Université Paris-Nord — CP2I 1
Institut Galilée — Année 2008-2009
Eléments d'Informatique - 1er semestre

Devoir sur table n° 1

Question 1

barême : 2 pt

Décrire brièvement le cycle d'exécution du processeur dans une machine de Von Neumann.

Question 2

barême : 2 pt

Décrire brièvement le cycle d'exécution d'un système d'exploitation mono-tâche.

Question 3

barême : 3 pt

Les questions suivantes concernent une machine de Von Neumann dont le jeu d'instructions est donné à la fin du sujet. Il s'agit du jeu d'instructions du simulateur AMIL.

1. Décrire (en français) un algorithme en trois étapes qui échange les contenus des mémoires 100 et 101.
2. Ecrire le programme correspondant en utilisant le jeu d'instructions d'AMIL. Commenter le programme afin de le rendre compréhensible.

```
1 lecture 100 r0
2 lecture 101 r1
3 ecriture r1 100
4 ecriture r0 101
5 stop
```

Instructions	Cycles	CP	r0	r1	100	101
INIT	0	1	?	?	18	124
`lecture 100 r0`	1	2	18			
`lecture 101 r1`	2	3		124		
`ecriture r1 100`	3	4			124	
`ecriture r0 101`	4	5				18
`stop`	5	6				

Question 4

barême : 5 pt

On considère un appartement équipé d'un chauffage électrique, et d'un abonnement avec l'option *Heures Pleines / Heures Creuses*, qui permet de payer moins chère l'électricité la nuit. Par soucis d'économie, on souhaite que le chauffage se mette en route automatiquement pendant les heures creuses et s'éteigne pendant les heures pleines.

On suppose que le début des heures pleines est inscrit dans la mémoire à l'adresse 100, que le début des heures creuses se trouve à l'adresse 101 et que l'heure actuelle se trouve à l'adresse 102 (les heures sont des entiers compris entre 0 et 24). On souhaite que la machine AMIL décide

de l'état du chauffage, et écrive à l'adresse 103 la valeur 1 s'il faut que les radiateurs soient (ou restent) allumés, ou bien la valeur 0 s'il doivent être (ou rester) éteints.

1. Décrire (en français) un algorithme répondant à ce souhait.
2. Ecrire et commenter le programme correspondant en utilisant le jeu d'instructions d'AMIL.
3. Faire les traces d'exécution du programme :
   (a) lorsque les cases mémoires 100, 101 et 102 contiennent respectivement les valeurs 7, 22, et 1 ;
   (b) lorsque les cases mémoires 100, 101 et 102 contiennent respectivement les valeurs 7, 22 et 12 ;
   (c) lorsque les cases mémoires 100, 101 et 102 contiennent respectivement les valeurs 7, 22 et 23 !

```
1  lecture 100 r0          # calcul de (h - debut hp)
2  inverse r0              #
3  lecture 102 r1          # r1 contient l'heure courante h
4  add r1 r0               #
5  sisaut r0 7             # si h < debut hp
6  saut 14                 #  saut vers l'ecriture de 1
7  lecture 101 r0          # sinon : calcul de (h - debut hc)
8  inverse r0              #
9  add r1 r0               #
10 sisaut r0 14            # si h < debut hc
11 init 0 r0               #  ecriture de 0 dans 103
12 ecriture r0 103         #
13 stop                    # sinon
14 init 1 r0               #  ecriture de 1 dans 103
15 ecriture r0 103         #
```

Instructions	Cycles	CP	r0	r1	100	102	103
INIT	0	1	?	?	7	1	?
`lecture 100 r0`	1	2	7				
`inverse r0`	2	3	-7				
`lecture 102 r1`	3	4		1			
`add r1 r0`	4	5	-6				
`sisaut r0 7`	5	6					
`saut 14`	6	**14**					
`init 1 r0`	7	15	1				
`ecriture r0 103`	8	16					1
`stop`	9	17					

Instructions	Cycles	CP	r0	r1	100	101	102	103
INIT	0	1	?	?	7	22	12	?
`lecture 100 r0`	1	2	7					
`inverse r0`	2	3	-7					
`lecture 102 r1`	3	4		12				
`add r1 r0`	4	5	5					
`sisaut r0 7`	5	**7**						
`lecture 101 r0`	6	8	22					
`inverse r0`	7	9	-22					
`add r1 r0`	8	10	-10					
`sisaut r0 14`	9	11						
`init 0 r0`	10	12	0					
`ecriture r0 103`	11	13						0
`stop`	12	14						

