Mémoire - Microprogrammation

TD 5

I. Mémoire :

1. Temps de propagation :

Le temps d'accès à une mémoire ne dépend que des circuits internes de son boitier :

- Temps de propagation des signaux à l'intérieur de ce dernier
- Temps de commutation des transistors…

Donc, il est totalement indépendant de la distance entre le processeur et la mémoire.

Mais :

- la demande part du processeur
- elle doit arriver à la mémoire
- la mémoire prépare (produit) la (les) donnée(s)
- les données partent de la mémoire
- elles arrivent au processeur pour être utilisées dans l'opération

Alors, la distance entre les deux composants intervient entre les étapes 1 et 2, et ensuite entre les étapes 4 et 5.

Le temps de propagation peut être considéré comme négligeable, mais il ne faut pas oublier que le courant électrique se propage à environ 2/3 de la vitesse de la lumière : 200.000 km/s.

- **Exemple :**

Le courant électrique parcours 20 cm en 1 ns ! Le temps de propagation devient de moins en moins négligeable au fur et à mesure que le temps d'accès de la mémoire baisse et que la fréquence de l'horloge du processeur augmente.

Aujourd'hui, la fréquence de l'horloge des µ-processeurs est de 3.5GHz, ce qui correspond à une période de :

$$T = 1/3.5 \times 10^{9}\ s = 0.3 \times 10^{-9}\ s = 0.3\ ns$$

I. Mémoire :

2. Transfert des données :

Une mémoire RAM, relié au processeur, a un temps de cycle de 20 ns pour le premier accès et de 10 ns pour les trois accès suivants qui sont accélérés. Chaque accès récupère 8 octets.

Quelle est la bande passante (en Octets / seconde) du transfert d'information entre la mémoire et le processeur ?

On récupère 8 octets en 20 ns puis encore 3 x 8 octets les 30 ns suivants. On transfert donc 4 x 8 = 32 octets au total. Et ce transfert se fait en 20 + 3 x 10 = 50 ns.

La bande passante est donc de :

$32 / 50x10^{-9} = 640.000.000$ octets/seconde ≈ 640 Mo/s

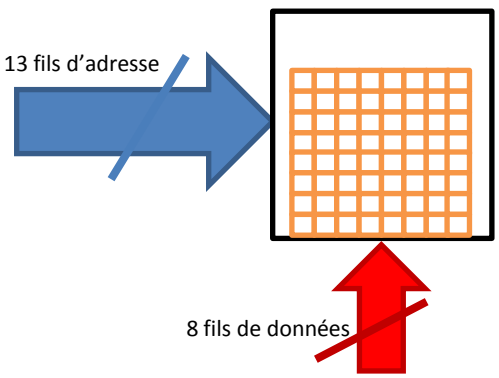
… approximativement !

Car 1 ko= 2^{10} octets = 1024 octets et 1Mo = 2^{10} ko = 1024 ko = $2^{10}x2^{10}$ octets = 2^{20} octets = 1.048.576 octets (pas exactement 1.000.000) → 610,4 Mo

I. Mémoire :

3. Mémoire et espace d'adressage :

1. Capacité d'une mémoire ayant 13 fils d'adresse et 8 fils de données (en bits, en kb, en octets et en ko) ?



13 fils d'adresses et 8 fils de données (8 fils → 8 bits → 1 octet) :

Nb de cases mémoire adressables :

2^{13} octets = 2^3x2^{10} octets = 2^3 ko = **8 ko**

Or chaque cases contient 1 octet c-à-d 8 bits (2^3 bits), cela fait :

