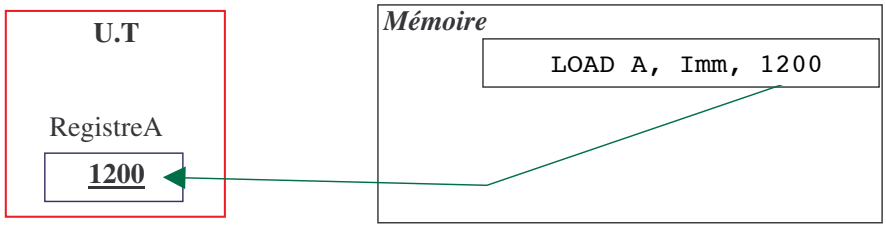
- Il définit la valeur ou adresse (en fonction du Mode d'Adressage à manipuler avec l'instruction
- Exemples :
  - LOAD A, Immédiat, 12 – 12 est une valeur qui doit être chargée dans le registre A
  - LOAD B, Direct, 153 – 153 est une adresse à laquelle se trouve la valeur à charger dans le registre B.

19

19

Exemples

- Adressage **Immédiat**

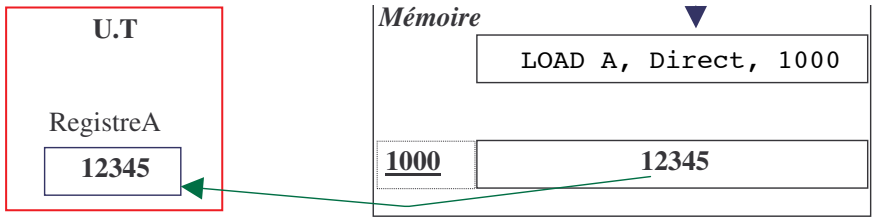


20

20

Exemples

- Adressage **Direct**

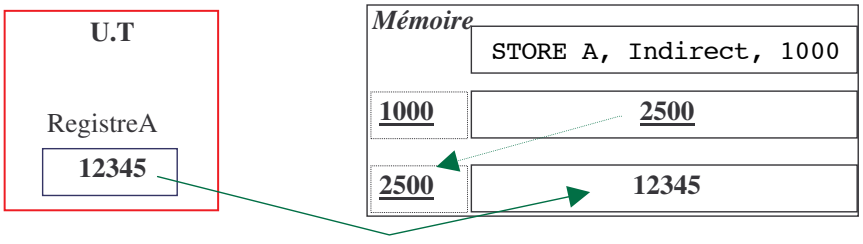


21

21

Exemples

- Adressage **Indirect**

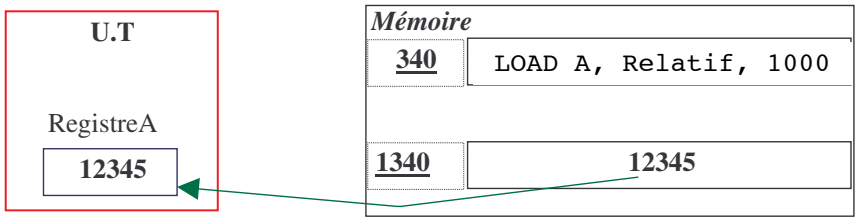


22

22

Exemples

- Adressage **Relatif**

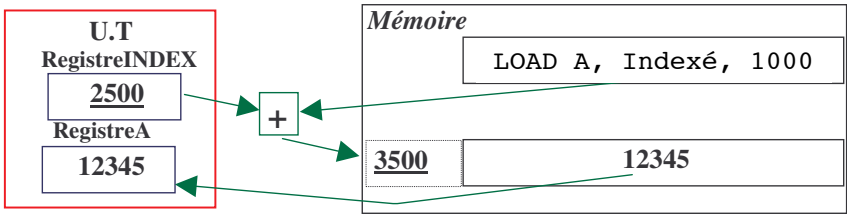


23

23

Exemples

- Adressage **Indexé**



24

24

Architecture

Chapitre 2

Unité de Traitement

1

Plan

- Primitives de construction
- Exemple de structure
- Séquencement des instructions
  - Cycle Fetch
  - Exemples d'instruction

2

2

Primitives de Construction

- Les registres
  - Il sont constitués d'un ensemble de bascules (D flip-flop) dont le comportement et le rôle sont homogènes
  - Chaque bascule mémorise un bit d'un mot
  - L'ensemble des bascules constitue le Registre
  - Le Registre contient le mot

3

3

Primitives de Construction

- Chargement du Registre
  - Sur chaque entrée doit être présent un bit du mot
  - Une impulsion d'écriture commune à toutes les bascules provoque la mémorisation simultanée de toutes les bascules, donc du mot

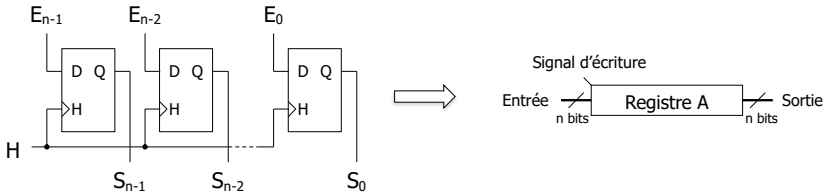
4

4

Primitives de Construction

- Lecture du Registre
  - Le contenu est présent en permanence sur la sortie

Mot d'entrée E sur n bits

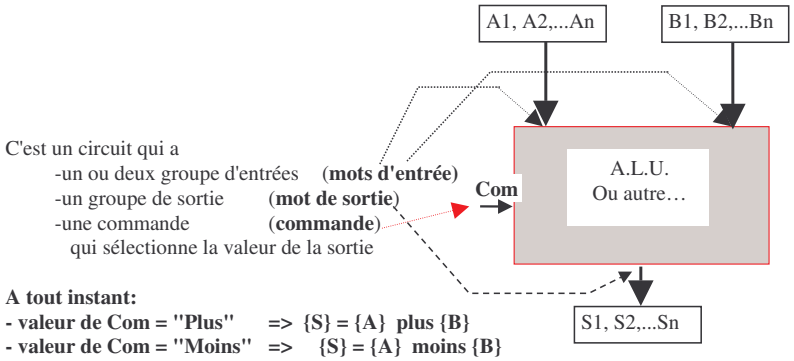


Mot de sortie S sur n bits

5

Primitives de Construction

- Unité Arithmétique et Logique (ALU)



C'est un circuit qui a
-un ou deux groupe d'entrées (**mots d'entrée**)
-un groupe de sortie (**mot de sortie**)
-une commande (**commande**)
qui sélectionne la valeur de la sortie

A tout instant:
- valeur de Com = "Plus" => {S} = {A} plus {B}
- valeur de Com = "Moins" => {S} = {A} moins {B}
- valeur de Com = "F1" => {S} = {A} F1 {B}
- valeur de Com = "Fn" => {S} = {A} F2 {B}

6

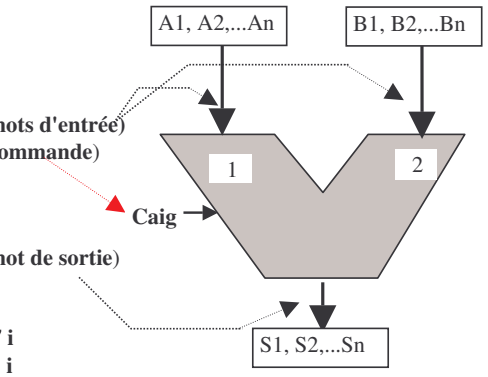
Primitives de Construction

- Multiplexage

C'est un circuit qui a
-plusieurs groupe d'entrées (**mots d'entrée**)
-une commande (**commande**)
qui sélectionne le mot d'entrée
appliqué en sortie

-un groupe de sortie (**mot de sortie**)

A tout instant:
- valeur de Caig = "1" => Si = Ai ∀ i
- valeur de Caig = "2" => Si = Bi ∀ i



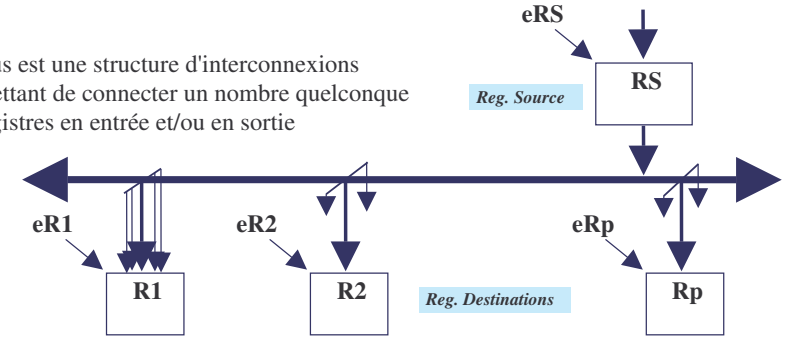
7

7

Primitives de Construction

- Bus - Principe

Un bus est une structure d'interconnexions permettant de connecter un nombre quelconque de registres en entrée et/ou en sortie



- Pour écrire dans un registre, **l'information doit être présente sur le Bus**, et une **impulsion d'écriture du registre doit être envoyée.**

8

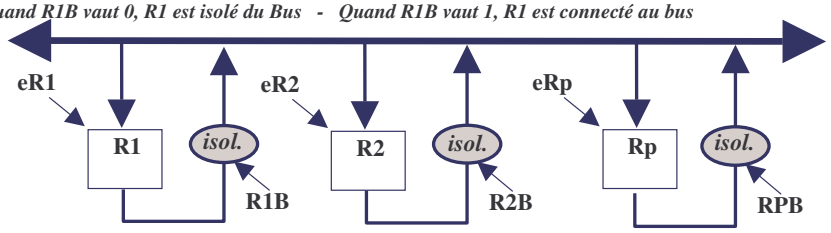
8

Primitives de Construction

- Bus bidirectionnel

Chaque registre est aussi connecté sur le Bus en sortie. Pour que toutes les valeurs ne se mélangent pas, un seul Registre à la fois doit être connecté sur le Bus. Il faut donc un **circuit d'isolement**.

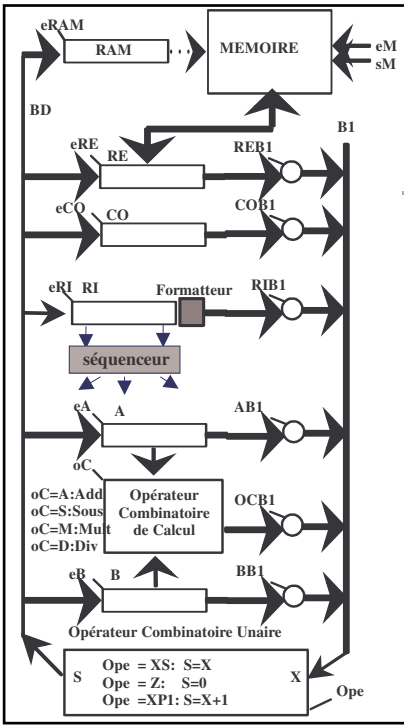
- Quand R1B vaut 0, R1 est isolé du Bus - Quand R1B vaut 1, R1 est connecté au bus



Conséquence: ***Dés qu'un signal RiB vaut 1, les autres valent obligatoirement 0***.

