EXAMEN ARC TCA 2019 **3TCA**

Examen ARC 3TCA 2019, Examen du 11/10/2019

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

⟵ codez votre numéro d'étudiant ci-contre, et inscrivez votre nom et prénom ci-dessous. ↓

Nom et prénom :

Durée : 2 heures

Tout document papier autorisé. L'usage de la calculatrice est interdit. Cochez les réponses des question QCM au crayon à papier pour éviter les ratures. Pour les autres questions, le cadre donne une idée de la taille des réponses attendues. **Ne pas écrire** *dans les case grisées marquées* ***"Réservé"***

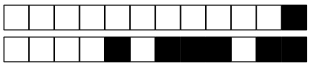
Les questions faisant apparaître le symbole ♣ peuvent présenter zéro, une ou plusieurs bonnes réponses (pour ces questions, les mauvaises réponses peuvent entrainer des points négatifs). Les autres ont une unique bonne réponse.

L'enseignant responsable ne répond pas aux questions qui concernent la mauvaise compréhension d'une question (pour que tout le monde soit dans les mêmes conditions), il répond uniquement aux questions qui concernent les éventuelles erreurs dans le sujet.

IMPORTANT*: n'oubliez pas d'indiquer votre numéro d'étudiant et votre nom ci-dessus.*

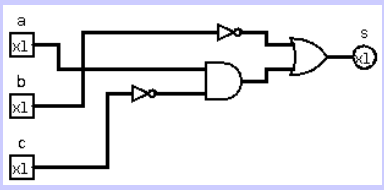
Attention Version avec Solution

Calcul Booleen et Circuit (5pts)



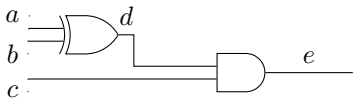
Question 1 Dessinez un circuit à 3 entrées (a, b et c) et une sortie (out) ayant pour table de vérité la table suivante:

a	b	c	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

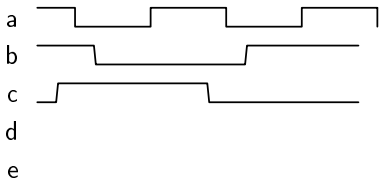


☐f ☐p ☐j *Reservé*

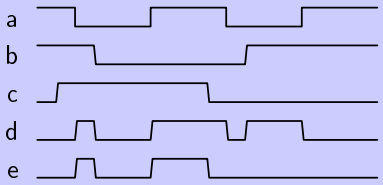
Question 2 On considère le circuit suivant.



Complétez le chronogramme ci-dessous avec les valeurs des signaux d et e.



Quelle est l'équation logique qui définit e en fonction de a, b et c? (répondez dans le cadre ci-dessous).



☐f ☐p ☐j *Reservé*

Question 3 Que vaut $0xDE3A$ en binaire?

- ☐ 1101 1110 0011 0110
- ☐ 1101 1100 0110 1100
- ☐ 56790
- ☐ 1101 1110 0011 1010

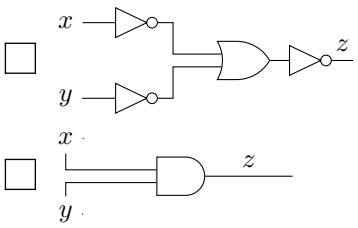
Question 4
L'expression booleenne suivante $(\bar{A} + \bar{B}).(\bar{A} + B)$ se simplifie en

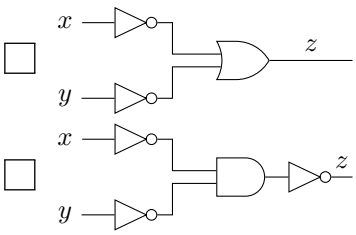
- ☐ $\bar{A}$
- ☐ $\bar{A} + \bar{B}$
- ☐ $\bar{B}$
- ☐ $\overline{A + B}$

Question 5 Que vaut $(163)_{10}$ en base 2 sur 8 bits

- ☐ 1010011
- ☐ 10100111
- ☐ 10100011
- ☐ 10100101

Question 6 ♣ Parmis les circuits suivants, quels sont ceux qui implémentent la formule logique suivante: $z = x.y$?

