



# Compteur de programme (CNT)

Laboratoire Digital Design

## Contenu

1 Objectifs .....	1
2 ROM de commande de l'ALU .....	2
2.1 Circuit .....	2
2.2 Séquencement des opérations .....	2
3 Réalisation logicielle d'un port série .....	3
3.1 Transmission série .....	3
3.2 Algorithme linéaire .....	3
3.2.1 Réalisation .....	3
3.3 Algorithme avec boucles .....	4
3.3.1 Réalisation .....	4
3.4 Comparaison .....	4

## 1 | Objectifs

Ce laboratoire présente l'implémentation d'un code de programme à l'aide d'une mémoire morte (ROM) et exerce la réalisation d'un compteur de programme (Program Counter (ROM)).  
Il exerce aussi le dessin de circuits hiérarchiques.



## 2 ROM de commande de l'ALU

### 2.1 Circuit

La Fig. 1 présente la vue simplifiée d'un processeur, avec une ALU, des registres et un compteur de programme.

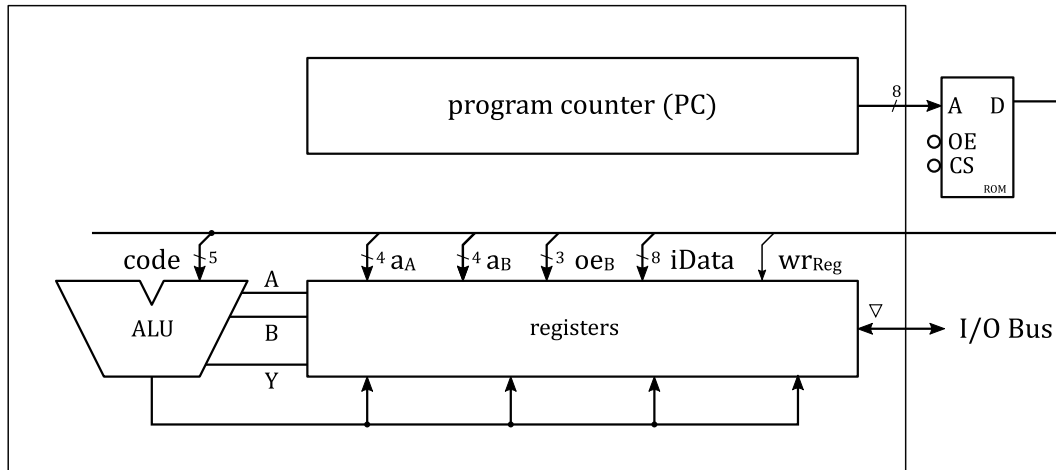


Fig. 1. – ROM contrôlant l'ALU et les registres.abbr

Les adresses de la ROM sont pilotées par un compteur de programme, ce qui permet de parcourir séquentiellement le programme qu'elle contient. Les données de la ROM codent les signaux de contrôle de l'ALU ainsi que ceux des registres.

L'entrée OE commande la sortie haute impédance de la ROM. L'entrée CS est le signal de sélection de la ROM. Toutes deux doivent être actives pour que le circuit émette des données en sortie.

### 2.2 Séquencement des opérations

Les instructions sont exécutées en 2 phases, comme présenté à la Figure Fig. 2.

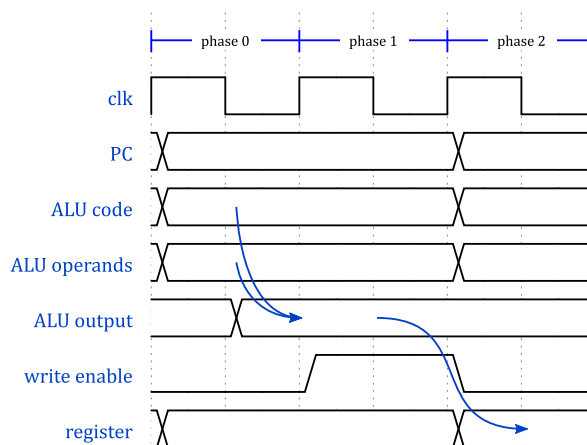


Fig. 2. – Phases des instructions

La sortie de l'ALU est stable en fin de phase 0. La nouvelle valeur du registre sélectionné est enregistrée au flanc montant de l'horloge en fin de phase 1.



