Instructions

- Durée 1 heure.
- Lisez bien toutes les questions.
- **Tout votre code sera déposé dans deux fichiers sur TOMUSS**.
- Tout document interdit, pas de navigateur web, sauf TOMUSS.

Exercice 1 – Circuit combinatoire (travail à réaliser avec LogiSim)

Nous nous proposons de réaliser ici un additionneur 10 bits à *sélection* de retenue.
Les circuits sont à réaliser avec des portes NAND et des portes NOT uniquement.
Vous déposerez votre fichier .circ dans la case TOMUSS TPN/rendu_circ.

1. Proposez un circuit réalisant un multiplexeur comportant une entrée de contrôle 1 bit, deux entrées de données 1 bit et une sortie 1 bit.
2. Proposez un circuit réalisant un multiplexeur comportant une entrée de contrôle 1 bit, deux entrées de données 5 bits et une sortie 5 bits.
3. Proposez un circuit réalisant un full-adder, c'est-à-dire un circuit prenant en **entrées** deux bits a_1 et b_1, une retenue entrante r_0 et ayant en **sorties** un bit s_1 représentant la somme de a_1 et $b_1$ainsi qu'un bit représentant la retenue générée r_1.
4. Proposez un circuit réalisant, à l'aide des full-adders que vous avez réalisé à la question précédente, la somme de deux entiers codés sur 10 bits. De combien de traversées de portes NAND avons-nous besoin pour obtenir la retenue sortante finale ? (Indiquez votre réponse sous le circuit).
5. Le principe de la sélection de retenue est d'effectuer, en parallèle de la somme des 5 bits de poids faible, deux calculs simultanés des 5 bits de poids fort, l'un supposant que la retenue générée par les poids faibles est 1, l'autre supposant que la retenue générée par les bits de poids faible est 0. Lorsque la retenue générée par le calcul sur les bits de poids faible est connue, il suffit de *sélectionner* le bon résultat pour les bits de poids fort, ainsi que la bonne retenue générée par les poids forts.
   Proposez un circuit réalisant un additionneur pour deux entiers de 10 bits selon ce principe. De combien de traversées de portes NAND avons-nous besoin pour obtenir la retenue sortante finale dans ce circuit ? (Indiquez votre réponse sous le circuit).

Exercice 2 – Programmation en langage d'assemblage (travail à réaliser avec PennSim)

On dispose d'un tableau d'entiers signés représentés en complément à 2 sur 16 bits. Le tableau débute en mémoire à l'adresse désignée par le label `tabdeb` et termine à l'adresse désignée par le label `tabfin`.
Écrivez un programme en langage d'assemblage du LC-3 qui permet de trouver la valeur MIN et la valeur MAX de ce tableau. Les valeurs MIN et MAX seront rangées en mémoire aux labels `min` et `max`.

Récupérez dans TOMUSS le fichier .asm à compléter dans la case TPN_sujet_asm.

Vous déposerez votre fichier .asm complété dans la case TOMUSS TPN_rendu_asm.

Syntaxe	action	NZP	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				
LEA DR,label	DR <- PC + SEXT(PCoffset9)	*	1	1	1	0	DR			PCoffset9								
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								
LDR DR,BaseR,Offset6	DR <- mem[BaseR + SEXT(Offset6)]	*	0	1	1	0	DR			BaseR			Offset6					
STR SR,BaseR,Offset6	mem[BaseR + SEXT(Offset6)] <- SR		0	1	1	1	SR			BaseR			Offset6					
BR[n][z][p] label Si (cond)	PC <- PC + SEXT(PCoffset9)		0	0	0	0	n	z	p	PCoffset9								