9

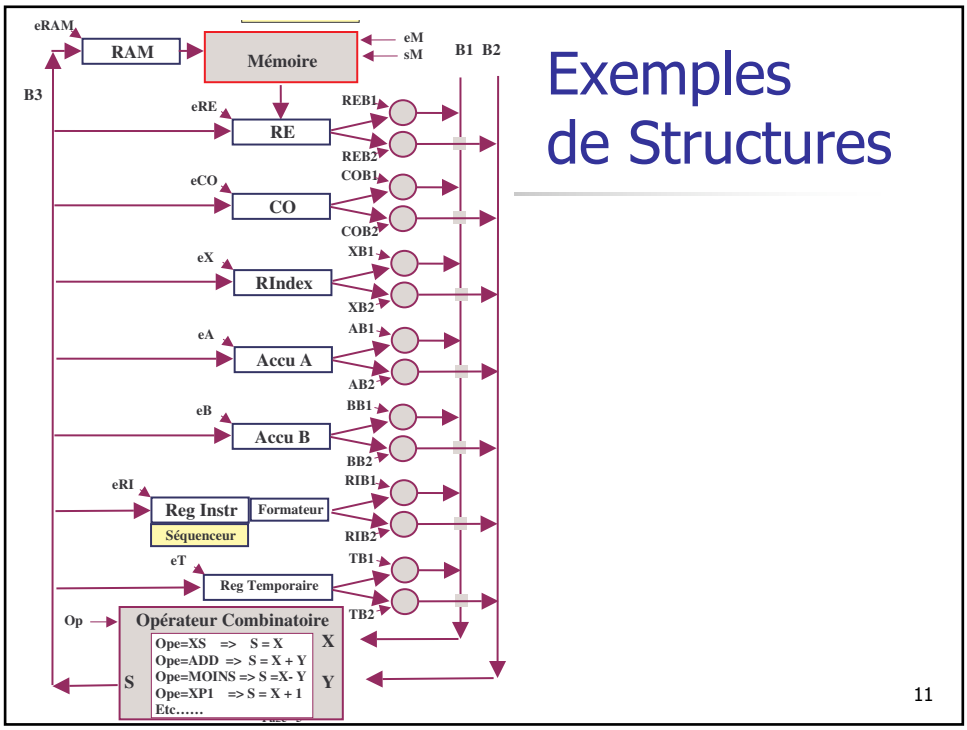
9



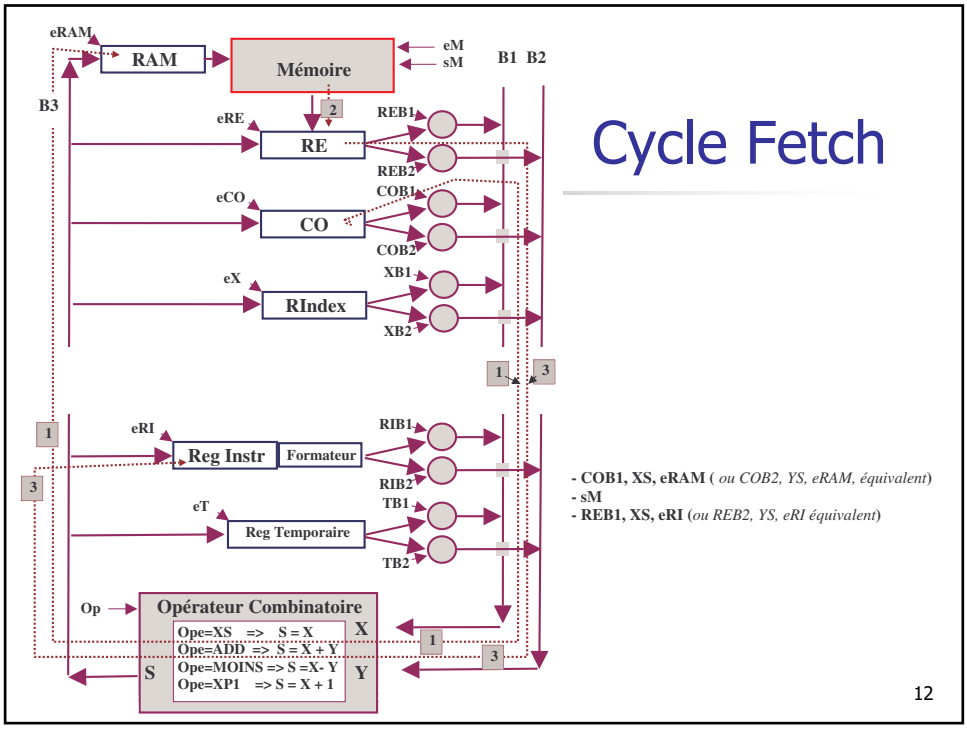
Exemples de Structures

10

10

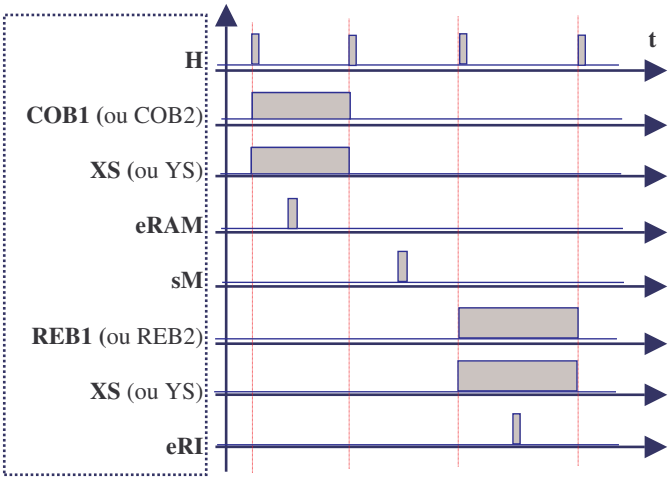

Exemples
de Structures
eRAM
RAM
Mémoire
eM
sM
B1 B2
B3
eRE
RE
REB1
REB2
eCO
CO
COB1
COB2
eX
RIndex
XB1
XB2
eA
Accu A
AB1
AB2
eB
Accu B
BB1
BB2
eRI
Reg Instr
Formateur
Séquenceur
RIB1
RIB2
eT
Reg Temporaire
TB1
TB2
Op
Opérateur Combinatoire
Ope=XS => S = X
Ope=ADD => S = X + Y
Ope=MOINS => S =X- Y
Ope=XP1 => S = X + 1
Etc……
X
Y
S
11

11


Cycle Fetch
eRAM
RAM
Mémoire
eM
sM
B1 B2
B3
2
eRE
RE
REB1
REB2
eCO
CO
COB1
COB2
eX
RIndex
XB1
XB2
1
3
eRI
Reg Instr
Formateur
RIB1
RIB2
eT
Reg Temporaire
TB1
TB2
Op
Opérateur Combinatoire
Ope=XS => S = X
Ope=ADD => S = X + Y
Ope=MOINS => S =X- Y
Ope=XP1 => S = X + 1
X
Y
S
- COB1, XS, eRAM (ou COB2, YS, eRAM, équivalent)
- sM
- REB1, XS, eRI (ou REB2, YS, eRI équivalent)
12

