Circuits Numériques et Éléments d'Architecture
Examen

ENSIMAG 1A

Année scolaire 2016–2017

— durée : 3h ;
— résumé sur feuille A4 manuscrite et calculatrices autorisés ;
— le barème est donné à titre indicatif ;
— les exercices sont indépendants et peuvent être traités dans le désordre, et certaines questions au sein du même exercice peuvent aussi être traitées indépendamment ;
— pensez à indiquer votre **nom** et **numéro** de groupe sur chacune de vos copies.
— l'annexe est à rendre avec la copie car elle vous permet de répondre à certaines questions du dernier exercice. N'oubliez pas d'y ajouter votre nom.

Ex. 1 : Questions de cours (2.5 pts)

Question 1 Dessinez le schéma d'un opérateur qui réalise le ou-exclusif ($\oplus$) bit à bit entre deux entrées a et b sur 4 bits et produit une sortie s sur 4 bits.

Question 2 À quoi sert un élément séquentiel ? (une seule réponse)
(a) à calculer la valeur d'une fonction booléenne
(b) à mémoriser la valeur d'un signal

Question 3 Soit un automate comprenant 10 états. Indiquer le nombre de bascules D nécessaires à la réalisation de l'automate : (répondre pour les deux cas)
(a) en codage logarithmique
(b) en codage 1 parmi n

Question 4 Soit un cache de 4k octets possédant 64 lignes, sachant que, classiquement, les mots sont constitués de 4 octets.
Combien de mots contient une ligne de cache ?
Sur combien de bits sont codés (a) le champ *offset*, (b) le champ *index*, (c) le champs *tag* de l'adresse (qui est sur 32 bits) ?

Ex. 2 : Circuits combinatoire (2.5 pts)

On cherche à faire un circuit combinatoire qui indique sur sa sortie s_0 si un nombre compris entre 0 et 13 est premier, et sur sa sortie s_1 s'il est strictement positif et n'est pas un multiple de 3. La table de vérité correspondante est donnée ci-dessous.

n	s_1	s_0
0	1	0
1	1	0
2	1	1
3	0	1
4	1	0
5	1	1
6	0	0
7	1	1
8	1	0
9	0	0
10	1	0
11	1	1
12	0	0
13	1	1

Question 1 Sur combien de bits est codée l'entrée n ? On notera e_i le bit en position i de l'entrée.

Question 2 En tenant compte du fait que la fonction est incomplète et en utilisant des tables de Karnaugh, donnez les expressions simplifiées des s_i sous forme canonique (sommes de produits).

Ex. 3 : Synthèse d'automate (4 pts)

On désire créer un automate qui produise la séquence des nombres de 0 à 7 sous forme de code *Gray*, rappelée ci-dessous :

```
n     Code
0     000
1     001
2     011
3     010
4     110
5     111
6     101
7     100
```

L'automate est doté d'une entrée e sur 1 bit. Lorsque $e = 1$, on doit passer au code suivant, selon le tableau ci-dessus. Quant $e = 0$, on doit passer au code correspondant au nombre courant multiplié par 2, modulo 8

Question 1 Construisez le graphe de transitions possédant 8 états (numérotés de 0 à 7) permettant de décrire ce comportement.

On définit comme codage de l'état la valeur du code Gray correspondant au numéro de l'état (ainsi par exemple l'état 5 sera codé `111`). On note q_i les bits du registre contenant l'état.

Question 2 Donnez la table de transition exprimant les d_i du registre d'état et les sorties s_i en fonction des q_i et de e. *On ne demande pas les expressions simplifiées des différents signaux.*

Ex. 4 : Conception PC/PO d'un circuit de chiffrement symétrique (5 pts)

L'algorithme *Tiny Encryption Algorithm* est un algorithme de chiffrement par bloc [1] très simple qui peut être utilisé dans les objets connectés pour lesquels un niveau de sécurité élevé n'est pas une obligation.

— on ne cherchera pas à comprendre ce que fait cet algorithme, on le prendra donc comme une entrée de l'exercice sans plus se préoccuper de sa fonction ;

1. Du à David Wheeler et Roger Needham, du laboratoire d'informatique de Cambridge, UK.

— l'opérateur $v >> p$ décale le mot binaire v à droite de p positions, les p bits de poids faible étant perdus et p '0' étant injectés à gauche ; il s'agit donc d'un décalage *logique* à droite ;
— l'opérateur $v << p$ décale le mot binaire v à gauche de p positions, les p bits de poids fort étant perdus et p '0' étant injectés à droite ; il s'agit donc d'un décalage à gauche ;
— l'opérateur $a \oplus b$ effectue le « ou exclusif » bit à bit entre les mots binaires a et b.