8 kox8 bits = 2^{13} octetsx2^3 bits

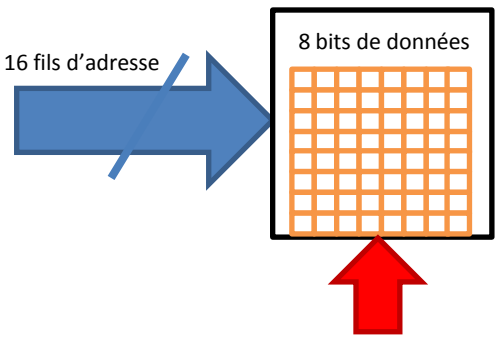
= **2^{16} bits** = 2^6x2^{10} bits

= 2^6kbits = **64 kbits**

I. Mémoire :

3. Mémoire et espace d'adressage :

2. Espace adressable d'un µprocesseur possédant un bus d'adresse de 16 fils et d'un bus de données de 8 bits (en bits, en kb, en octets et en ko) ?



16 fils d'adresses et des données sur 8 bits :

Nb de cases mémoire adressables :

2^{16} **octets** = $2^6 x 2^{10}$ octets = **64 ko**

Or chaque cases contient 1 octet c-à-d 8 bits (2^3 bits), cela fait :

64 kox8 bits = 2^{16} octets$x2^3$ bits = 2^{19} **bits**

= $2^9 x 2^{10}$ bits = **512 kbits**

I. Mémoire :

4. Décodage d'adresse :

Décodage d'adresse par des portes logiques et 2 circuits décodeurs 741HC38.

- *$\overline{CS_1}$ et $\overline{CE}$: signaux sortie du décodage d'adresse qui valident les circuits mémoires sur niveau bas*
- *JP : Jumper (cavalier : court-circuit)*
- *74HC00 : porte logique NAND*
- *74HC373 : bascules D latch*
- *74LS245 : 8 buffers (amplificateurs)*
- *MC68HC24 : PRU (port replacement Unit) pour le fonctionnement de l'UC 68HC11 en mode étendu*
- *MAX232 : Transciever (adaptateur de niveau de tension)*
- *MAX692 : Circuit de RESET avec Watchdog (chien de garde de surveillance) integré*

1. Les différents blocs fonctionnels sur le schéma ?

   L'Unité centrale : U1

   Les mémoires :

   EPROM : U16

   RAM : U17 + U18

   Les décodeurs d'adresses

I. Mémoire :

4. Décodage d'adresse :

2. Capacité mémoire des circuits U16, U17 et U18 :

U16 - Adresses : A0-A13 → 14 fils ; Données : AD0-AD7 → 8 bits donc 16 ko ou 128 kbits

U17 - Adresses : A0-A12 → 13 fils ; Données : AD0-AD7 → 8 bits donc 8 ko ou 64 kbits

U18 - Adresses : A0-A12 → 13 fils ; Données : AD0-AD7 → 8 bits donc 8 ko ou 64 kbits

3. Les équations logiques des signaux d'activation pour les blocs ROM et RAM :

ROM U16 : $\overline{CE} = \overline{A_{15}A_{14}}$ → $CE = A_{15}A_{14}$

RAM U17 : $\overline{CS_1} = \overline{Y_4}$ → CBA = HLL → $A_{15}A_{14}A_{13} = 100$ → $CS_1 = A_{15}\overline{A_{14}}\overline{A_{13}}$

RAM U18 : $\overline{CS_1} = A_{15}$ → $CS_1 = \overline{A_{15}}$

I. Mémoire :

4. Décodage d'adresse :

4. Pour les mêmes blocs adresses de début et de fin d'accès en binaire : valeurs des lignes d'adresse A15 à A0 et en hexadécimal ($0000 à $FFFF) :