12

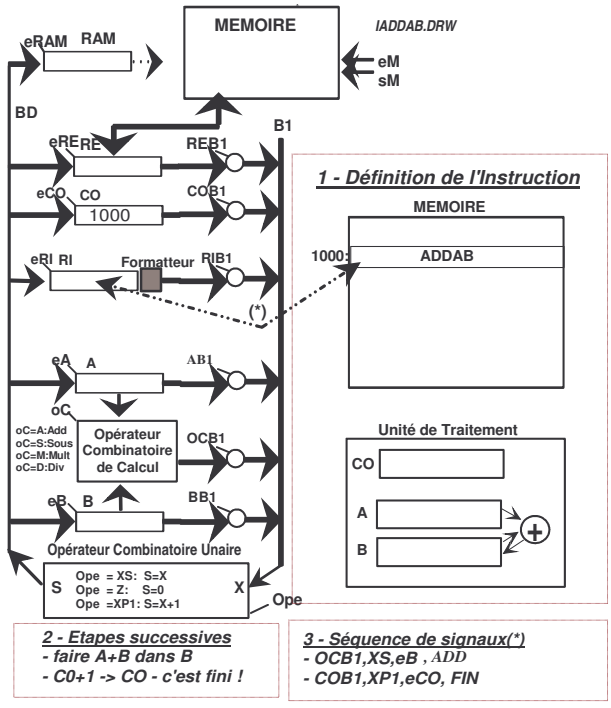
Cycle Fetch



13

13

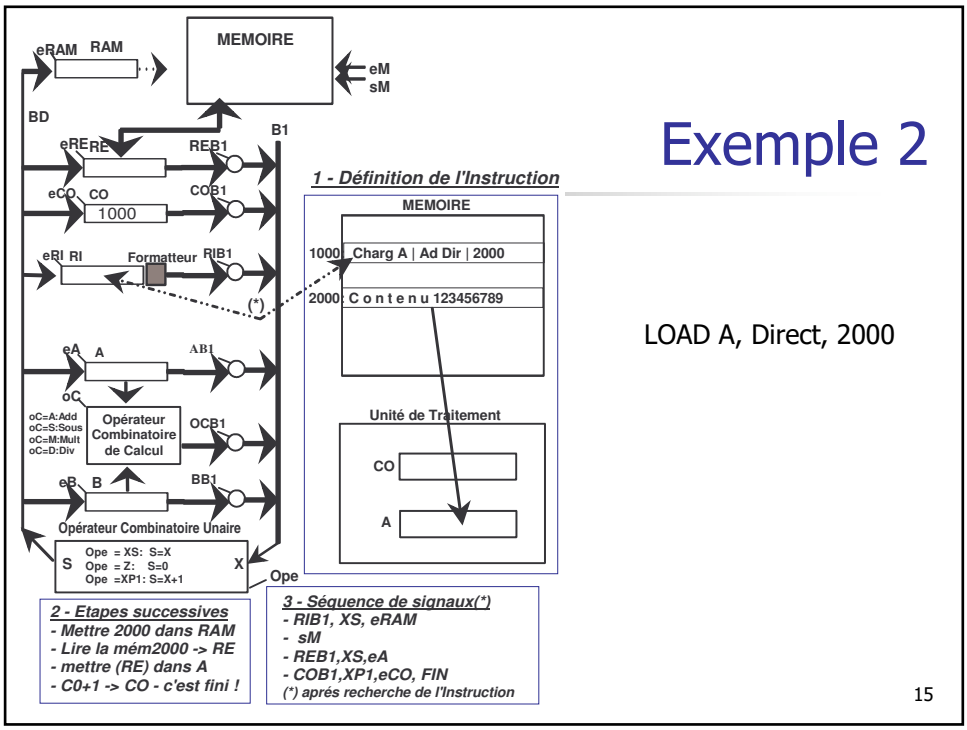
Exemple 1



ADD AB -> B = A+B

14

14

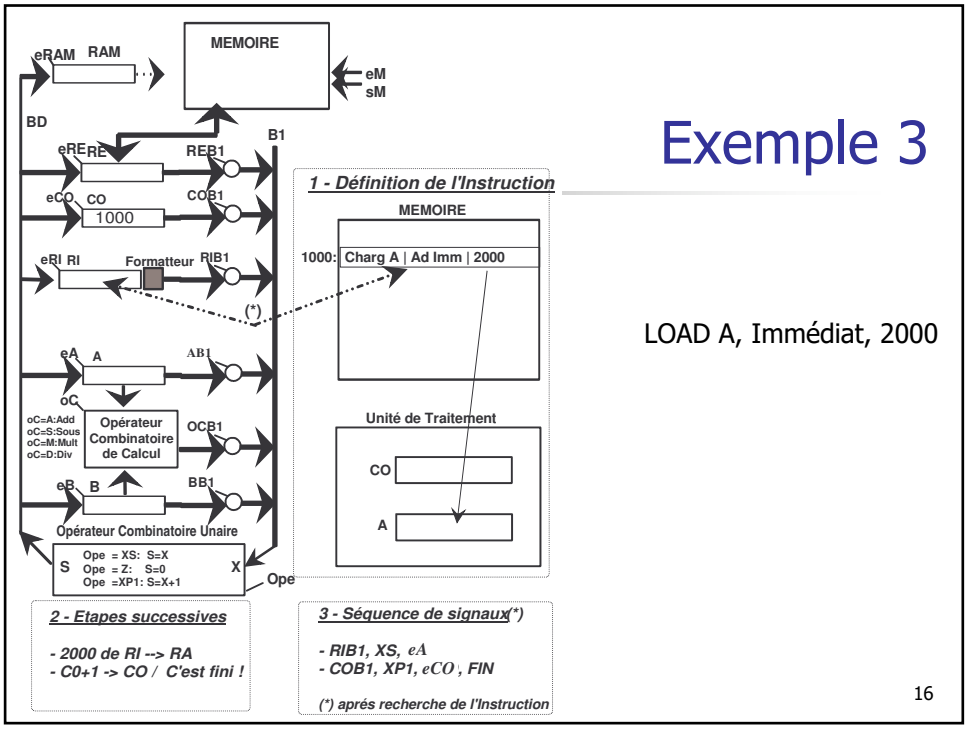


Exemple 2

LOAD A, Direct, 2000

15

15

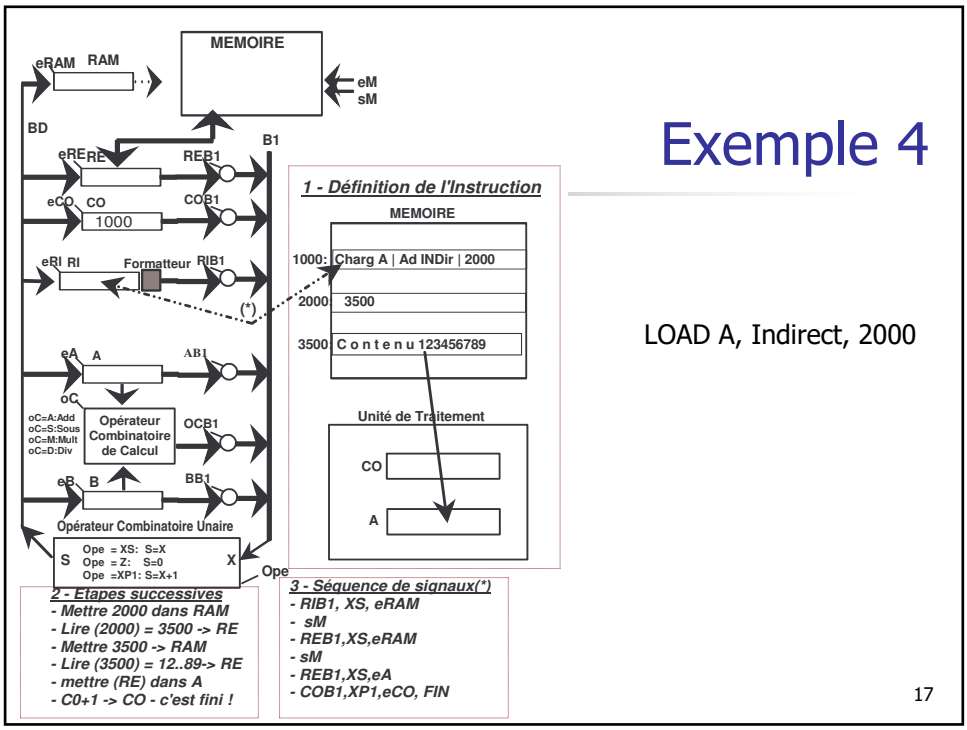


Exemple 3

LOAD A, Immédiat, 2000

16

16

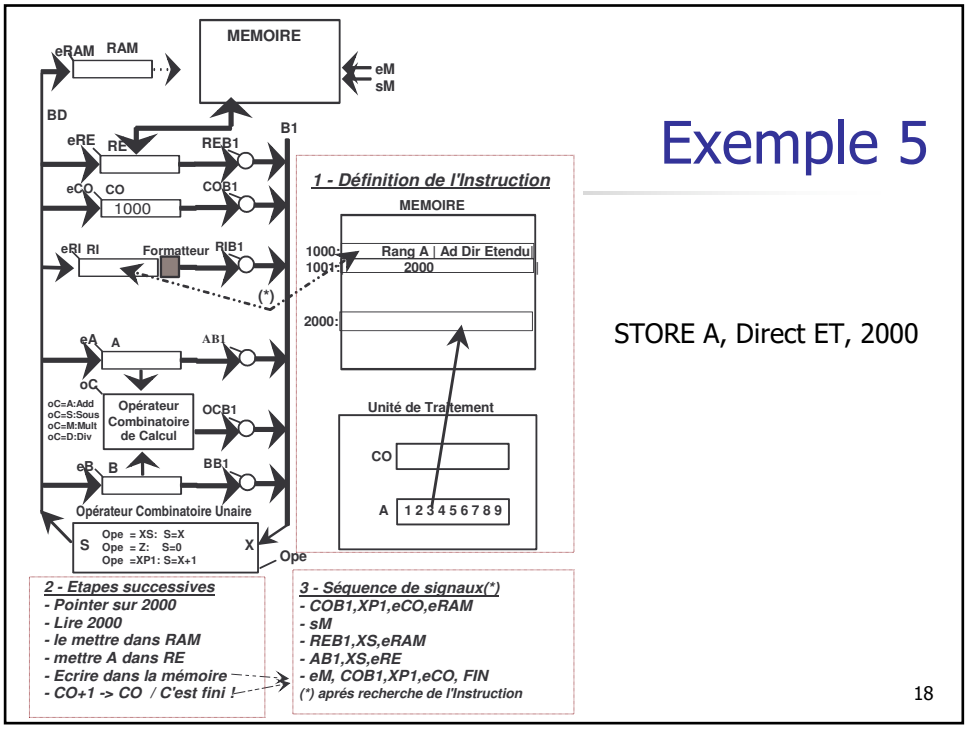


Exemple 4

LOAD A, Indirect, 2000

17

17

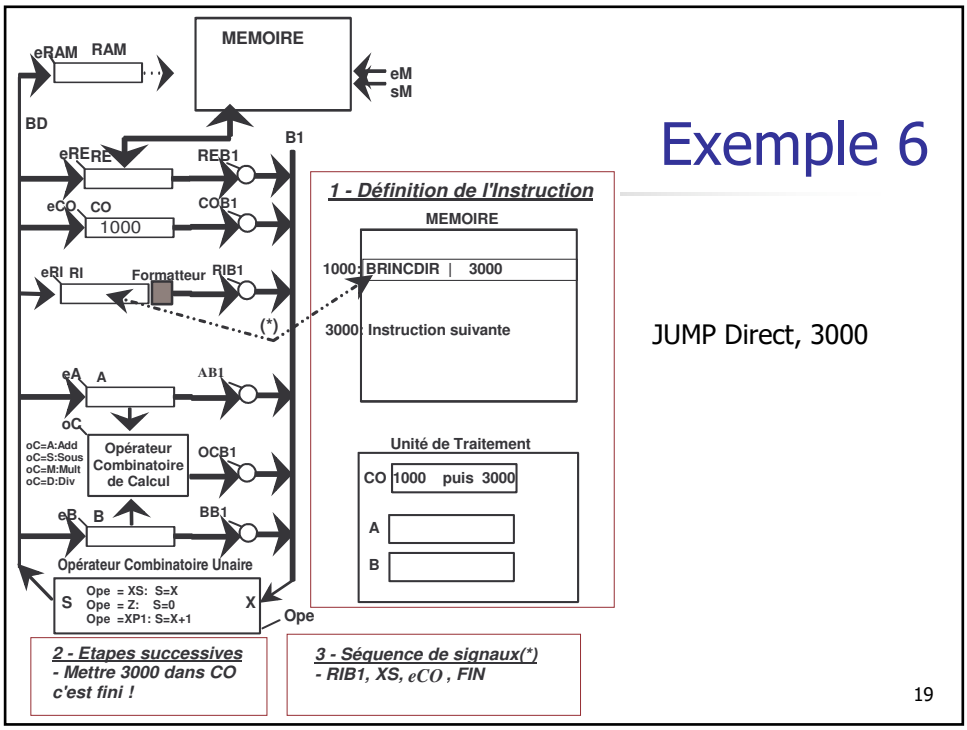


Exemple 5

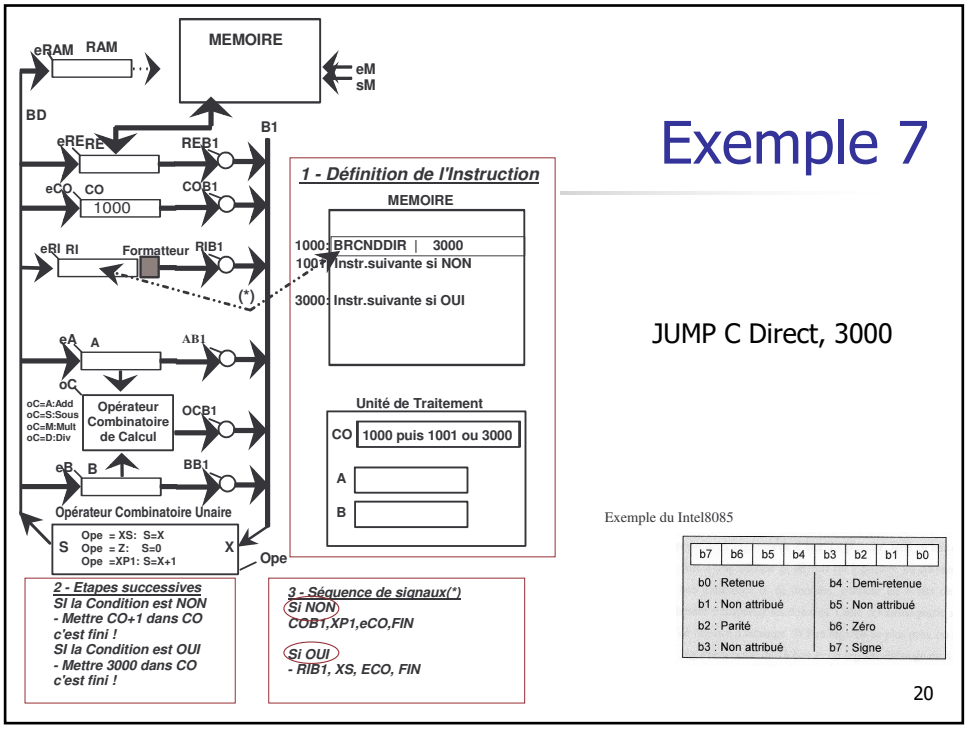
STORE A, Direct ET, 2000

18

18


MEMOIRE
eRAM RAM
eM
sM
BD
B1
eRE RE REB1
eCO CO COB1
1000
eRI RI Formatteur RIB1
(*)
eA A AB1
oC
oC=A:Add
oC=S:Sous
oC=M:Mult
oC=D:Div
Opérateur Combinatoire de Calcul
OCB1
eB B BB1
Opérateur Combinatoire Unaire
S
Ope = XS: S=X
Ope = Z: S=0
Ope =XP1: S=X+1
X
Ope
1 - Définition de l'Instruction
MEMOIRE
1000: BRINCDIR | 3000
3000: Instruction suivante
Unité de Traitement
CO 1000 puis 3000
A
B
2 - Etapes successives
- Mettre 3000 dans CO
c'est fini !
3 - Séquence de signaux(*)
- RIB1, XS, eCO , FIN
Exemple 6
JUMP Direct, 3000
19