1 – registres initialisées avant le lancement de l'algorithme
2 – la clef, sur 4 mots de 32 bits
3 $k_0 \leftarrow 0x8badf00d, k_1 \leftarrow 0xdeadbeef, k_2 \leftarrow 0xd15ea5e, k_3 \leftarrow 0xbada55$
4 – une constante magique
5 $\delta \leftarrow 0x9e3779b9$
6 Get(v_0) – lecture du premier mot de 32 bits à chiffrer
7 Get(v_1) – lecture du second mot de 32 bits à chiffrer
8 $sum \leftarrow 0$
9 **for** $i \leftarrow 0$ **to** 31 **step** 1 **do**
10 $\quad v_1 \leftarrow v_1 - ((v_0 << 4) + k_2) \oplus (v_0 + sum) \oplus ((v_0 >> 5) + k_3)$
11 $\quad v_0 \leftarrow v_0 - ((v_1 << 4) + k_0) \oplus (v_1 + sum) \oplus ((v_1 >> 5) + k_1)$
12 $\quad sum \leftarrow sum - \delta$
13 **end for**
14 Put(v_0) – émission du premier résultat chiffré
15 Put(v_1) – émission du second résultat chiffré

Question 1 La variable sum doit-elle être mise en registre ? Justifiez ce choix.

Question 2 Y-a-t-il des opérations pour lesquelles l'utilisation de circuits logiques n'est pas nécessaire dans cet algorithme ? Précisez les lignes et les opérateurs.

Question 3 On ne cherche pas à paralléliser l'exécution des lignes 10, 11, et 12, on fait donc l'hypothèse que l'on veut exécuter *chacune* les lignes 10, 11, et 12 en 1 cycle. Donnez le nombre et la nature des opérateurs strictement nécessaires.

Question 4 Proposez une partie opérative permettant d'implanter cet algorithme, toujours avec l'hypothèse que l'on passe 1 cycle pour exécuter la ligne 10, un cycle pour exécuter la ligne 11 et un cycle pour exécuter la ligne 12.

Question 5 Proposez un automate d'états qui pilote cette partie opérative sous forme de graphe d'état (au sein de chaque état on précisera, le nom et la description RTL). Spécifiez la valeur des différents signaux de commande des registres et des multiplexeurs pour chaque état dans un tableau comme fait en TD.

Question 6 Peut-on réécrire sous forme d'affectation concurrent les lignes 10 et 11 ? Justifiez votre réponse.

Ex. 5 : Conception de processeur (6 pts)

L'objectif de cet exercice est d'ajouter trois nouvelles instructions au processeur 2-adresses étudié durant les TD 9, 10 et 11. Pour mémoire, le jeu d'instructions, la partie opérative ainsi que la partie contrôle (sans signaux et conditions) sont rappelés en annexe.

La première instruction est une addition avec retenue entrante (de 1 bit bien sûr), notée `adc`, très répandue dans les processeurs 8 bits pour pouvoir faire des opérations sur 16 bits.
L'instruction se note `adc rs, rd` et effectue donc le calcul $rd \leftarrow rd + rs + carry$.
La seconde instruction permet de faire un reset de la retenue, notée `clc`, pour *clear carry*. Elle exécute $carry \leftarrow 0$.
La troisième instruction permet de faire un chargement avec une indirection mémoire. L'instruction se note `ldi rd, addr` (pour *load indirect*) et exécute $rd \leftarrow mem[mem[addr]]$.

Question 1 Proposez un encodage des instructions `adc`, `clc` et `ldi`.

Question 2 On s'intéresse tout d'abord aux instructions `adc` et `clc`. On suppose que l'on dispose dans l'UAL d'une entrée de retenue c_{in} et d'une sortie de retenue c_{out}. Est-il suffisant d'ajouter uniquement ces deux fils pour pouvoir effectuer l'opération correspondant à `adc` ? Justifiez en quelques mots votre réponse, et ajoutez dans la partie opérative (à compléter sur l'annexe à rendre avec votre copie) le ou les registres nécessaires, ainsi que les multiplexeurs et les fils permettant de réaliser cette instruction.

Question 3 Ajoutez dans la partie contrôle (à compléter sur l'annexe et à rendre avec votre copie) les états nécessaires à l'exécution de l'instruction `adc` en précisant la valeur des différents signaux dans un tableau comme en TD.

Question 4 Que faut-il ajouter dans la partie opérative pour supporter l'opération `clc` ? Modifier l'automate d'états pour prendre en compte cette instruction.