Instructions	Cycles	CP	r0	r1	100	101	102	103
INIT	0	1	?	?	7	22	23	?
`lecture 100 r0`	1	2	7					
`inverse r0`	2	3	-7					
`lecture 102 r1`	3	4		23				
`add r1 r0`	4	5	16					
`sisaut r0 7`	5	**7**						
`lecture 101 r0`	6	8	22					
`inverse r0`	7	9	-22					
`add r1 r0`	8	10	1					
`sisaut r0 14`	9	**14**						
`init 1 r0`	10	15	1					
`ecriture r0 103`	11	16						1
`stop`	12	17						

Question 5

barême : 4 pt

On appelle n l'entier stocké à l'adresse 100 de la mémoire.

1. Décrire en français l'algorithme permettant de placer $(n!)$ à l'adresse 101 de la mémoire.
2. Ecrire et commenter le programme correspondant en utilisant le jeu d'instructions d'AMIL.
3. Faire la trace d'exécution du programme lorsque n vaut 3.

```
1 init 1 r1                  # r1 vaut i allant de 1 a n
2 init 1 r2                  # r2 vaut i!
3 lecture 100 r0             # calcul de i-n
4 inverse r0                 #
5 add r1 r0                  #
6 sisaut r0 10               # si i<n
7 add 1 r1                   #  incrementation de i
8 mult r1 r2                 #  mise-a-jour de i!
9 saut 3                     #  et on recommence
```

```
10 ecriture r2 101                    # sinon, ecriture de i! en 101
11 stop
```

Instructions	Cycles	CP	r0	r1	r2	100	101
INIT	0	1	?	?	?	3	?
`init 1 r1`	1	2		1			
`init 1 r2`	2	3			1		
`lecture 100 r0`	3	4	3				
`inverse r0`	4	5	-3				
`add r1 r0`	5	6	-2				
`sisaut r0 10`	6	7					
`add 1 r1`	7	8		2			
`mult r1 r2`	8	9			2		
`saut 3`	9	**3**					
`lecture 100 r0`	10	4	3				
`inverse r0`	11	5	-3				
`add r1 r0`	12	6	-1				
`sisaut r0 10`	13	7					
`add 1 r1`	14	8		3			
`mult r1 r2`	15	9			6		
`saut 3`	16	**3**					
`lecture 100 r0`	17	4	3				
`inverse r0`	18	5	-3				
`add r1 r0`	19	6	0				
`sisaut r0 10`	20	**10**					
`ecriture r2 101`	21	11					6
`stop`	22	12					

Question 6

barême : 4 pt

Soient deux vecteurs de dimension n, dont les composantes sont des entiers relatifs :

$$\vec{u} = \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{pmatrix}, \ \vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{pmatrix} \text{ avec } u_1 \dots u_n, v_1 \dots v_n \in \mathbb{Z}.$$

Le but de cet exercice est de faire calculer leur produit scalaire ($\vec{u} \ . \ \vec{v} = \sum_{i=1}^{n} u_i v_i$) par le simulateur AMIL.

On suppose que la dimension n des deux vecteurs est stockée à l'adresse 100 de la mémoire.
Les composantes u_1, u_2, ..., u_n de $\vec{u}$ sont stockées aux adresses 101, 102, ..., $100 + n$.
Les composantes v_1, v_2, ..., v_n de $\vec{v}$ sont stockées aux adresses $100 + n + 1$, $100 + n + 2$, ..., $100 + 2n$.