19


MEMOIRE
eRAM RAM
eM
sM
BD
B1
eRE RE REB1
eCO CO COB1
1000
eRI RI Formatteur RIB1
(*)
eA A AB1
oC
oC=A:Add
oC=S:Sous
oC=M:Mult
oC=D:Div
Opérateur Combinatoire de Calcul
OCB1
eB B BB1
Opérateur Combinatoire Unaire
S
Ope = XS: S=X
Ope = Z: S=0
Ope =XP1: S=X+1
X
Ope
1 - Définition de l'Instruction
MEMOIRE
1000: BRCNDDIR | 3000
1001: Instr.suivante si NON
3000: Instr.suivante si OUI
Unité de Traitement
CO 1000 puis 1001 ou 3000
A
B
2 - Etapes successives
SI la Condition est NON
- Mettre CO+1 dans CO
c'est fini !
SI la Condition est OUI
- Mettre 3000 dans CO
c'est fini !
3 - Séquence de signaux(*)
Si NON
COB1,XP1,eCO,FIN
Si OUI
- RIB1, XS, ECO, FIN
Exemple 7
JUMP C Direct, 3000
Exemple du Intel8085
b7 b6 b5 b4 b3 b2 b1 b0
b0 : Retenue
b1 : Non attribué
b2 : Parité
b3 : Non attribué
b4 : Demi-retenue
b5 : Non attribué
b6 : Zéro
b7 : Signe
20

20

Important

- Les Accumulateurs dont la valeur n'est pas définie dans l'instruction **doivent rester inchangés** après.
- Les registre débanalisés (dont la fonction, comme par exemple le Compteur Ordinal ou le Registre Instruction) **ne peuvent pas servir de registre de stockage** intermédiaire.

21

Architecture

Chapitre 3

Décodage des Instructions

Plan

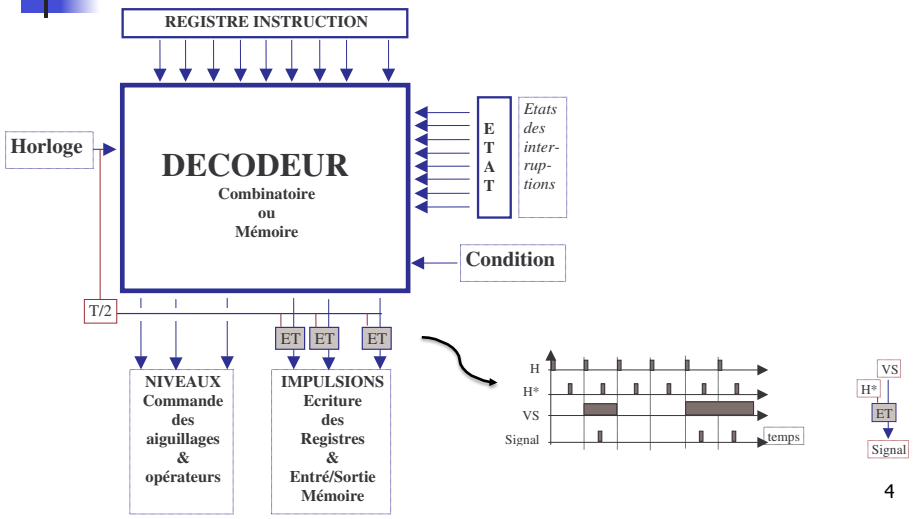
- Principe
- Décodeur « câblé »
- Décodeur « microprogrammé »
- Simulateur

Principe du décodage

- C'est l'opération qui consiste à produire les signaux nécessaires à la réalisation de chacune des instructions. Elle est effectuée par le Décodeur ou Séquenceur d'instruction.

3

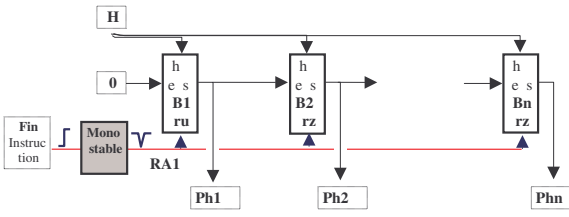
Schéma de Principe



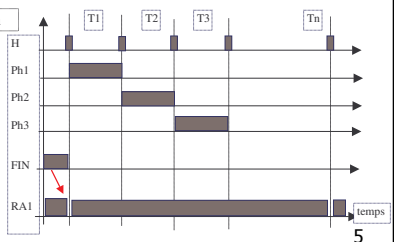
4

Décodeur « Câblé »

- Marquage du temps : Compteur de Phase



rz *est une entrée qui force la bascule à 0, indépendamment de l'horloge*
ru *est une entrée qui force la bascule à 1, indépendamment de l'horloge*



5

Pour chaque signal, on cherche pour quelle instruction,
et à quelle date (Phi) il doit être vrai.
(Phi est vrai pendant la iéme période d'horloge).

ABCD	*CodeOp*
OOOO	NoOp
1OOO	Chargt A
O1OO	Rangt A
11OO	BrIncond
etc....	

EF	*ModeAdr*
OO	Immédiat
1O	Direct
O1	Indirect
11	Indexé
etc....	

Registre Instruction

Format des Instructions

ABCD EF XXXXXXXXXX

CodeOp MA suite

Instruction _ABCDEF	NoOP OOOO OO	ChargtA - AdrImm 1OOO OO	ChargtA - AdrDir 1OOO 1O	etc... etc...
PH1	COB1,XS,eRAM	COB1,XS,eRAM	COB1,XS,eRAM	etc...
PH2	sM	sM	sM	etc...
PH3	REB1,XS,eRI	REB1,XS,eRI	REB1,XS,eRI	etc...
PH4	COB1,XP1,eCO,Fin	RIB1,XS,EA	RIB1,XS,eRAM	
PH5		COB1,XP1,eCO,Fin	sM	
PH6			REB1,XS,eA	
PH7			COB1,XP1,eCO,Fin	
etc....				

Equations logiques des signaux

COB1 = PH1 + PH4.$\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}\bar{F}$ + PH5.$\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}\bar{F}$ + PH7.$\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}\bar{F}$ + etc.....

eRAM = PH1 + PH4.$\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}\bar{F}$ + etc....

etc...

Page 20

6

Décodeur « Câblé »

- Avantages
  - Rapidité car créé spécifiquement
- Inconvénients
  - « Difficulté » de mise au point
  - Flexibilité

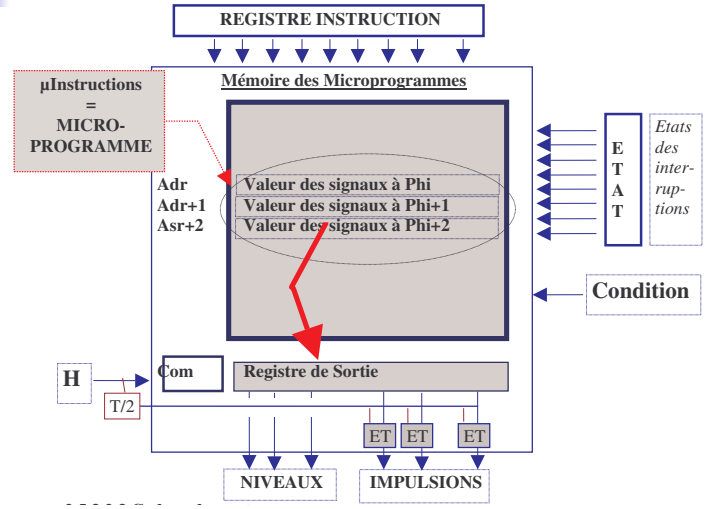
7

Décodeur « Microprogrammé »

- Principe
  - Une mémoire contient, pour chaque instruction et pour chaque phase de l'instruction, un mot dans lequel est codé les signaux à appliquer
  - Un nouveau mot est lu à chaque période d'horloge et sa sortie sur les lignes produit les signaux

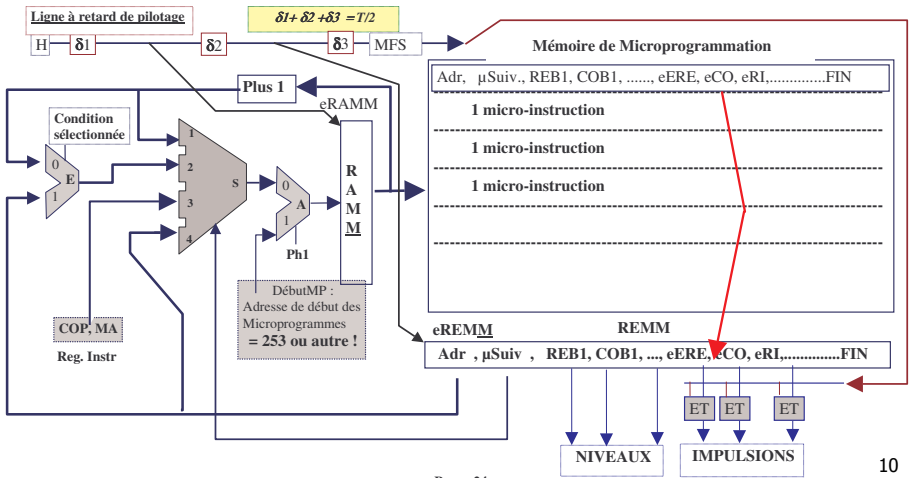
8

Décodeur « Microprogrammé »



9

Décodeur « Microprogrammé »