	Circuit	[A15	A14	A13	A12]	[A11	A10	A9	A8] A7… A0	@ hexa	
Adr A_0-A_{13}	U 16	1	1	0	0	0	0	0	0	0000	@début
CE = $A_{15}A_{14}$ → A_{15}=1; A_{14}=1		1	1	1	1	1	1	1	1	FFFF	@fin
Adr A_0-A_{12}	U 17	1	0	0	0	0	0	0	0	8000	@début
$\overline{CS_1} = A_{15}\overline{A_{14}}\overline{A_{13}}$ → A_{15}=1; A_{14}=0; A_{13}=0		1	0	0	1	1	1	1	1	9FFF	@fin
Adr A_0-A_{12}	U 18	0	X	X	0	0	0	0	0	0000	@début
$\overline{CS_1} = \overline{A_{15}}$ → A_{15}=0		0	X	X	1	1	1	1	1	7FFF	@fin

U18 (8ko) : 0x0000 à 0x7FFF (4 fois l'espace nécessaire car 2 bits non utilisés → zone miroir)
U17 (8ko) : 0x8000 à 0x9FFF
U16 (16ko) : 0xC000 à 0xFFFF

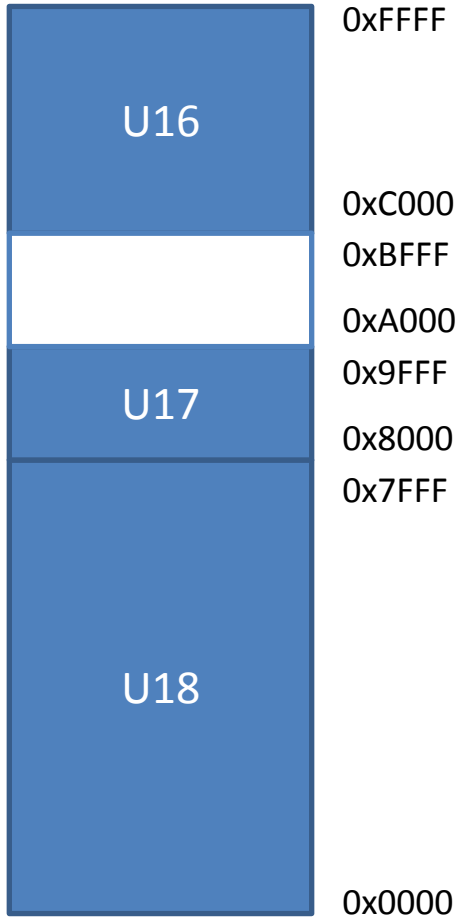
(Actuellement le décodage d'adresse est plutôt fait à l'aide de circuits de type PAL – Programmable Array Logic, programmé en Abel ou VHDL)

I. Mémoire :

4. Décodage d'adresse :

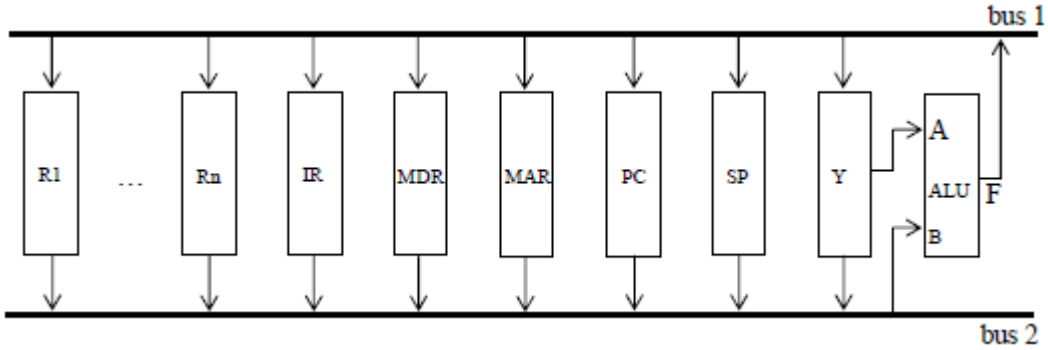
5. Dessin de la carte mémoire du système :