1. Décrire en français un algorithme qui calcule le produit scalaire $\vec{u} \ . \ \vec{v}$ et qui stocke le résultat à l'adresse 99 de la mémoire.
2. Ecrire le programme correspondant en utilisant le jeu d'instructions d'AMIL. Commenter le programme autant que possible. Préciser en particulier le rôle de chacun des registres que le programme utilise.

```
1  init 0 r0          # r0 : indice courant i
2  init 0 r1          # r1 : produit scalaire partiel
3  lecture 100 r2     # calcul de i-n
4  inverse r2         #
5  add r0 r2          #
6  sisaut r2 17       # si i>=n, saut vers l'ecriture du resultat
7  add 1 r0           # incrementation de i
8  init 100 r2        # calcul de 100+i
9  add r0 r2          #
10 lecture *r2 r3     # lecture de u_i dans r3
11 lecture 100 r4     # lecture de n dans r4
12 add r4 r2          # calcul de 100+n+i
13 lecture *r2 r4     # lecture de v_i dans r4
14 mult r4 r3         # calcul de u_i * v_i
15 add r3 r1          # ajout au produit partiel
16 saut 3             # et on recommence
17 ecriture r1 99     # ecriture du resultat en 99
18 stop
```

Instructions	Cycles	CP	r0	r1	r2	r3	r4	99	100	101	102	103	104	105	10
INIT	0	1	?	?	?	?	?	?	3	5	3	1	-3	5	0
`init 0 r0`	1	2	0												
`init 0 r1`	2	3		0											
`lecture 100 r2`	3	4			3										
`inverse r2`	4	5			-3										
`add r0 r2`	5	6			-3										
`sisaut r2 17`	6	7													
`add 1 r0`	7	8	1												
`init 100 r2`	8	9			100										
`add r0 r2`	9	10			101										
`lecture *r2 r3`	10	11				5									
`lecture 100 r4`	11	12					3								
`add r4 r2`	12	13			104										
`lecture *r2 r4`	13	14					-3								
`mult r4 r3`	14	15				-15									
`add r3 r1`	15	16		-15											
`saut 3`	16	**3**													
`lecture 100 r2`	17	4			3										
`inverse r2`	18	5			-3										
`add r0 r2`	19	6			-2										
`sisaut r2 17`	20	7													
`add 1 r0`	21	8	2												
`init 100 r2`	22	9			100										
`add r0 r2`	23	10			102										
`lecture *r2 r3`	24	11				3									
`lecture 100 r4`	25	12					3								
`add r4 r2`	26	13			105										
`lecture *r2 r4`	27	14					5								
`mult r4 r3`	28	15				15									
`add r3 r1`	29	16		0											
`saut 3`	30	**3**													
`lecture 100 r2`	31	4			3										
`inverse r2`	32	5			-3										
`add r0 r2`	33	6			-1										
`sisaut r2 17`	34	7													
`add 1 r0`	35	8	3												
`init 100 r2`	36	9			100										
`add r0 r2`	37	10			103										
`lecture *r2 r3`	38	11				1									
`lecture 100 r4`	39	12					3								
`add r4 r2`	40	13			106										
`lecture *r2 r4`	41	14					0								
`mult r4 r3`	42	15				0									
`add r3 r1`	43	16		0											
`saut 3`	44	**3**													
`lecture 100 r2`	45	4			3										
`inverse r2`	46	5			-3										
`add r0 r2`	47	6			0										
`sisaut r2 17`	48	**17**													
`ecriture r1 99`	49	18						0							
`stop`	50	19													

Rappel du jeu d'instructions du simulateur AMIL

`stop`	Arrête l'exécution du programme.
`noop`	N'effectue aucune opération.
`saut i`	Met le compteur ordinal à la valeur i.
`sisaut ri j`	Si la valeur contenue dans le registre i est positive ou nulle, met le compteur ordinal à la valeur j.
`init x ri`	Initialise le registre i avec la valeur x.
`lecture i rj`	Charge, dans le registre j, le contenu de la mémoire d'adresse i.
`lecture *ri rj`	Charge, dans le registre j, le contenu de la mémoire dont l'adresse est la valeur du registre i.
`ecriture ri j`	Écrit le contenu du registre i dans la mémoire d'adresse j.
`ecriture ri *rj`	Écrit le contenu du registre i dans la mémoire dont l'adresse est la valeur du registre j.
`inverse ri`	Inverse le signe du contenu du registre i.
`add x rj`	Ajoute x au contenu du registre j.
`add ri rj`	Ajoute la valeur du registre i à celle du registre j.
`mult`, `div`, `et`	Même syntaxe que pour `add` mais pour la multiplication, la division entière et le et bit à bit.