Question 5 On s'intéresse maintenant à l'instruction `ldi`. Quel matériel faut-il ajouter et/ou modifier dans la partie opérative pour pouvoir supporter cette indirection supplémentaire lors de la lecture de la donnée ? (Complétez sur l'annexe à rendre avec votre copie)

Question 6 Précisez également les modifications à apporter à la partie contrôle, en décrivant les états ajoutés et les actions effectuées sur la partie opérative dans chaque état. (Complétez sur l'annexe à rendre avec votre copie)

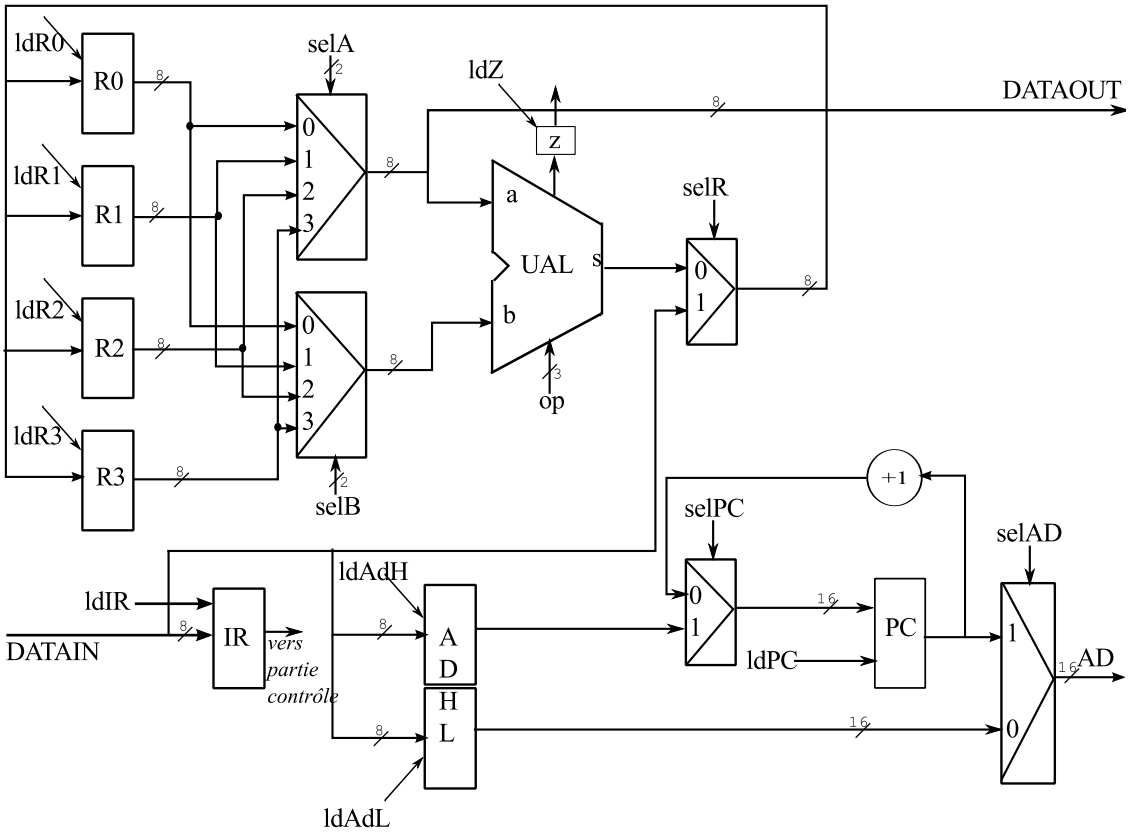
Annexe

NOM : PRENOM : GROUPE :

Jeu d'instructions et encodage

Instruction	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Opérations de la forme : $rd := rd\,op\,rs$								
`or rs, rd`	0	0	0	0	rs_1	rs_0	rd_1	rd_0
`xor rs, rd`	0	0	0	1	rs_1	rs_0	rd_1	rd_0
`and rs, rd`	0	0	1	0	rs_1	rs_0	rd_1	rd_0
`add rs, rd`	0	1	0	0	rs_1	rs_0	rd_1	rd_0
`sub rs, rd`	0	1	0	1	rs_1	rs_0	rd_1	rd_0
Opérations de la forme : $rd := op\,rd$								
`not rd`	0	0	1	1	0	0	rd_1	rd_0
`shl rd`	0	1	1	0	0	0	rd_1	rd_0
`shr rd`	0	1	1	1	0	0	rd_1	rd_0
Chargement : $rd := MEM(AD)$								
`ld AD, rd`	1	0	0	0	0	0	rd_1	rd_0
	ADH							
	ADL							
Stockage : $MEM(AD) := rs$								
`st rs, AD`	1	1	0	0	0	0	rs_1	rs_0
	ADH							
	ADL							
Branchement inconditionnel : $PC := AD$								
`jmp AD`	1	0	0	0	0	1	0	0
	ADH							
	ADL							
Branchements conditionnels : si Z=1 alors $PC = AD$								
`jz AD`	1	0	0	0	0	1	0	1
	ADH							
	ADL							

Partie opérative



Partie Contrôle