TD N°1_Architecture
Durée conseillée : **45min**

Afin de finaliser l'étude du codage et du séquencement de notre machine je vous propose de traiter un ou les 2 petits exercices.
On rappelle la structure de la machine qui permet de réaliser ces exercices :

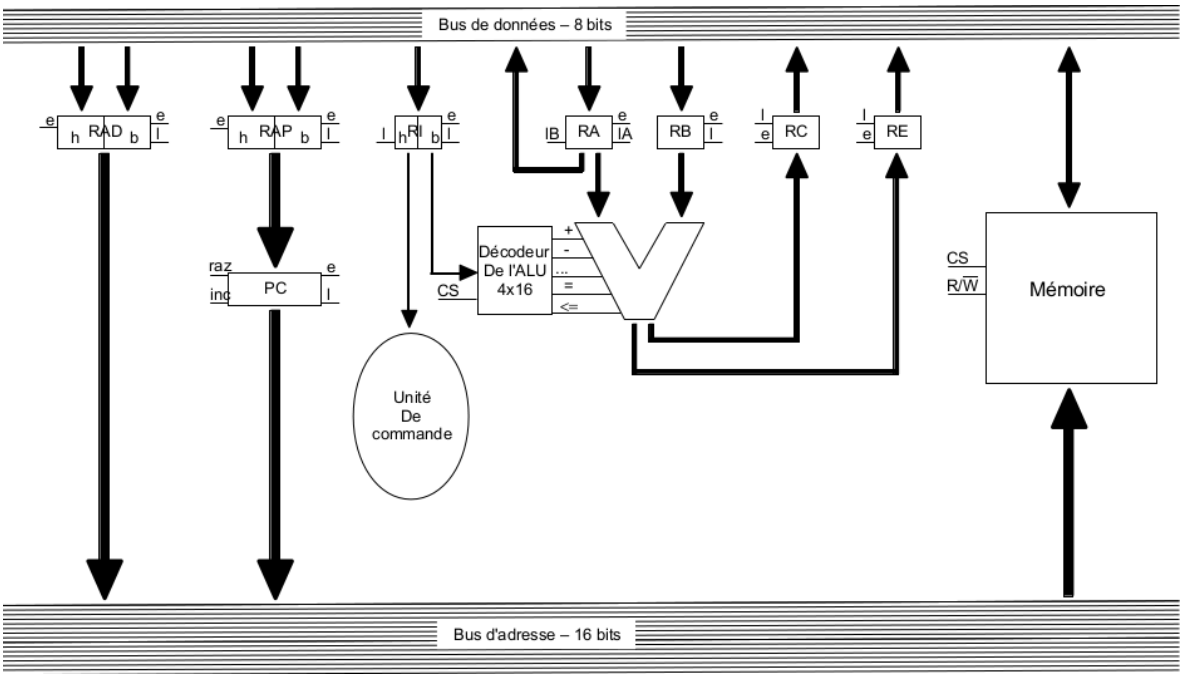


Tableau de codage des instructions et des opérations:

Instruction	Code
A := cte	1
A := B	2
A := B opa cte	3
A := B opa C	4
vers ET	5
si A opl cte vers ET	6
si A opl B vers ET	7

	Opérations arithmétiques				Opérations logiques					
Opération	+	-	*	/	=	≠	>	<	≥	≤
Code	2	3	4	5	6	7	8	9	A	B

Exercice N°1.

Soit le bout de programme suivant, les 3 premières instructions ont été décodées et certaines parties de la première instruction suivante ont été encadrées pour vous aider :

100 :	N := 30	120 :	07
103 :	S := 0	121 :	20
106 :	I := 0	122 :	17
109 : BOUCLE	78	123 :	32
110 :	17	124 :	17
111 :	07	125 :	07
112 :	20	126 :	03
113 :	16	127 :	17
114 :	01	128 :	07
115 :	32	129 :	50
116 :	42	130 :	01
117 :	20	131 :	09
118 :	17	132 : FIN	
119 :	17		