Circuit	[A15	A14	A13	A12]	[A11	A10	A9	A8] A7… A0	@ hexa	
U 16	1	1	0	0	0	0	0	0	C000	@début
	1	1	1	1	1	1	1	1	FFFF	@fin
U 17	1	0	0	0	0	0	0	0	8000	@début
	1	0	0	1	1	1	1	1	9FFF	@fin
U 18	0	X	X	0	0	0	0	0	0000	@début
	0	X	X	1	1	1	1	1	7FFF	@fin



II. Machine de von Neumann :

1. Ex : BNZ R4, dep :



- R1 à Rn : registres généraux
- IR : registre d'instruction
- MDR : registre de donnée d'échange avec la mémoire
- MAR : registre d'adresse d'échange avec la mémoire
- PC : compteur ordinal
- SP : pointeur de pile
- Y : registre pour mémoriser l'entrée A de l'ALU
- ALU : Unité Arithmétique et Logique

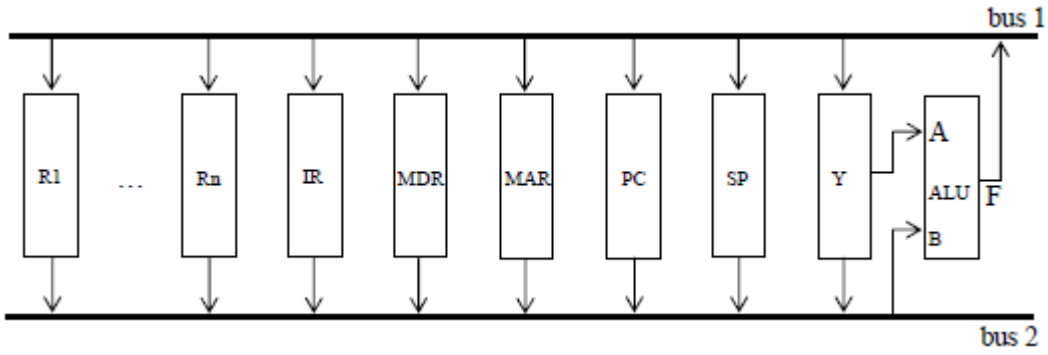
Micro-instructions (étapes possibles du traitement d'une instruction machine) :

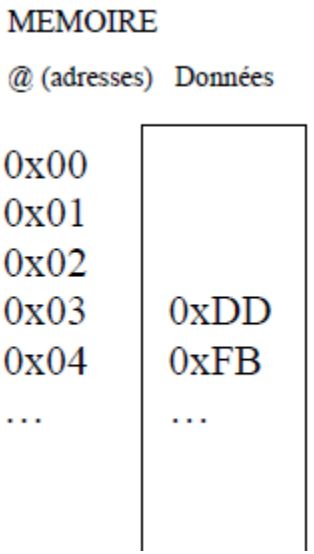
- Reg out : sort le contenu du registre Reg sur le bus 2
- Reg in : met la valeur du bus 1 dans le registre Reg
- Lecture : effectue une lecture de la mémoire à l'adresse MAR et met le résultat dans MDR
- Ecriture : effectue une écriture dans la mémoire à l'adresse MAR de la valeur contenue dans MDR
- ADD : additionne les entrées A et B de l'ALU, met le résultat sur la sortie de l'ALU (F = A+B)
- INCRA : incrémente l'entrée A de l'ALU (F = A+1)
- DECRA : décrémente l'entrée A de l'ALU (F = A-1)
- INCRB : incrémente l'entrée B de l'ALU (F = B+1)
- DECRB : décrémente l'entrée B de l'ALU (F = B-1)
- REPA : reporte l'entrée A de l'ALU sur sa sortie (F = A)
- REPB : reporte l'entrée B de l'ALU sur sa sortie (F = B)