10

Exemple de microprogramme

- LOAD A, Immédiat, RA — COP, MA = 1

  Ph1
  Ph2 Cycle de recherche de l'instruction
  Ph3
  Ph4 RIB,1 XS, eA
  Ph5 COB, XP1, eCO, FIN

- JUMP Direct, RA — COP, MA = 2

  Ph1
  Ph2 Cycle de recherche de l'instruction
  Ph3
  Ph4 RIB1, XS, eCO, FIN

- JUMPC Direct, RA — COP, MA = 3

  Ph1
  Ph2 Cycle de recherche de l'instruction
  Ph3

  Si la Condition sélectionnée est VRAIE
  Ph4 RIB,1 XS, eCO, FIN

  Si la Condition sélectionnée est FAUSSE
  Ph4 COB1 ,XS, eCO, FIN

Hypothèse: le jeu d'instruction a 65 instructions seulement, sinon la zone de début serait plus longue…	Adresses	Sel.Suiv	Adr. µSuiv	FIN	REB1	RIB1	COB1	XS	XP1	…	…	eRA	eCO	eRI	eRAM	sM	…	*Commentaire*
Début de Zone de Début des Instructions	1	4	66			1		1				1						Chargemt A - Immédiat - RIB1,XS,eRA
	2	x	x	1		1		1					1					Branch Incond Direct - RIB1,XS,eCO,FIN
	3	4	67															Branch Cond Direct - Saut Obligatoire
	4																	
	…																	
	…																	
	64																	
Fin de Zone de Début des Instructions	65																	
Début Zone de suite des instructions	66	x	x	1			1		1				1					COB1,XP1,eCO,FIN
	67	2	69															saut conditionnel
	68	x	x	1			1		1				1					COB1,XP1,eCO,FIN
	69	x	x	1		1		1					1					RIB1,XS,eCO,FIN
	70																	
Fin Zone de suite des instructions	252																	
Fetch	253	1	x				1	1						1				COB1,XS,eRAM
Cycle de recherche de l'instruction	254	1	x													1		sM
	255	3	x		1			1					1					REB1,XS,eRI

Décodeur « Microprogrammé »

- Avantages
  - Flexibilité
- Inconvénients
  - Surface
  - Performance

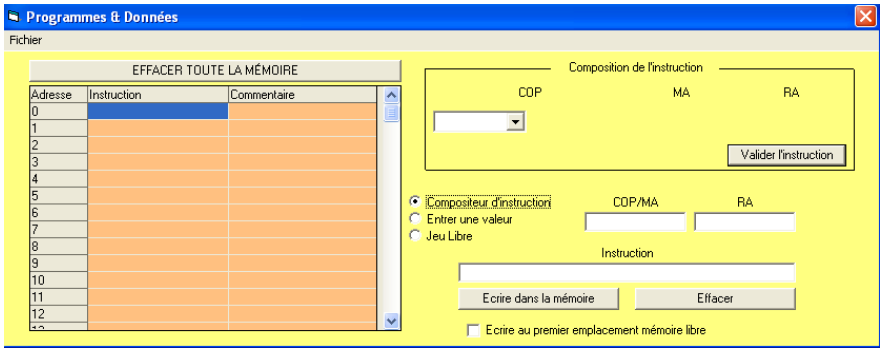
13

Simulateur

- Emulateur d'architecture
  - Mémoire Programme et donnée avec compositeur d'instruction
  - Unité de traitement : Architecture 3 bus
  - Séquenceur : Décodeur d'instruction microprogrammé

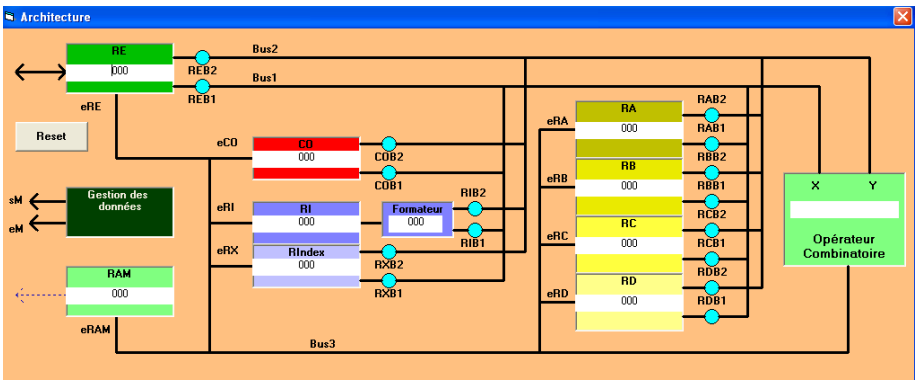
14

Mémoire Programme/Data



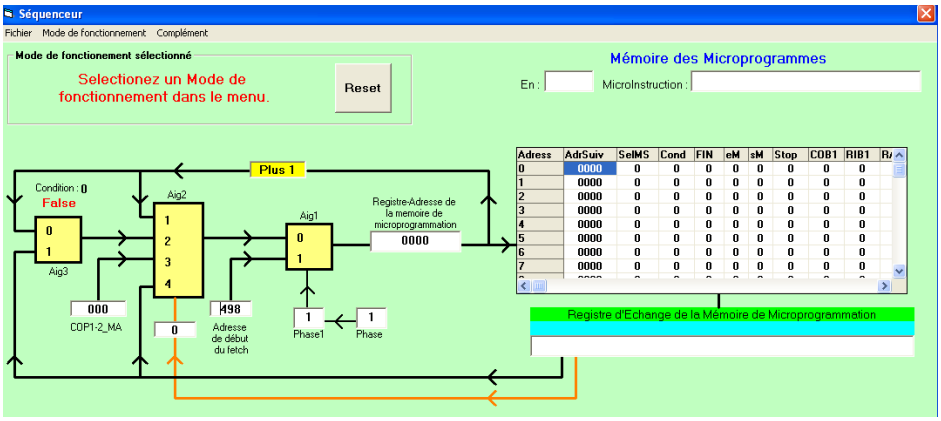
15

Unité de Traitement



16

Séquenceur



17

Architecture

Chapitre 4

Interruptions

Plan

- **Principe**
- Schéma d'acquisition
- Les états d'une interruption
- Décentralisation des interruptions
- Exemples

Principe

- Le besoin
  - Répondre à un appel du clavier, d'une alarme pendant que l'UC fait autre chose
- Il faut donc :
  - **Suspendre** un programme pour en lancer un autre
  - Pouvoir faire ça **n'importe** quand
  - Pouvoir **revenir** au programme suspendu

3

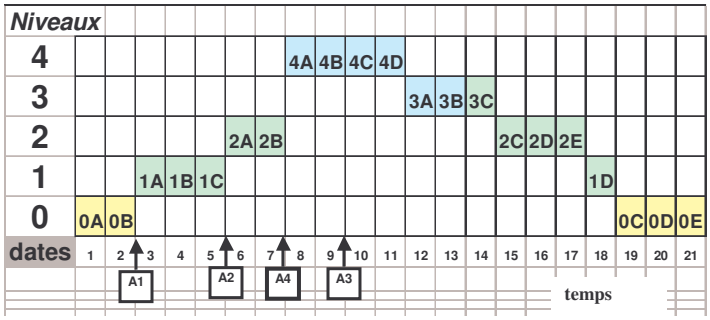
Principe

- Conséquence :
  - Les programmes sont classés en **NIVEAUX DE PRIORITE**
  - A tout instant, c'est le programme le plus prioritaire qui s'exécute.

4

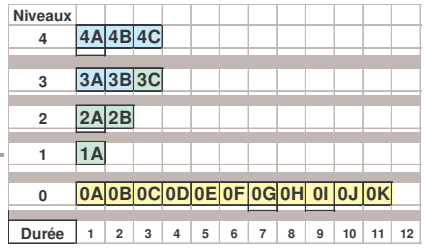
Exemple

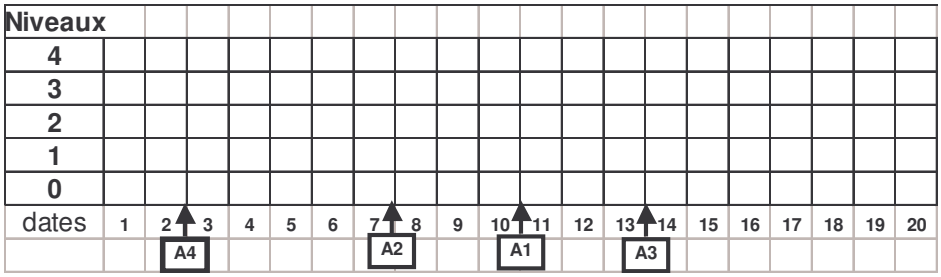
- Soient les programmes 0-ABCDE, 1-ABC, 2-ABCDE, 3-ABC, 4-ABCD associés chacun à un niveau de priorité.



5

Exercice 1





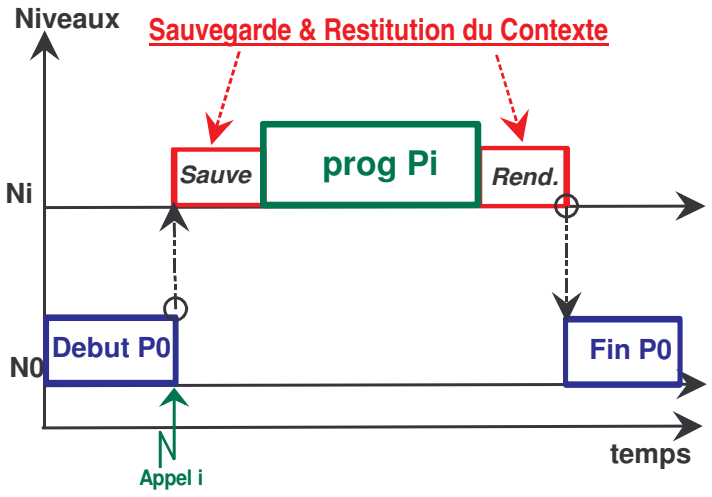
6

Conséquence de la Définition

- Puisque les 2 programmes utilisent les même éléments de l'UC (Accus, registres, Drapeaux : **LE CONTEXTE**) l'exécution de Pi écrase le contenu du contexte de P0.
- Il faut **sauvegarder le contexte** de P0 **avant** d'exécuter Pi pour pouvoir revenir en P0 après. Quand Pi est fini, il faut revenir exécuter la suite de P0. Pour reprendre P0 il faut **restituer le contexte.**