*Sachant que les variables **N**, **S** et **I** se trouvent respectivement aux adresses **2016**, **2017** et **1707** et que les 2 étiquettes **BOUCLE** et **FIN** sont placées respectivement dans le programme aux adresses **0109** et **0132***

*a) Décodez le reste du programme en décrivant clairement quelles instructions se trouvent à partir de l'adresse **109**.*

*b) Exécutez le programme et donner le contenu de **N**, **S** et **I** après l'exécution totale du programme.*

c) Dites en quelques mots ce que fait ce bout de programme.

d) Rappeler le rôle du registre RAD.

e) Donnez le séquencement de l'instruction **I := J + 12** *en commençant par le "fetch"*
On rappelle le codage de l'instruction **A:=B opa Cte***:*

CodeInst	CodeOp	Bh	Bb	Cte	Ah	Ab

Exercice N°2.

Cet exercice est dans le même esprit que le précédent. Vous pouvez ne répondre qu'aux questions a) b) et c) plus si affinité...

Soit le programme suivant:

0172		N := 27
0176		D := 5
0180		S :=0
0184	Boucle	79
0185		27
0186		34
0187		51
0188		20
0189		02
0190		07
0191		43
0192		27
0193		34
0194		51
0195		20
0196		27
0197		34
0198		32
0199		62
0200		42
0201		01
0202		62
0203		42
0204		50
0205		01
0206		84
0207	Fin	

*Sachant que les 3 variables **N**, **S**, **D** se trouvent respectivement aux adresses **2734**, **6242**, **5120** et sachant que les 2 étiquettes **Boucle** et **Fin** sont placées dans le programme aux adresses **0184**, **0207** respectivement :*

a) Décodez ce programme en décrivant clairement quelles instructions se trouvent à partir de l'adresse 0184 .

0184 :

b) Exécutez le programme et donnez l'évolution des valeurs de N, S et D pendant l'exécution du programme.

N : 27
S : 0
D : 5

c) *Dites en quelques mots ce que fait ce bout de programme.*

d) *Décrivez le rôle du registre RAP*

e) *Donnez le séquencement de l'instruction* **A:= B/2** *en commençant par le "fetch"*

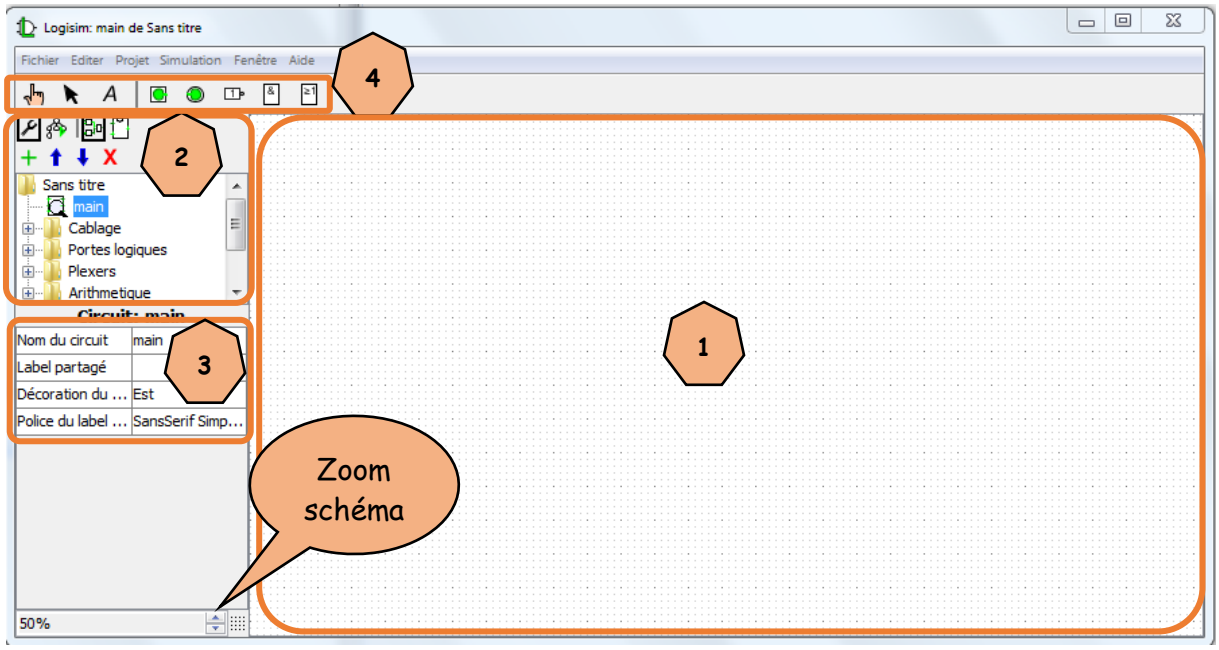
DIU
Enseigner l'Informatique au Lycée
Bloc 3 Architectures matérielles Spécialité NSI