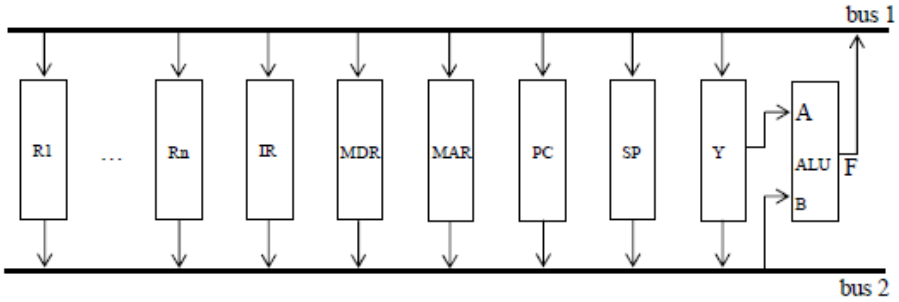
II. Machine de von Neumann :

1. Ex : BNZ R4, dep :

BNZ R4, -5 = (0xDD et 0xFB), en mémoire aux adresses 0x03 et 0x04,
Le registre R4 contient 0x02 et on est juste avant le traitement de cette instruction (PC=IP=CO=0x03)







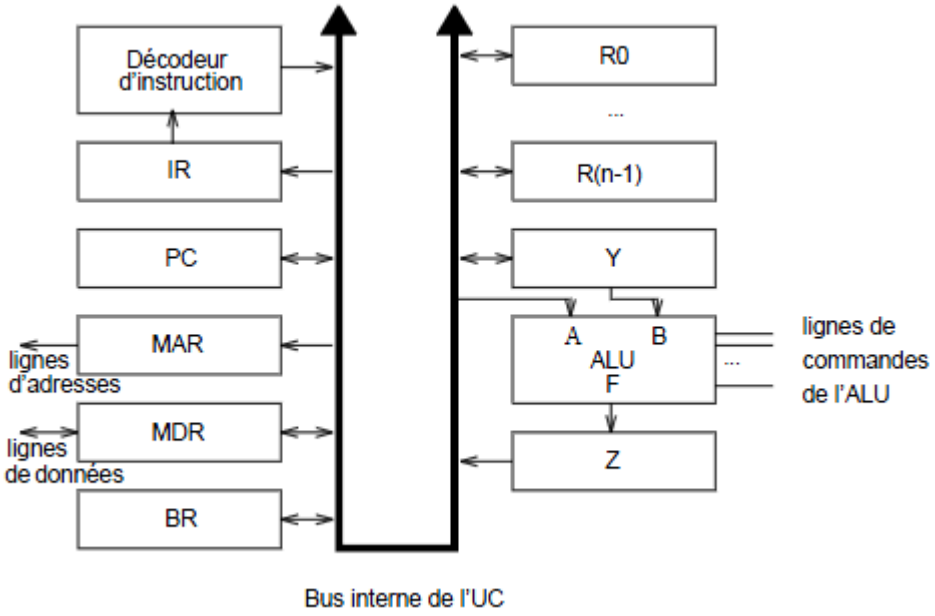
MEMOIRE

@ (adresses)	Données
0x00	
0x01	
0x02	
0x03	0xDD
0x04	0xFB
...	...

R4	IR	MDR	MAR	PC	Y	Bus1	Bus2	µ-inst	Commentaire	Step
0x02				0x03					Etat initial	1
							0x03	PC out	(PC) → Bus 2	2
						0x03		REPB	F=(PC)=0x03	3
			0x03					MAR in	MAR = 0x03	4
						0x04		INCRB	F=0x04	5
				0x04				PC in	PC = 0X04	6
		0xDD						Lecture	0xDD → MDR	7
							0xDD	MDR out	(MDR) → Bus2	8
						0xDD		REPB	F = (MDR) = 0xDD	9
	0xDD							IR in	IR = 0xDD	10
							0x04	PC out	(PC) → Bus2	11
						0x04		REPB	F = (PC) = 0x04	12
			0x04					MAR in	MAR = 0X04	13
						0x05		INCRB	F = (PC) + 1 = 0x05	14
				0x05				PC in	PC = 0x05	15
		0xFB						Lecture	0xFB → MDR	16
					0x05			Y in	Y = (PC)	17
							0xFB	MDR out	0xFB → Bus2	18
						0x00		ADD	F = (PC) + (MDR) = 0x05+0xFB	19
				0x00				PC in	PC = 0x00	20