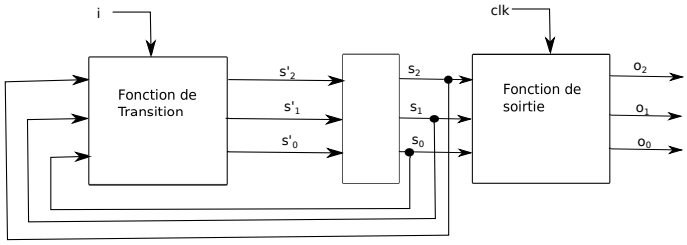




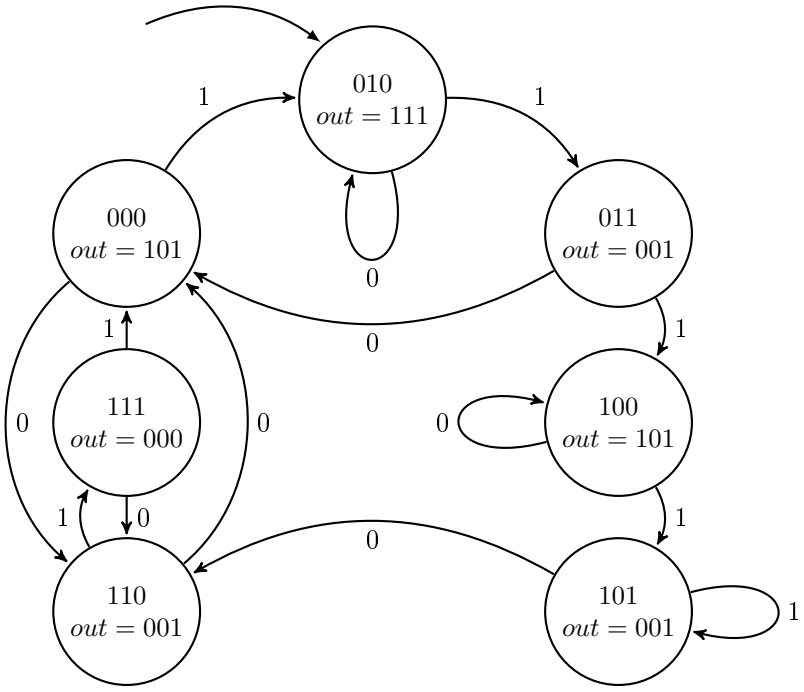
☐ *Aucune de ces réponses n'est correcte.*

Automates (8pts)

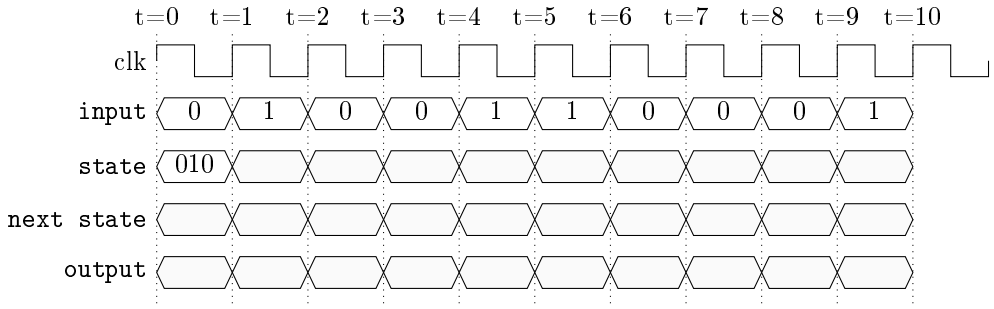
Le circuit séquentiel ci-dessous est un automate qui présente une entrée i sur 1 bit, ainsi qu'une sortie $o = (o_2, o_1, o_0)$ sur 3 bits. L'état courant (s_2, s_1, s_0) comporte trois bits, stockés dans un registre composé de trois flip-flops régies par le front montant de l'horloge *clk*.

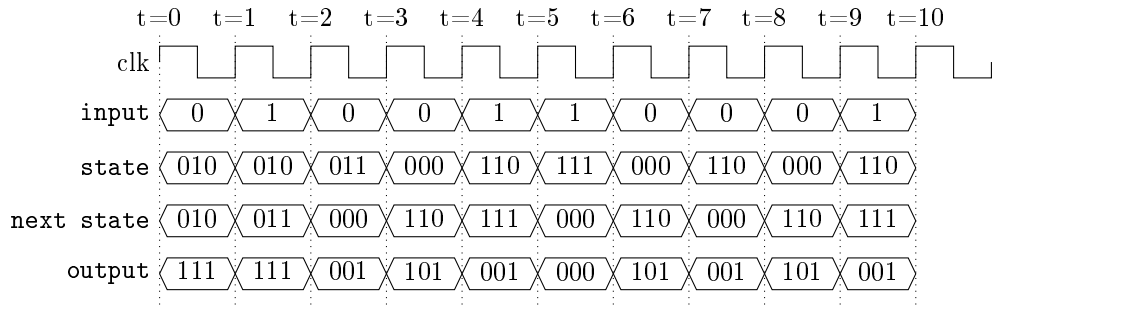


L'automate décrivant le comportement est donné ci-dessous, dans chaque état est décrit la sortie o correspondant à cet état et les transitions sont étiquetées par la valeur de l'entrée i qui les provoque. Par exemple, l'état initial est l'état 010, lorsque l'entrée i vaut 1, on passe sur le front montant de l'horloge *clk* dans l'état 011.



