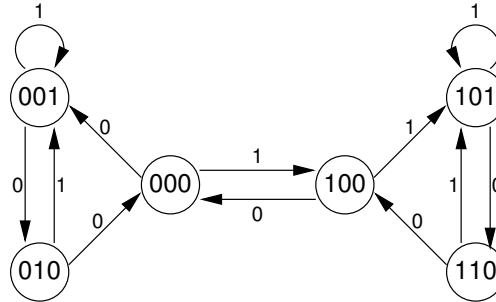
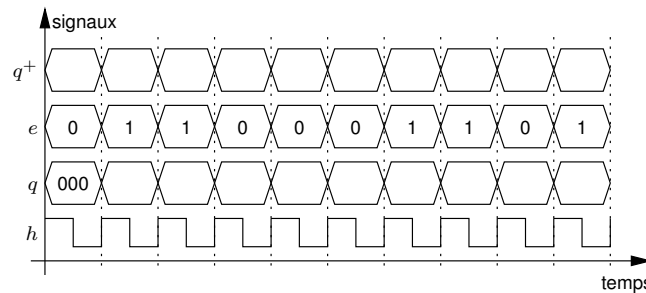


Aucun document autorisé; calculatrices, téléphones et ordinateurs interdits. Le barème pourra être modifié.
Le sujet n'est pas à rendre : notez proprement vos réponses sur votre copie.

Exercice 1 (Automates séquentiels — 7 pts). On considère l'automate fini séquentiel ci-dessous. Les états sont notés $q = (q_2q_1q_0)$, et l'entrée e ; les transitions sont étiquetées sur le schéma par e . On souhaite mettre au point un circuit séquentiel implantant cet automate. On suppose que les transitions se produisent sur un front montant de l'horloge, en fonction de l'état courant q et de l'entrée e du circuit.



Q. 1 (2 pts). Complétez le chronogramme suivant, en indiquant à chaque cycle l'état suivant et l'état courant.



Q. 2 (1 pt). Complétez la table de vérité suivante pour la fonction de transition $F : (q_2, q_1, q_0, e) \rightarrow (q_2^+, q_1^+, q_0^+)$. F peut prendre n'importe quelle valeur dans les états 011 et 111, ce qui est indiqué par les « - ».

q_2	q_1	q_0	e	q_2^+	q_1^+	q_0^+
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0	-	-	-
0	1	1	1	-	-	-
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0	-	-	-
1	1	1	1	-	-	-

Q. 3 (2 pts). Donnez des expressions pour q_2^+ , q_1^+ et q_0^+ en fonction de q_2 , q_1 , q_0 et e . Effectuez les simplifications classiques dans vos expressions pour essayer de réduire le nombre de portes utilisées.

Q. 4 (2 pts). Vous disposez d'un registre 3 bits, d'un circuit implantant la fonction de transition F , d'un multiplexeur 2×3 vers 3, et vous pouvez utiliser une constante sur 3 bits. Donnez un circuit séquentiel implantant l'automate fini demandé. Le circuit prendra en entrée les signaux h et e . Vous ajouterez une entrée r , pour faire passer le circuit à l'état initial : lorsque r est à 1 sur un cycle, le circuit repasse dans l'état 000 au cycle suivant. Il produira simplement en sortie l'état courant $q = (q_2q_1q_0)$. Annotez votre schéma.

Exercice 2 (Programmation du LC3 — 7 pts). On rappelle ci-dessous les instructions du LC3 que vous pourrez utiliser, ainsi que quelques macros :

syntaxe	action	NZP
NOT DR,SR	$DR \leftarrow \text{not } SR$	*
ADD DR,SR1,SR2	$DR \leftarrow SR1 + SR2$	*
ADD DR,SR1,Imm5	$DR \leftarrow SR1 + \text{SEXT}(Imm5)$	*
AND DR,SR1,SR2	$DR \leftarrow SR1 \text{ and } SR2$	*
AND DR,SR1,Imm5	$DR \leftarrow SR1 \text{ and } \text{SEXT}(Imm5)$	*
LEA DR,label	$DR \leftarrow PC + \text{SEXT}(PCoffset9)$	*
LD DR,label	$DR \leftarrow \text{mem}[PC + \text{SEXT}(PCoffset9)]$	*
ST SR,label	$\text{mem}[PC + \text{SEXT}(PCoffset9)] \leftarrow SR$	
LDR DR,BaseR,Offset6	$DR \leftarrow \text{mem}[BaseR + \text{SEXT}(Offset6)]$	*
STR SR,BaseR,Offset6	$\text{mem}[BaseR + \text{SEXT}(Offset6)] \leftarrow SR$	
BR[n][z][p] label	Si (cond) $PC \leftarrow PC + \text{SEXT}(PCoffset9)$	
NOP	No Operation	
RET	$PC \leftarrow R7$	
JSR label	$R7 \leftarrow PC; PC \leftarrow PC + \text{SEXT}(PCoffset11)$	

instruction	macro	description
TRAP x00	HALT	termine un programme (rend la main à l'OS)
TRAP x20	GETC	lit au clavier un caractère ASCII et le place dans R0
TRAP x21	OUT	écrit à l'écran le caractère ASCII placé dans R0
TRAP x22	PUTS	écrit à l'écran la chaîne de caractères pointée par R0