II. Machine de von Neumann :

2. Ex : Modes d'adressage :

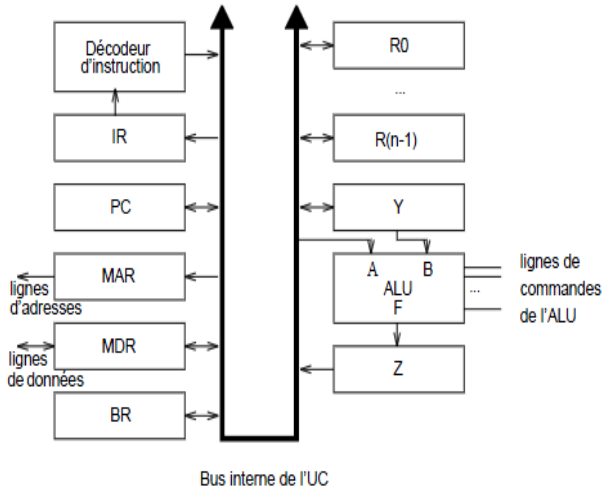


Code	Anglais	Français
IR	Instruction Register	Registre d'Instruction
MDR	Memory Data Register	Registre de Données Mémoire
MAR	Memory Address Register	Registre d'Adresse Mémoire
PC	Program Counter	Compteur Ordinal
SP	Stack Pointer	Pointeur de Pile
ALU	Arithmetical and Logic Unit	Unité Arithmétique et Logique

Reg out	Sort le contenu du registre Reg sur le bus
Reg in	Met la valeur du bus dans le registre Reg
Lecture	Effectue une lecture de la mémoire à l'adresse MAR et met le résultat dans MDR
Ecriture	Effectue une écriture dans la mémoire à l'adresse MAR de la valeur contenue dans MDR
Attente	Permet d'attendre la fin d'une lecture ou d'une écriture
ADD	Additionne les entrées A et B de l'ALU, met le résultat sur la sortie de l'ALU (F=A+B)
INCRA	Incrémente l'entrée A de l'ALU (F=A+1)
DECRA	Décrémente l'entrée A de l'ALU (F=A-1)
INCRB	Incrémente l'entrée B de l'ALU (F=B+1)
DECRB	Décrémente l'entrée B de l'ALU (F=B-1)
REPA	Reporte l'entrée A de l'ALU sur sa sortie (F=A)
REPB	Reporte l'entrée A de l'ALU sur sa sortie (F=B)

II. Machine de von Neumann :

2. Ex : Modes d'adressage :

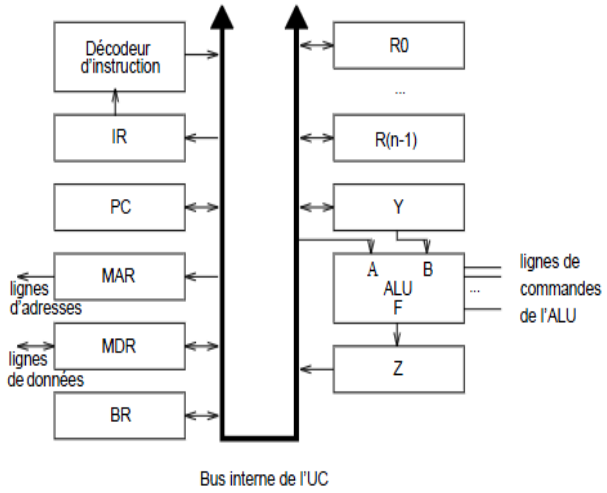


0x00	...	Instruction 1
0x01	...	
0x02	ADD/1	Instruction 2 : a <- a + 10
0x03	0x0A	
0x04	ADD/2	Instruction 3 : a <- a + b
0x05	0Xff	
0x06	ADD/3	Instruction 4 : a <- a + *c
0x07	0xFE	
0x08	...	Instruction 5 :
0x09	...	
...	...	
0x10	0x92	Début du tas
...	...	
0xFE	0x10	
0xFF	0x11	Début de la pile