## 3 | Réalisation logicielle d'un port série

### 3.1 Transmission sérielle

La Figure Fig. 3 présente le déroulement temporel de l'envoi en série d'un mot de donnée.

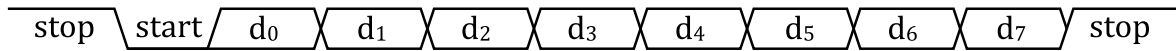


Fig. 3. – Transmission sérielle

Le signal sériel est transmis sur le bit de poids faible du port de sortie.

### 3.2 Algorithme linéaire

L'algorithme contenu dans la ROM est le suivant:

```

LOAD      s3, FF          ; load stop bit
OUTPUT    s3              ; output stop bit
LOAD      s3, s3          ; no operation
LOAD      s3, s3          ; no operation
LOAD      s3, s3          ; no operation
LOAD      s3, s3          ; no operation
LOAD      s0, 00          ; load start bit
OUTPUT    s0              ; output start bit
INPUT     s1              ; load word to send
OUTPUT    s1              ; output word, LSB is considered
SR0       s1              ; shift word, bit 1 -> LSB
OUTPUT    s1              ; output bit 1
SR0       s1              ; bit 2 -> LSB
OUTPUT    s1              ; output bit 2
SR0       s1              ; bit 3 -> LSB
OUTPUT    s1              ; output bit 3
SR0       s1              ; bit 4 -> LSB
OUTPUT    s1              ; output bit 4
SR0       s1              ; bit 5 -> LSB
OUTPUT    s1              ; output bit 5
SR0       s1              ; bit 6 -> LSB
OUTPUT    s1              ; output bit 6
SR0       s1              ; bit 7 -> LSB
OUTPUT    s1              ; output bit 7
LOAD      s3, s3          ; no operation
OUTPUT    s3              ; output stop bit

```

#### 3.2.1 Réalisation

Pour réaliser l'algorithme linéaire, le PC est un compteur unidirectionnel. Il ne permet pas de chargement d'une nouvelle valeur.

Dessiner le circuit hiérarchique du compteur de programme du microprocesseur. Ce compteur doit s'incrémenter au flanc montant de l'horloge, lorsque **incPC** = '1'. A noter que dans notre système, **incPC** est activé chaque seconde période d'horloge. Ignorer la commande de chargement d'une nouvelle valeur dans le compteur.

Simuler le système et vérifier le bon fonctionnement du compteur et du processeur.



### 3.3 Algorithme avec boucles

L'algorithme suivant permet une écriture plus compacte du programme:

```

LOAD      s3, FF          ; load stop bit
OUTPUT    s3              ; output stop bit
LOAD      s2, 04          ; initialize loop counter 3
SUB       s2, 01          ; decrement loop counter 4
JUMP NZ   03              ; loop back if not end of count 5
LOAD      s0, 00          ; load start bit 6
OUTPUT    s0              ; output start bit 7
LOAD      s2, 08          ; initialize loop counter 8
INPUT     s1              ; load word to send 9
LOAD      s3, s3          ; no operation
OUTPUT    s1              ; output word, LSB is considered
SR0       s1              ; next bit -> LSB
SUB       s2, 01          ; decrement loop counter
JUMP NZ   0A              ; loop back if not end of count
OUTPUT    s3              ; output stop bit

```

#### 3.3.1 Réalisation

Pour réaliser l'algorithme avec boucles, le PC doit permettre le chargement d'une nouvelle valeur.

Modifier le circuit hiérarchique du compteur de programme du microprocesseur afin de permettre chargement d'une nouvelle valeur dans le compteur. Changer la vue par défaut de la ROM pour sélectionner la version avec le code incluant des boucles.

Simuler le système et vérifier le bon fonctionnement du compteur et du processeur.

### 3.4 Comparaison

Pour les deux algorithmes à disposition, comparer la vitesse de transmission du mot série.