On considère le programme en langage d'assemblage du LC3 ci-dessous. Ce programme est incomplet ; vous devrez donner le code de la routine **aff12** sur votre copie.

```
.ORIG x3000
LD R6,spinit      ; on initialise le pointeur de pile
LEA R0,chaîne      ; chargement dans R0 de l'adresse de la chaîne
JSR aff12          ; appel de la routine aff12
HALT               ; rend la main à l'OS
chaîne: .STRINGZ "hteultluo"
```

```
aff12:
; Affiche les caractères de rangs paires d'une chaîne.
; Entrée : R0, adresse du début de la chaîne à afficher.
; *** A COMPLETER ***
RET

; Gestion de la pile
spinit: .FILL stack
        .BLKW #16
stack:  .BLKW #1      ; fond de la pile
        .END
```

La routine **aff12** doit afficher la chaîne de caractères dont l'adresse de début lui est passée via **R0**, mais elle n'affiche qu'un caractère sur deux en commençant par le premier de la chaîne. Par exemple, avec la chaîne **hteultluo**, la routine doit afficher **hello**.

Q. 5 (1 pts). Pourquoi est-il important de commencer par sauvegarder l'adresse de retour (contenue dans **R7**) sur la pile en début de routine, puis de la restaurer en fin de routine ?

Q. 6 (2 pts). Proposez un pseudo-code pour le corps de la routine **aff12**. Vous pouvez utiliser une boucle **while**, pour charger à chaque tour de boucle deux caractères successifs de la chaîne ; attention, il faut vérifier que les deux sont non nuls. Vous pouvez utiliser une instruction **break** pour sortir en avance de la boucle.

Q. 7 (4 pts). Donnez le code de la routine en langage d'assemblage du LC3. Veillez à ajouter quelques commentaires, de façon à ce que votre correcteur puisse faire le parallèle avec la question précédente.

Exercice 3 (Programme mystère — 7 pts). On considère un programme en langage machine du LC3 : **x3000 : 5020 122F 0403 1001 127F 0FFC 3201 F025 0000**. On n'est pas avancé avec ça... Dans cet exercice, vous devez traduire ce programme en langage d'assemblage, pour comprendre ce qu'il fait. Vous disposez du tableau suivant.

syntaxe	action	codage																
		opcode				arguments												
		F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
NOT DR,SR	DR ← not SR	1 0 0 1				DR		SR		1 1 1 1 1 1								
ADD DR,SR1,SR2	DR ← SR1 + SR2	0 0 0 1				DR		SR1		0	0 0		SR2					
ADD DR,SR1,Imm5	DR ← SR1 + SEXT(Imm5)	0 0 0 1				DR		SR1		1	Imm5							
AND DR,SR1,SR2	DR ← SR1 and SR2	0 1 0 1				DR		SR1		0	0 0		SR2					
AND DR,SR1,Imm5	DR ← SR1 and SEXT(Imm5)	0 1 0 1				DR		SR1		1	Imm5							
LD DR,label	DR ← mem[PC + SEXT(PCoffset9)]	0 0 1 0				DR		PCoffset9										
ST SR,label	mem[PC + SEXT(PCoffset9)] ← SR	0 0 1 1				SR		PCoffset9										
BR[n][z][p] label	Si (cond) PC ← PC + SEXT(PCoffset9)	0 0 0 0				n	z	p	PCoffset9									
NOP	No OPeration	0 0 0 0				0	0	0	0 0 0 0 0 0 0 0 0									

On rappelle que dans ce tableau :

- Imm5 désigne un entier immédiat codé en complément à 2 sur 5 bits ;
- PCOffset9 désigne un entier immédiat codée en complément à 2 sur 9 bits ;
- SEXT est l’extention de signe pour obtenir un entier en complément à 2 sur 16 bits ;
- Si AdI est l’adresse d’une instruction faisant référence à l’adresse $PC + \text{SEXT}(\text{PCOffset9})$, alors il faut remplacer PC par AdI+1 dans le calcul d’adresse : autrement dit, $PC + \text{SEXT}(\text{PCOffset9})$ peut être remplacé par AdI+1+SEXT(PCOffset9).

Q. 8. (4 pts) Complétez un tableau comme celui donné ci-dessous. Dans la colonne « instruction binaire », convertissez les instructions de l’hexadécimal vers le binaire, et délimitez les différents champs que comporte chaque instruction. Dans la colonne « détails de traduction », notez les détails relatifs à la traduction, notamment les calculs d’adresses. Dans la colonne, « instruction assembleur, » notez la traduction en langage d’assemblage de chaque instruction. Notez juste les adresses pour l’instant (pour les instructions ST, LD ou BR).

adresse hexadécimal	instruction hexadécimal	instruction binaire	détails de traduction	instruction assembleur
3000	5020			
3001	122F			
3002	0403			
3003	1001			
3004	127F			
3005	0FFC			
3006	3201			
3007	F025			HALT
3008	0000			

Q. 9. (1 pt) Donnez proprement le code du programme, en langage d’assemblage du LC3, en introduisant des labels à la place des adresses.

Q. 10. (2 pts) Proposez un pseudo-code pour le programme que vous avez traduit : que fait ce programme ?