Question 7 Assurez-vous que vous comprenez bien le comportement de ce circuit. Pour cela, complétez le chronogramme ci-dessous.





☐f ☐p ☐j *Reservé*

Question 8 Remplissez les tables de vérités ci-dessous, correspondant à la fonction de transition et la fonction de sortie de cet automate.

Fonction de transition:

s_2	s_1	s_0	i	s'_2	s'_1	s'_0
0	0	0	0			
0	0	0	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			

Fonction de sortie:

s_2	s_1	s_0	o_2	o_1	o_0
0	0	0			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

☐f ☐p ☐j *Reservé*

Transition:

s_2	s_1	s_0	i	s'_2	s'_1	s'_0
0	0	0	0	1	1	0
0	0	0	1	0	1	0
0	1	0	0	0	1	0
0	1	0	1	0	1	1
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	1	0
1	0	1	1	1	0	1
1	1	0	0	0	0	0
1	1	0	1	1	1	1
1	1	1	0	1	1	0
1	1	1	1	0	0	0

Sortie:

s_2	s_1	s_0	o_2	o_1	o_0
0	0	0	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	0	0	1
1	1	1	0	0	0

Question 9

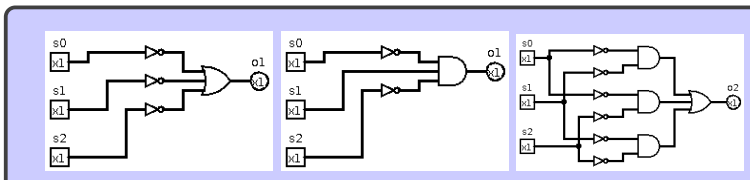
Fournissez une fonction booléenne pour chacun des signaux de sortie: o_2, o_1, o_0. Minimisez si vous pouvez.

☐ff ☐f ☐p ☐j ☐jj *Reservé*

$o_0 = \overline{s_0} + \overline{s_1} + \overline{s_2}$
$o_1 = \overline{s_0}.s_1.\overline{s_2}$
$o_2 = \overline{s_0 s_1} + \overline{s_0 s_2} + \overline{s_1 s_2}$

Question 10 Implémentez maintenant la fonction de sortie en un circuit composé de portes logiques, correspondant aux fonctions booléennes que vous venez de définir pour o_2, o_1 et o_0.

☐f ☐p ☐j *Reservé*



Micro-Architecture et assembleur (7pts)

Question 11 Un MSP430 exécute le code de la colonne de gauche du second tableau ci-dessous. On rappelle les définitions des instructions concernées:

Mnemonic	description	operation
MOV(.B) src, dst	Move source to destination	src → dst
PUSH(.B) src	Push source on stack	SP-2 → SP, src → @SP
RET	Return from subroutine	@SP → PC, SP+2 → SP

Pour chaque ligne, remplissez la table ci-dessous avec les valeurs des registres et de la mémoire *après* l'exécution de l'instruction correspondante (vous pouvez laisser en blanc les cases inchangées).

Instruction	Registres				Cases mémoires				
	PC	SP	R11	R12	0	2	4	6	8
(état initial)	0x0400	8	4	2	0	0	0	0	0
0x0400 `MOV #6, SP`									
0x0404 `PUSH R12`									
0x0406 `PUSH R11`									
0x0408 `RET`									

Instruction	Registres				Cases mémoires				
	PC	SP	R11	R12	0	2	4	6	8
(état initial)	0x0400	8	4	2	0	0	0	0	0
0x0400 `MOV #6, SP`	0x0404	6							
0x0404 `PUSH R12`	0x0406	4					2		
0x0406 `PUSH R11`	0x0408	2				4	2		
0x0408 `RET`	0x4	4							

☐ff ☐f ☐p ☐j ☐jj *Reservé*

Question 12 Ecrivez une portion de code assembleur MSP430 qui travaille sur les registres R5 et R6 et qui fait comme si R6 et R5 *comptaient* les heures et les minutes, avec 2000 minutes par heures et 400 heures par jour. soit l'équivalent du code suivant:

```
R6=0;
for i in range(400):
    R6=R6+1;
    R5=0;
    for j in range(2000):
        R5=R5+1;
#ici R5 vaut 2000, et R6 vaut 400
```

```
.section .init9
main:
        mov.b #1, &34
        mov.b #1, &33


        mov #0, R6
loop:
        inc R6
        mov #0, R5
loop2:
        inc R5
        cmp #2000, R5
        jnz loop2
        cmp #400, R6
```

```
        jnz loop
        mov.b #0, &34
wait:
        jmp wait
```