TP N°1_Architecture
Prise en main rapide de Logisim 2.7.2
Réalisation d'une petite ALU8bits
Durée conseillée ***1h30***

A) Installation de Logisim 2.7.2 sur votre ordinateur... si ce n'est déjà fait.
Il existe diverses versions de Logisim qui ont suivi son arrêt de développement En octobre 2014. Je vous renvoie sur le lien suivant http://www.cburch.com/logisim/ pour télécharger la version "officielle" 2.7.1.
Une archive version 2.7.2 légèrement différente est téléchargeable sur: http://www.metz.supelec.fr/metz/personnel/fix_jer/Archi/Web/logisim-2.7.2.jar
Une fois que vous avez récupéré l'archive il faut vérifier que vous disposez d'une installation du java runtime environnement à jour.
Dans le répertoire où se trouve cette archive entrez la ligne de commande dans un terminal (windows et Linux): `java -jar logisim-2.7.2.jar`

Logisim s'ouvre alors. Par la suite(normalement) il vous suffira de cliquer sur `logisim-2.7.2.jar` *pour qu'il se lance.*

B) Environnement de développement.
Une fois le logiciel lancé, l'interface graphique de Logisim s'affiche. Elle comprend 4 zones principales :
1. *La surface de travail : zone d'édition du schéma à simuler,*
2. *Le panneau de navigation : Explorateur de projets, Bibliothèques de composants, Edition de sous-circuits (schémas hiérarchiques).*
3. *La table des attributs : Propriétés des circuits et des composants.*
4. *La barre d'outils rapide.*



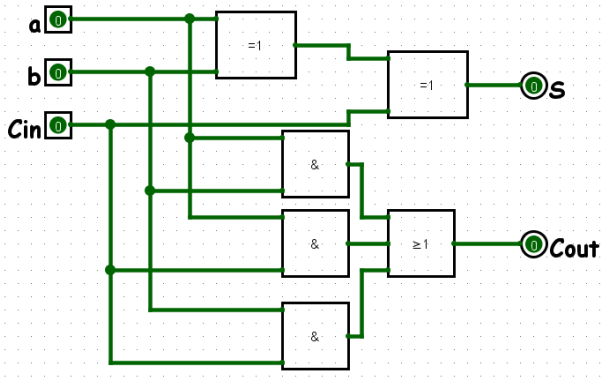
La conception de circuits avec Logisim se réalise en 2 étapes :
1- Edition du schéma (hiérarchique pour les plus complexes)
2- Simulation.

Travail demandé.
Edition et simulation de l'additionneur complet 1 bit vu en cours.
Ouvrir une session Logisim et enregistrer votre projet sous le nom **FullAdd**
L'extension **.cir** *est automatiquement associée.*

Étape N°1: *Edition du schéma (n'hésitez pas à utiliser les copier-coller)*

- *Vérifier que le bouton d'édition*  *est bien actif (encadré)*

→ *Rappel du schéma de l'additionneur complet 1 bit* (**main**) *à réaliser :*



- *Rechercher et placer les portes logiques nécessaires à la conception de notre* additionneur complet 1bit*: ET et XOR à 2 entrées, OU à 3 entrées.*
- *Modifier les propriétés* **Taille de la porte logique** *en* **moyen** *et* **Nombres d'entrées** *à* **2** et **3** *de la table des attributs.*

Les entrées sont repérées par des points bleus et les sorties par des points rouges.