7

Sauvegarde du Contexte

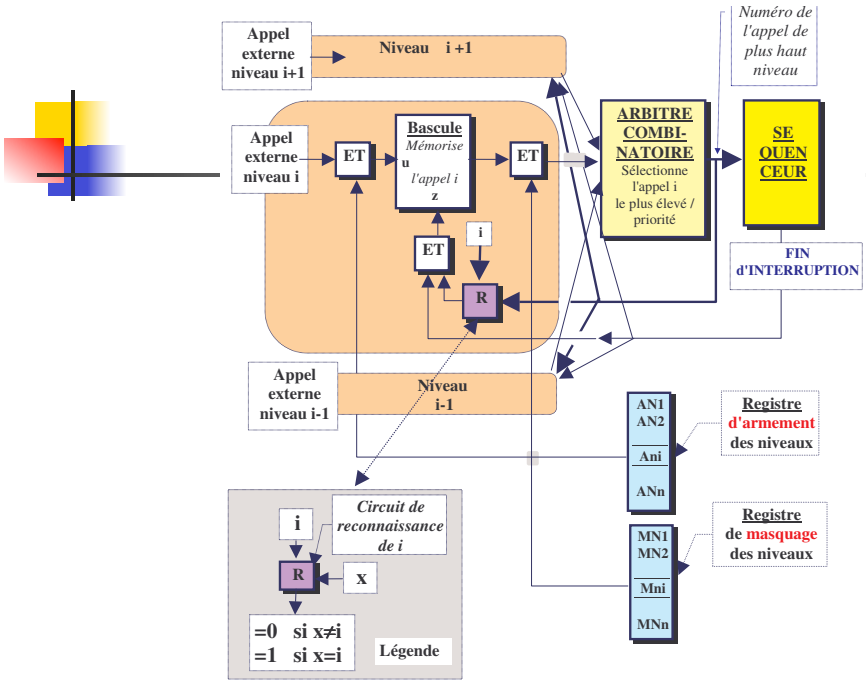


8

Plan

- Principe
- **Schéma d'acquisition**
- Les états d'une interruption
- Décentralisation des interruptions
- Exemples

9

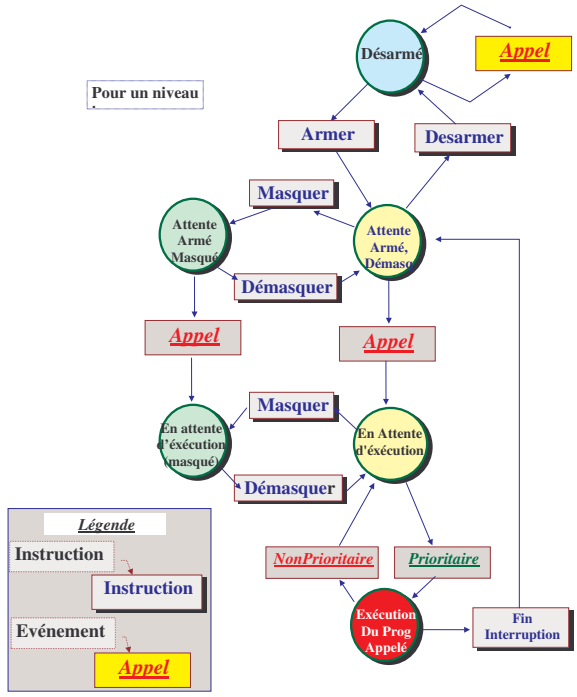


10

Plan

- Principe
- Schéma d'acquisition
- **Les états d'une interruption**
- Décentralisation des interruptions
- Exemples

11



12

Plan

- Principe
- Schéma d'acquisition
- Les états d'une interruption
- **Décentralisation des interruptions**
- Exemples

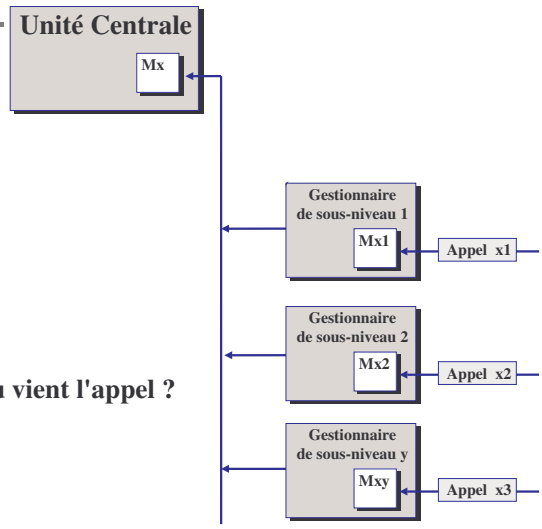
13

Décentralisation des Interruptions

- Besoin
  - Augmenter le nombre d'appels possibles sans augmenter le nb de connexions de l'UC
  - Rendre le système modulaire
- Conséquence
  - Décentralisation de l'acquisition

14

Principe

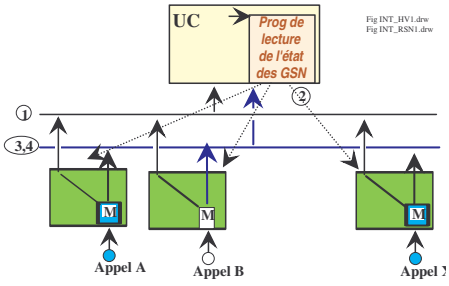


Comment reconnaitre d'ou vient l'appel ?

- **par scrutation**
- **par vecteur**

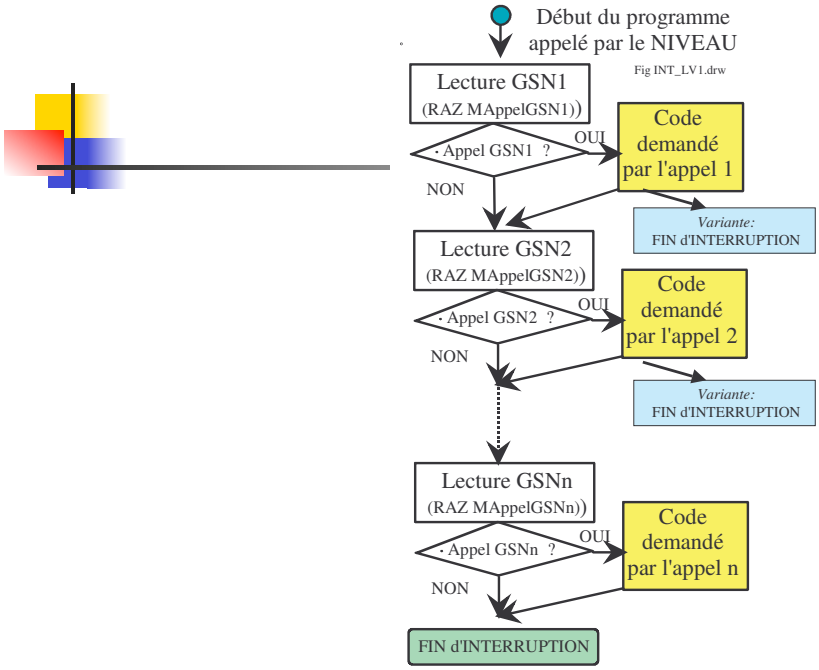
15

Traitement par Scrutation



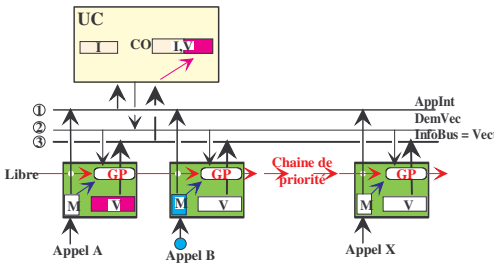
1. L'appel déclenche un programme
2. Ce programme vient lire SUCCESSIVEMENT tous les GSNs pour connaître leur état (Appel ou Non). Leur lecture RAZ l'appel.
3. Si le GSN a un appel, le programme se branche sur « une branche » contenant le programme Pn demandé par l'appel n.
4. Sinon le programme passe à la lecture du GSN suivant.
5. Quand il les a tous explorés, il se termine par l'instruction « FIN d'INTERRUPTION ».
6. Si un GSN a été appelé après, l'UC repart en interruption.

16



17

Traitement par Vectorisation

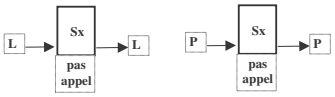


1. Un appel arrive sur un GSN. Il met la « valeur active » sur la ligne (1) ce qui appelle le niveau N de l'UC.
2. L'UC envoie un signal (2) de demande du Vecteur du GSN appelant à tous les GSN.
3. LE GSN appelant (et prioritaire si plusieurs appels) présente sur le Bus d'information (3) le Vecteur.
4. L'UC concatène « I » et « V » pour constituer l'adresse du programme qui doit être lancé.

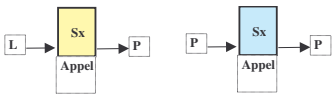
18

Chaîne de Priorité

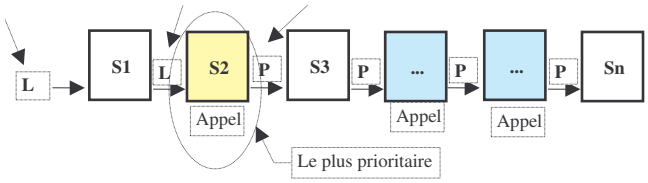
- S'il n'y a pas d'appel, la valeur de sortie est égale à celle de l'entrée



- S'il y a un appel, la valeur de sortie est égal à P



- Exemple :



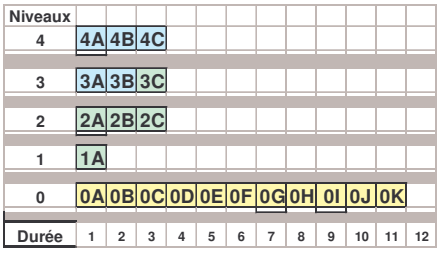
19

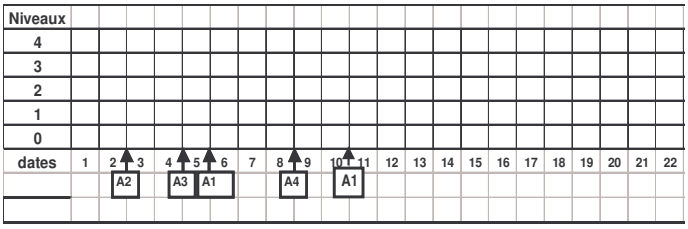
Plan

- Principe
- Schéma d'acquisition
- Les états d'une interruption
- Décentralisation des interruptions
- **Exemples**

