Nom, prénom, numéro d'étudiant :

CC-TP - LIF6 - lundi 14/12/15 - noté sur 20 - durée : 2h

Calculatrices, téléphones et ordinateurs personnels interdits. Aucun document autorisé.

Récupérez le fichier `auto2015_CC-TP.tar.gz`. Vous devrez renvoyer les fichiers `exo1.circ` et `saisie.asm` à votre chargé de TP. Quelques questions nécessitent des réponses rédigées, répondez au verso de cette feuille.

Exercice 1 (Circuits logiques dans Logisim). Le fichier `exo1.circ` contient un circuit qui envoie quatre entrées binaires x_3, x_2, x_1 et x_0 en entrée d'un afficheur hexadécimal, qui affiche $x = (x_3x_2x_1x_0)_{16}$. L'afficheur procède de la façon habituelle : il affiche `0`,...,`9` pour les chiffres plus petits que $(10)_{10}$, puis `A`,...,`F` pour les plus grands.

On souhaite modifier le circuit de façon à ce qu'il n'affiche que des chiffres décimaux. Plus précisément, si $x < (10)_{10}$, alors l'afficheur affiche le chiffre correspondant, mais si $x \geq (10)_{10}$, alors il affiche `E` pour « Erreur. »

Dans la suite de l'exercice, vous répondrez en complétant le fichier `exo1.circ`, en ajoutant dans `main` une nouvelle version du circuit à chaque question. Ajoutez quelques commentaires dans votre fichier pour aider votre correcteur à suivre votre cheminement !

1. Commencez par produire une version du circuit demandée dans laquelle vous utiliserez un comparateur d'entiers proposé dans la bibliothèque de Logisim (section `Arithmetic`) pour comparer x à $(10)_{10}$.
2. Ajoutez un sous-circuit `comp10a`, afin de comparer une entrée x sur 4 bits à $(10)_{10}$: une sortie s produira 1 si $x \geq (10)_{10}$, 0 sinon. Dans ce sous-circuit, utilisez un comparateur d'entiers de Logisim. Vous créerez une nouvelle version du circuit d'affichage utilisant `comp10a` dans `main`.
3. Ajoutez un sous-circuit `comp10b`, équivalent à `comp10a`, mais dans lequel les seules portes utilisées sont des portes `AND`, `OR` et `NOT`. Établissez la table de vérité de `comp10b`, et déduisez-en une expression boléenne donnant s en fonction de x_3, x_2, x_1 et x_0 : attention, il faut réfléchir un peu pour ne pas se retrouver avec une formule difficile à exploiter... Justifiez en tout cas votre expression.
4. Créez une nouvelle version du circuit d'affichage utilisant `comp10b` dans `main`. Cette fois-ci, utilisez un compteur 4 bits fourni par Logisim (section `Memory`) pour automatiser la validation de votre circuit.

Exercice 2 (Programmation LC3 avec Pennsim). Vous compléterez progressivement le fichier `saisie.asm`.

1. La routine `saisie` permet de lire un entier naturel en base 10 au clavier, et place l'entier lu dans `R1`. Modifiez-la pour qu'elle affiche le message « `Entrez un entier naturel :` » avant d'effectuer la saisie.
2. Complétez la routine `aff` de façon à ce qu'elle affiche autant d'étoiles « `*` » que l'entier naturel contenu dans `R1` lors de son appel. Après avoir affiché `R1` étoiles, la routine doit aussi afficher un retour à la ligne. Suivez les consignes données en commentaire dans le fichier.
3. Modifiez le programme principal `main` pour qu'il effectue la lecture d'un entier au clavier avec `saisie`, puis son affichage avec `aff`.
4. La routine `mul10` fournie permet de multiplier par 10 le contenu de `R1`. Mais, telle qu'elle vous est fournie, elle exécute 10 instructions : modifiez cette routine de façon à ce qu'elle exécute au plus 6 instructions, tout en calculant toujours le même résultat.

On rappelle ci-dessous les instructions du LC3 que vous pourrez utiliser :

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

Rappel de quelques directives d'assemblage :

`.FILL valeur`	Réserve un mot de 16 bits et le remplit avec la valeur constante donnée en paramètre.
`.BLKW nombre`	Cette directive réserve le nombre de mots de 16 bits passé en paramètre.
`.STRINGZ "chaine"`	Place en mémoire le chaîne de caractères `chaine`.
`;`	Les commentaires commencent par un point-virgule.

On rappelle aussi les quelques macros suivantes :

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