- *Réaliser le câblage. Pour cela utiliser la souris pour effectuer les liaisons électriques conformément au schéma.*
- *Penser à sauvegarder.*

Étape N°2: *Simulation du circuit.*

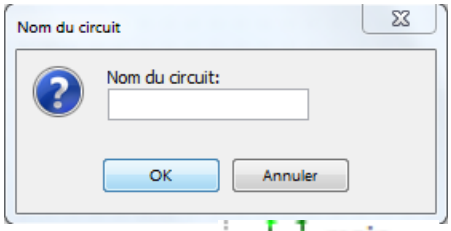
- *Dans le menu Simulation vérifier que* **Simulation enclenchée** *est cochée.*
- ***Cliquer** sur le bouton* 
- ***Cliquer** sur les pins d'entrées*  *pour produire les* **0** *et les* **1** *sur* **a**, **b** *et* **Cin**.

  (Les liaisons deviennent vertes claires avec un 1)
- ***Valider** le fonctionnement de notre Additionneur complet 1 bit.*

Etape N°3: *Encapsulation du circuit.*

Vous allez créer **un module** *additionneur complet 1bit qui contiendra votre schéma.*

- *Dérouler le menu Projet et choisir* **Ajouter Circuit**. *Vous pouvez également cliquer sur le bouton* 

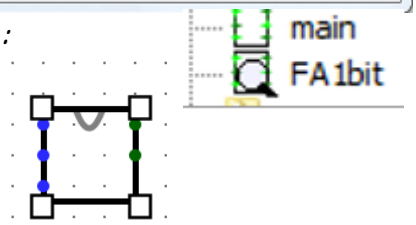


La fenêtre ci-contre apparaît:

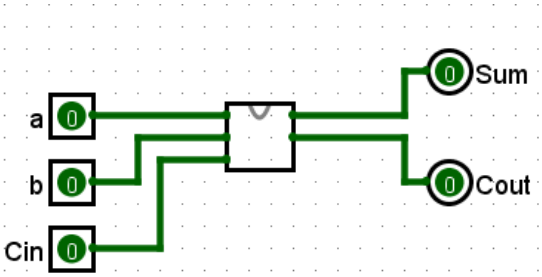
- *Renseigner-la en donnant le nom* **FA1bit**

Ce nouveau circuit est ajouté à l'explorateur du projet :

- *Cliquer sur* **main** *le module créé apparaît et présente bien 3 entrées et 2 sorties:*



Approcher *la souris près d'un point bleu ou vert et vous aurez le nom de la pin correspondante.*



Travail demandé.

Grâce à ce module de base créer un additionneur complet de 2 mots de 4 bits avec propagation de la retenue. Utiliser les **Splitters** *de la librairie* **Cablage** *et n'oubliez pas de redimensionner les "pins" d'E/S.*

C) Conception d'une ALU 8bits.

L'ALU que nous voulons réaliser maintenant doit être capable d'effectuer les opérations élémentaires suivantes :

- *Addition de 2 mots de 8bits,*
- *Soustraction de 2 mots 8bits,*
- *NON, ET, OU et OUEX sur 8bits.*

Elle devra également disposer d'un registre d'état (RE) de 3 bits (N, Z, C) dans cet ordre (N poids fort du registre).

La sélection des opérations arithmétiques et logiques est définie comme suit:

001 → Addition,
010 → Soustraction,
011 → ET,
100 → OU,
101 → OUEX,
110 → NON.

Travail demandé.

- *Créer un nouveau projet dans votre répertoire de travail :* **CPU8.cir**
- *Réaliser sous Logisim cette ALU en utilisant les modules "tous faits" de sa bibliothèque : (Portes Logiques, Plexers, Arithmétique, etc)*
- *Simuler l'ALU*
- *Sauvegarder-la après l'avoir encapsulée dans un module* **ALU8b**

Lors du prochain TP nous l'utiliserons pour qu'elle fasse partie intégrante d'un futur petit CPU 8bits.