20

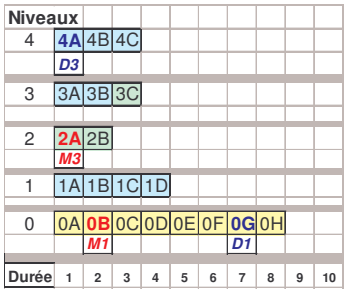
Exemple 1

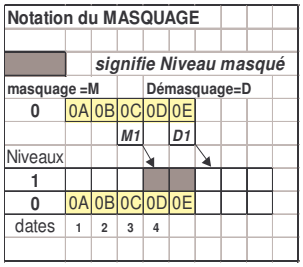


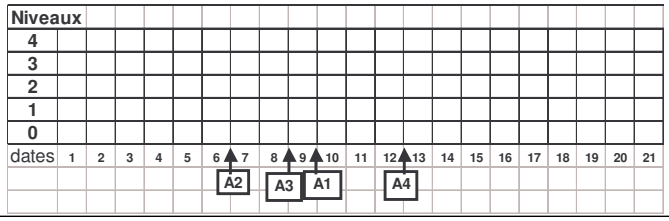


21

Exemple 2

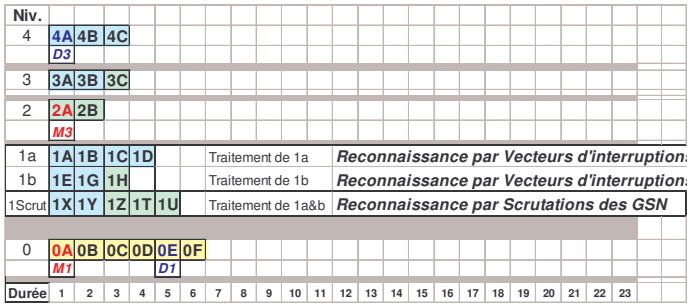






22

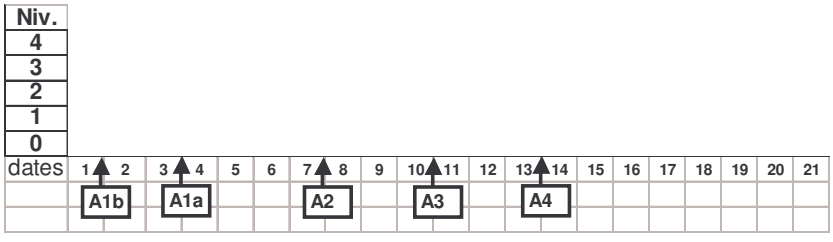
Exemple 3



Exemple 3.1

Cas n°1 - Reconnaissance par vecteurs d'interruption

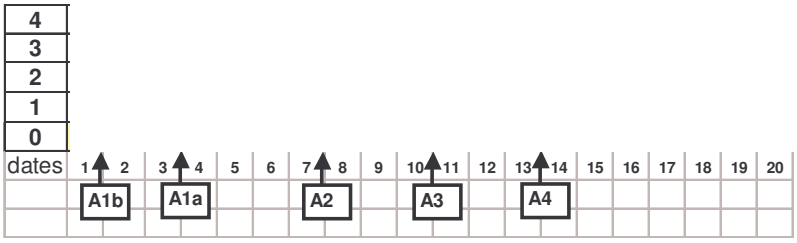
Ordre de priorité des GSN sur la chaine de priorité : 1a - 1b



Exemple 3.2

Cas n°2 - Reconnaissance par scrutation des GSN

Ordre de scrutation des GSN sur la chaine de priorité : 1ab - 1b



25

Architecture

Chapitre 5

Entrées / Sorties

Plan

- **Généralités**
- Liaison Programmée

Généralités

- Il existe différentes façons d'échanger des informations entre l'UC et l'extérieur :
  - la Liaison Programmée
    - 1 échange par instruction
  - la Liaison Canal
    - initialisée par programme et déroulement automatique
  - l'Accés Direct Mémoire
    - déroulement automatique de l'extérieur

3

Plan

- Généralités
- **Liaison Programmée**

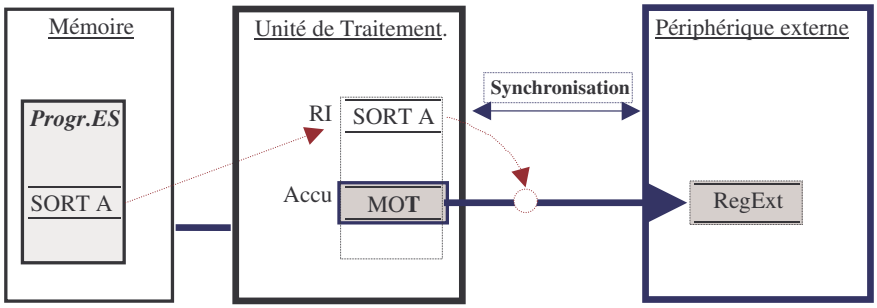
4

Définition

- C'est un type de liaison qui permet d'échanger un mot à la fois entre le contenu d'un registre de l'UC et l' extérieur.
- Il est effectué au moment de l'éxécution d'une instruction (spéciale ou banale selon la nature du périphérique) d'ou le nom de liaison programmée - cadencée par une instruction d'un programme.

5

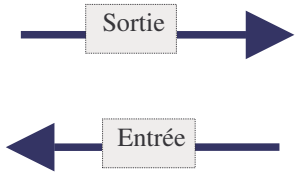
Principe



6

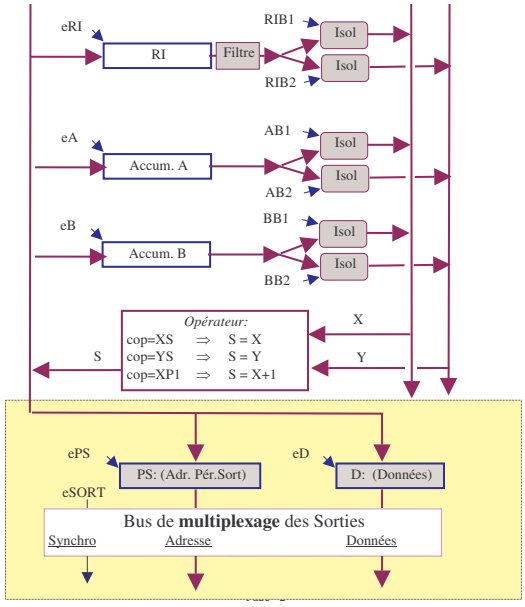
Convention

Entrée ou Sortie



7

Architecture de Sortie



Déroulement de l'Instruction

- Principe
  - Charger le N° (adresse) du périphérique dans PS
  - Charger A dans D
  - Emettre l'impulsion eSORT de synchronisation

9

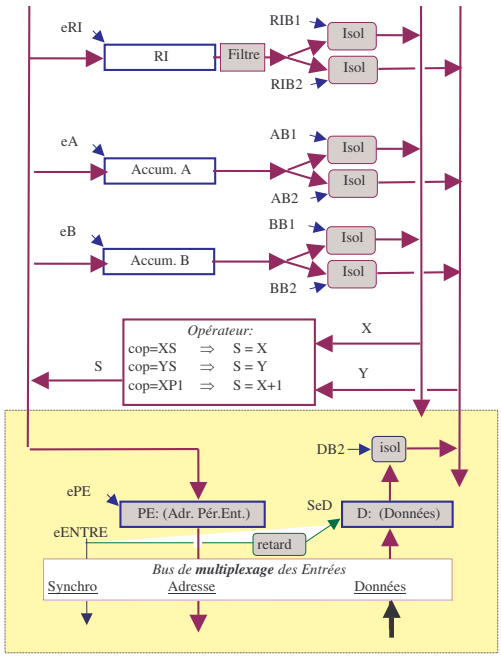
Déroulement de l'Instruction

OUT A, Direct, RA

- Séquence des signaux
  - RIB1, XS, ePS
  - AB1, XS, eD
  - eSORT, COB1, XP1, eCO, FIN

10

Architecture d'Entrée



Déroulement de l'Instruction

- Principe
  - Emettre l'impulsion eENTRE vers les périphériques
  - Celui dont le numéro est sur le bus d'adresse doit émettre son information sur le bus de Données.
  - le signal SeD (eENTRE retardé) permet alors d'enregistrer l'information dans le registre D.
  - Transfer de D vers un accumulateur.

12

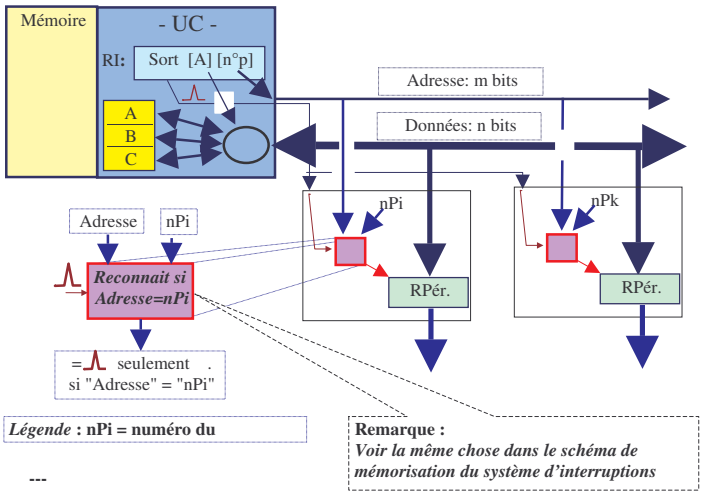
Déroulement de l'Instruction

IN A, Direct, RA

- Séquence des signaux
  - RIB1, XS, ePE
  - eENTRE, COB1, XP1, eCO
  - DB2, YS, eA, FIN

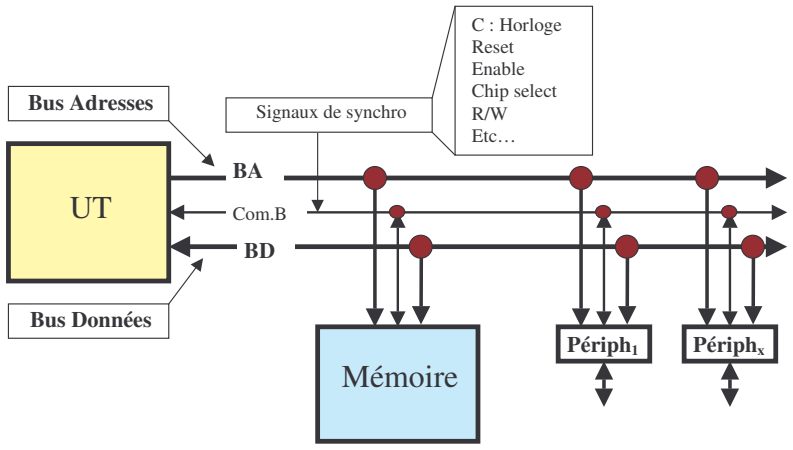
13

Structure d'Entrée/Sortie



14

Structure à Bus commun



15

Architecture

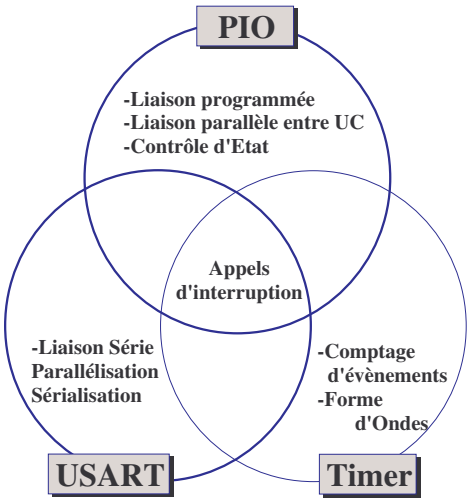
Chapitre 6

Circuits d'Interfaces

Plan

- **Généralités**
- PIO

Généralités



3

Plan

- Généralités
- **PIO**

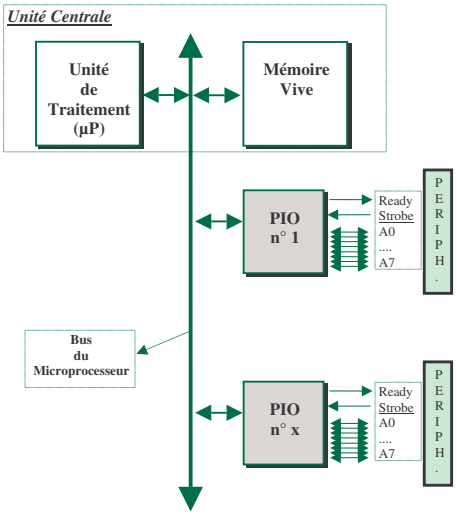
4

Définition

- C'est un circuit intégré que l'on connecte sur le bus du microprocesseur et qui permet de construire une liaison programmée ou un système d'appel par interruption