☐ff ☐f ☐p ☐j ☐jj *Reservé*

Question 13 Un MIPS exécute le code de la colonne de gauche du second tableau ci-dessous. On rappelle les définitions des instructions concernées:

Mnemonic		description	operation
li	$\$R_{dest}, cst$	Load immediate	$\$R_{dest} \leftarrow cst$
sw	$\$R_{source}, offset(\$R_{dest})$	Store word	$Memory[\$R_{dest} + offset] \leftarrow \$R_{source}$
bne	$\$R_s, \$R_t, label$	Branch not equal	IF $\$R_s \neq \$R_t$, $\$PC = label$
beq	$\$R_s, \$R_t, label$	Branch if equal	IF $\$R_s = \$R_t$, $\$PC = label$
addi	$\$R_d, \$R_s, const16$	Add immediate	$\$R_s = \$R_t + const16$

Pour chaque ligne, remplissez la table avec les valeurs des registres et de la mémoire *après* l'exécution de l'instruction correspondante (vous pouvez laisser en blanc les cases inchangées).

Instruction	Registres			Cases mémoires				
	$t2	$t3	$t4	0	4	8	12	16
(état initial)	16	15	0	0	0	4	4	4
li $t4 16								
Label1: bne $t2, $t4 Label2								
sw $t2 0($t4)								
Label2: addi $t3 $t3 1								
sw $t3 -4($t3)								

Instruction	Registres			Cases mémoires				
	$t2	$t3	$t4	0	4	8	12	16
(état initial)	16	15	0	0	0	4	4	4
li $t4 16			16					
Label1: bne $t2, $t4 Label2								
sw $t2 0($t4)								16
Label2: addi $t3 $t3 1		16						
sw $t3 -4($t3)							16	

☐ff ☐f ☐p ☐j ☐jj *Reservé*

Question 14 ♣
Dans le TD7 (Simulateur SPIM du MIPS), on exécutait le code assembleur MIPS suivant. Dans ce contexte, quelles sont les phrases vraies ci-dessous?

```
[...]
lw   $s0, x
lw   $s1, y
move  $a0, $s0
jal  fun
move  $s1,$v0
[...]
```

- ☐ Les opérations effectuées ne modifient que des registres.
- ☐ Les registres référencés dans ce code sont dans le CPU
- ☐ `$v0` contient le résultat de l'appel à la fonction `fun`.
- ☐ Les valeurs de `x` et `y` sont des addresses mémoires.
- ☐ L'instruction `lw` transfère une valeur de la mémoire à un registre.
- ☐ `jal fun` est un saut conditionnel.
- ☐ L'instruction `move $a0,$s0` met le contenu de $a0 dans $s0.
- ☐ `x` et `y` sont inconnues à la compilation.
- ☐ `fun` est un label qui correspond à une adresse mémoire.
- ☐ L'exécution de `jal fun` modifie le registre $v0 en y stockant l'addresse de retour.
- ☐ *Aucune de ces réponses n'est correcte.*

Question 15 ♣ Supposons que l'on exécute le code suivant sur un MSP430. Dans ce contexte, quelles sont les phrases vraies ci-dessous?

```
main:
        mov.b #1, &34
        mov.b #0, &33
        mov.b #1, &33
        mov.b #2, &33
loop:   jmp loop
```

- ☐ L'écriture de 1 à l'adresse 33 allume la LED rouge.
- ☐ Ce code influence le comportement du PORT2.
- ☐ Ce code fait clignoter la LED rouge.
- ☐ La dernière ligne du code représente une boucle infinie.
- ☐ L'écriture à l'adresse 34 allume la LED rouge.
- ☐ L'écriture de 2 à l'adresse 33 allume la LED verte et la LED rouge.
- ☐ L'écriture de 2 à l'adresse 33 allume la LED verte.
- ☐ Ce code influence le comportement uniquement du premier bit du PORT1.
- ☐ Il s'agit d'adresses exprimées en binaire.
- ☐ Pour allumer les deux LEDs (rouge et verte), il faut écrire `#3` aux adresses `33` puis `34`.
- ☐ Sur l'ez430 les deux LEDs sont branchées sur le même port.
- ☐ Tous les 8 bits du PORT 1 sont positionnés en "OUT".
- ☐ *Aucune de ces réponses n'est correcte.*

Question 16 Considérons le code assembleur que vous avez écrit (ou que vous allez écrire) à la question 12, en supposant que l'horloge du MSP430 tourne à 4MHz, qu'une instruction assembleur MSP430 est (environ) exécutée tous les deux cycles d'horloge. Donner (et expliquer) le temps approximatif au bout duquel les registres arrivent à la valeur max (i.e. 2000 pour R5 et 400 pour R6)

☐f ☐p ☐j *Reservé*