Reg out	Sort le contenu du registre Reg sur le bus
Reg in	Met la valeur du bus dans le registre Reg
Lecture	Effectue une lecture de la mémoire à l'adresse MAR et met le résultat dans MDR
Ecriture	Effectue une écriture dans la mémoire à l'adresse MAR de la valeur contenue dans MDR
Attente	Permet d'attendre la fin d'une lecture ou d'une écriture
ADD	Additionne les entrées A et B de l'ALU, met le résultat sur la sortie de l'ALU (F=A+B)
INCRA	Incrémente l'entrée A de l'ALU (F=A+1)
DECRA	Décrémente l'entrée A de l'ALU (F=A-1)
INCRB	Incrémente l'entrée B de l'ALU (F=B+1)
DECRB	Décrémente l'entrée B de l'ALU (F=B-1)
REPA	Reporte l'entrée A de l'ALU sur sa sortie (F=A)
REPB	Reporte l'entrée A de l'ALU sur sa sortie (F=B)

II. Machine de von Neumann :

2. Ex : Modes d'adressage :



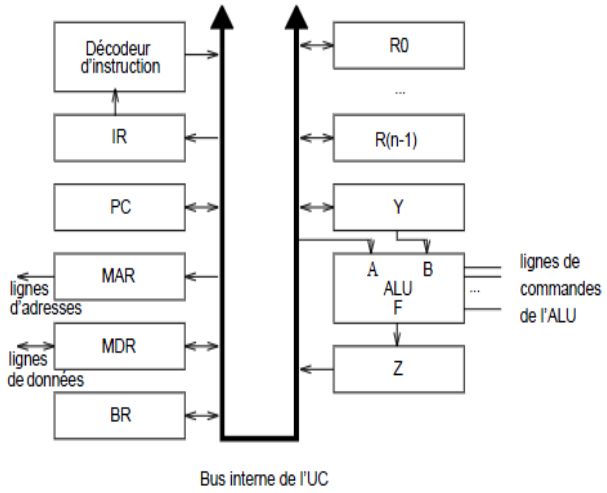
Instruction	Description
Reg out	Sort le contenu du registre Reg sur le bus
Reg in	Met la valeur du bus dans le registre Reg
Lecture	Effectue une lecture de la mémoire à l'adresse MAR et met le résultat dans MDR
Ecriture	Effectue une écriture dans la mémoire à l'adresse MAR de la valeur contenue dans MDR
Attente	Permet d'attendre la fin d'une lecture ou d'une écriture
ADD	Additionne les entrées A et B de l'ALU, met le résultat sur la sortie de l'ALU (F=A+B)
INCRA	Incrémente l'entrée A de l'ALU (F=A+1)
DECRA	Décrémente l'entrée A de l'ALU (F=A-1)
INCRB	Incrémente l'entrée B de l'ALU (F=B+1)
DECRB	Décrémente l'entrée B de l'ALU (F=B-1)
REPA	Reporte l'entrée A de l'ALU sur sa sortie (F=A)
REPB	Reporte l'entrée A de l'ALU sur sa sortie (F=B)

Quelle est la suite de micro-instructions qui permet de transférer une donnée d'un registre à l'autre ? Par exemple : R1 <- PC.