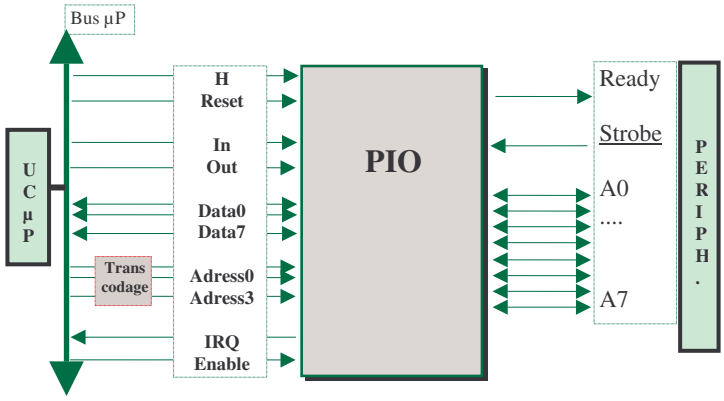
5

Implantation



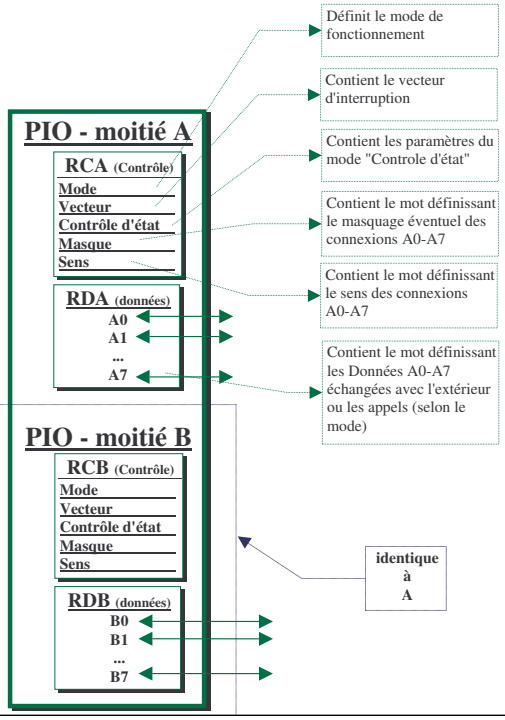
6

Connexions

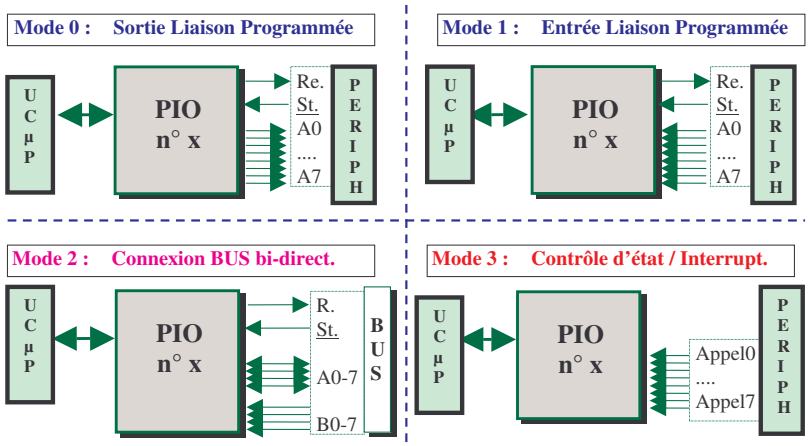


7

Structure

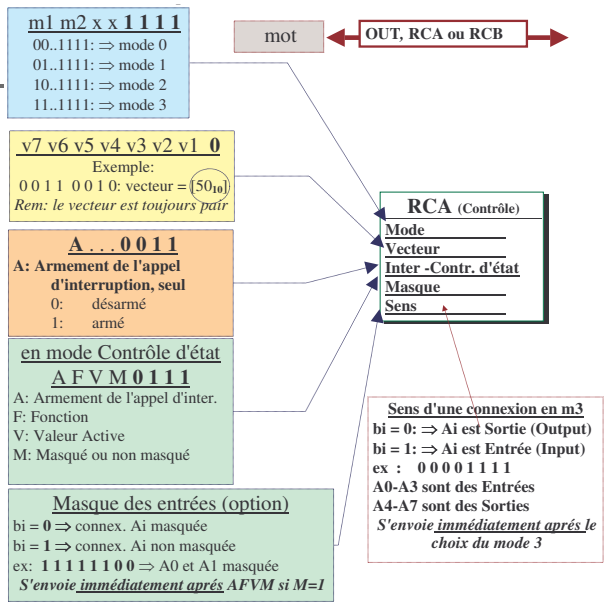


Modes de Fonctionnement



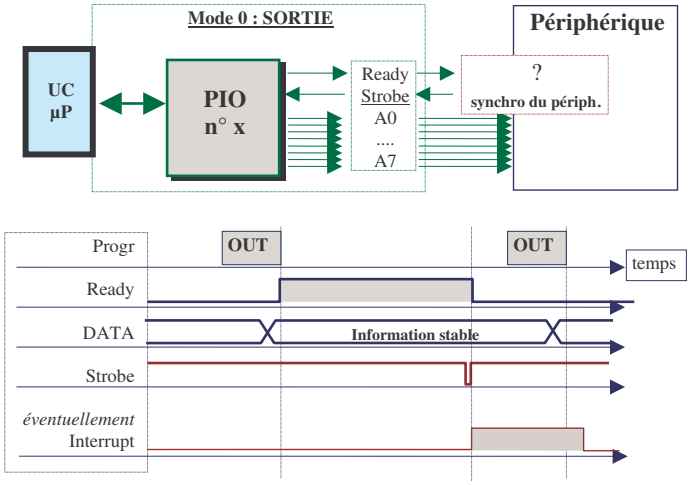
9

Configurations



10

Mode 0 - Sortie

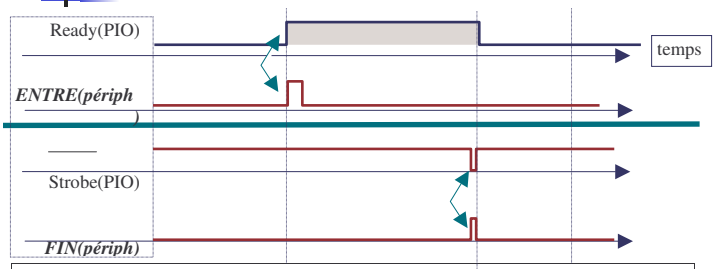


Mode 0 - Sortie

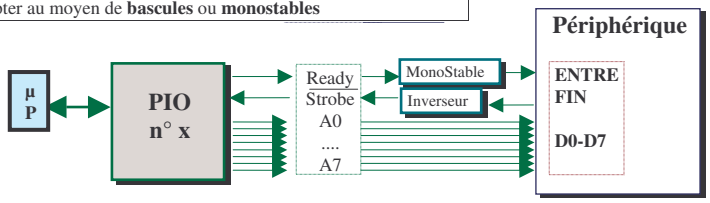
- Définition du mode — 00..*1111* — **OUT RCA**
- Définition du vecteur — [Vecteur] — **OUT RCA**
- Armement de l'interruption — 1...*0011* — **OUT RCA**
- Sortie de la donnée — Donnée — **OUT RDA**

Exemple 1

Périphérique de sortie 8 bits

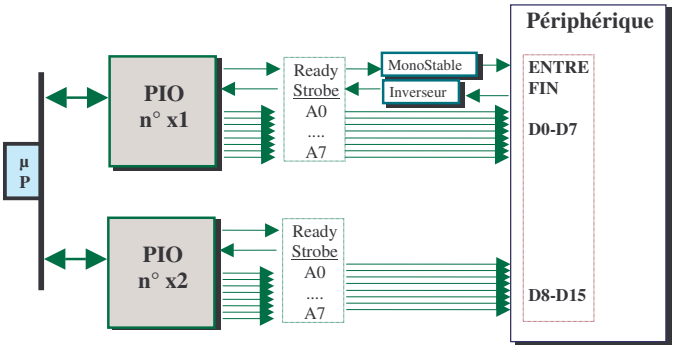


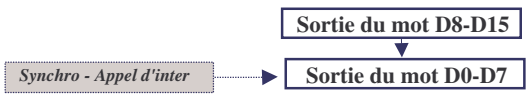
Les signaux des deux éléments répondent au même besoin de synchro mais n'ont pas la même forme. Il faut les adapter au moyen de **bascules** ou **monostables**



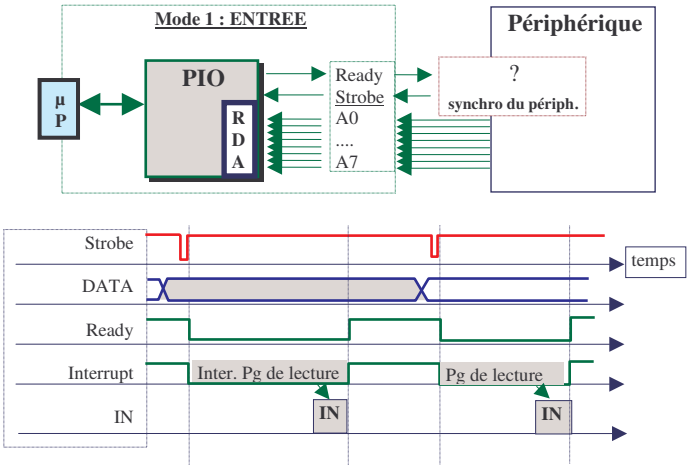
Exemple 2

Périphérique de sortie 16 bits





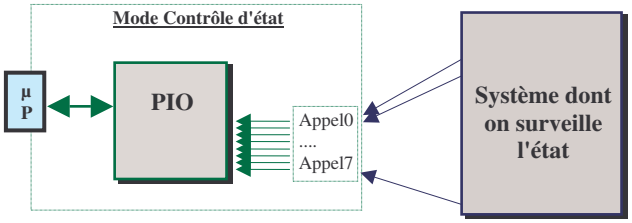
Mode 1 - Entrée



Mode 1 - Entrée

- Définition du mode
- Définition du Vecteur
- Armement de l'interruption
- Entrée de la donnée

Mode 3 – Contrôle d'état



17

Exemple 3

- Soit une pièce protégée par des détecteurs d'ouverture de porte (DP1, DP2, DP3, DP4). Quand la porte est ouverte le détecteur est à 1, sinon à 0.
- On veut déclencher un programme d'interruption si l'une des portes est ouverte

18