PC out
R1 in

II. Machine de von Neumann :

2. Ex : Modes d'adressage :



Instruction	Description
Reg out	Sort le contenu du registre Reg sur le bus
Reg in	Met la valeur du bus dans le registre Reg
Lecture	Effectue une lecture de la mémoire à l'adresse MAR et met le résultat dans MDR
Ecriture	Effectue une écriture dans la mémoire à l'adresse MAR de la valeur contenue dans MDR
Attente	Permet d'attendre la fin d'une lecture ou d'une écriture
ADD	Additionne les entrées A et B de l'ALU, met le résultat sur la sortie de l'ALU (F=A+B)
INCRA	Incrémente l'entrée A de l'ALU (F=A+1)
DECRA	Décrémente l'entrée A de l'ALU (F=A-1)
INCRB	Incrémente l'entrée B de l'ALU (F=B+1)
DECRB	Décrémente l'entrée B de l'ALU (F=B-1)
REPA	Reporte l'entrée A de l'ALU sur sa sortie (F=A)
REPB	Reporte l'entrée A de l'ALU sur sa sortie (F=B)

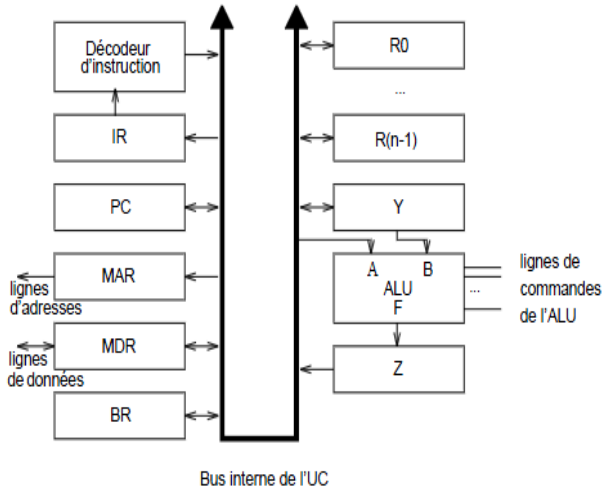
Quelle est la suite de micro-instructions qui permet d'incrémenter le compteur ordinal PC : PC++ ?

PC out
INCRA
Z out
PC in

II. Machine de von Neumann :

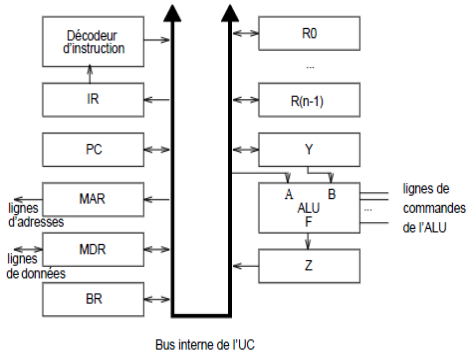
2. Ex : Modes d'adressage :

On suppose que les temps de propagation le long du bus et à travers l'ALU sont respectivement de 20 et 100 ns. Le temps d'établissement d'un registre est de 10 ns. Quel est le temps nécessaire pour effectuer les opérations vues dans les 2 questions précédentes ?



PC out → 20 ns
R1 in → 10 ns
→ **30 ns**

PC out → 20 ns
INCRA → 100 ns
Z out → 20 ns
PC in → 10 ns
→ **150 ns**



R1	R2	R3	R4	SP	IR	MDR	MAR	PC	Y	Z	Bus	µ-inst	Step
													0
													1
													2
													3
													4
													5
													6
													7
													8
													9
													10
													11
													12
													13
													14
													15
													16
													17
													18
													19
													20

0x00	…	Instruction 1
0x01	…	
0x02	ADD/1	Instruction 2 : a <- a + 10
0x03	0x0A	
0x04	ADD/2	Instruction 3 : a <- a + b
0x05	0Xff	
0x06	ADD/3	Instruction 4 : a <- a + *c
0x07	0xFE	
0x08	…	Instruction 5 :
0x09	…	
…	…	
0x10	0x92	Début du tas
…	…	
0xFE	0x10	
0xFF	0x11	Début de la pile