Université Sétif 1
Faculté des Sciences
Département d'informatique

2-ème année Inforamtique (année 2019)
Module Architecture des Ordinateurs
Enseignant Kara Abdelaziz

Mini-Projet

Contenants du projet

1. Préambule

Le mini-projet pour cette année est relativement simple par rapport à ceux des années précédente, ça consiste à réaliser une architecture 4 bits minimaliste de *Harvard* qui contient tous les composants de base nécessaires pour la conception d'une architecture hardware complète programmable avec son propre langage machine, l'architecture tourne autour de l'élément principal qui est le processeur, soutenu d'une mémoire RAM pour contenir les données et d'une ROM pour contenir le programme. L'architecture en question est décrite dans la section qui suit (Description de l'architecture), c'est une représentation directe du processeur CHUMP (Cheap Homebrew Understandable Minimal Processor) décrite par *Dave Feinberg* dans son papier **A Simple and Affordable TTL Processor for the Classroom**, L'architecture représentée ici est une réplique identique que celle présentée dans le papier. L'implémentation est à faire ou-bien sur le simulateur Logisim en suivant étape par étape la construction des différents composants, ou-bien une construction réelle physique à partir des circuits logiques TTL comme décrit dans le papier ou dans les sections qui suivent. Quoique avant de pouvoir commencer la construction, il est fortement conseillé à l'étudiant d'approfondir les concepts de base de la micro-architecture (*computer organisation* en anglais) en suivant le tutoriel de *Ben Eater* utilisé dans les années passées pour construire le processeur SAP-1, ou s'inspirer de l'architecture Mic-2 présentée année dernière comme mini-projet. Les concepts et les méthodes pour maîtriser la micro-architecture sont décrites dans la section 3 (Méthodes de construction). La section 4 est un guide sous forme de conseils et directives pour aider l'étudiant lors de la réalisation du projet. La section 5 définie les clauses et les conditions pour l'acceptation du mini-projet.

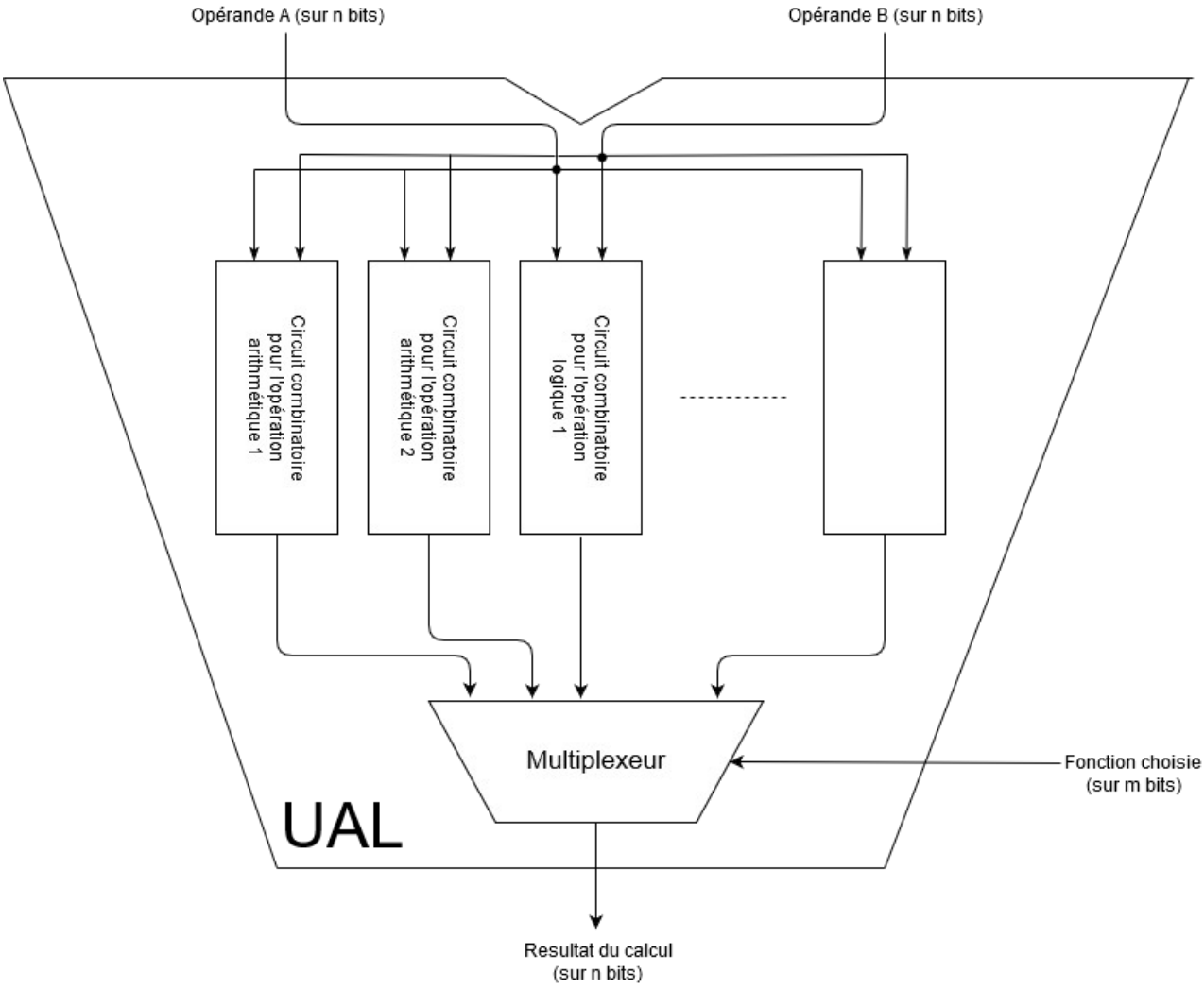
2. Description de l'architecture

La construction de l'architecture se déroule sur trois étapes, la première est celle de l'UAL (Unité Arithmétique et Logique), la deuxième est celle du datapath (Le chemin de l'instruction) et la troisième celle de l'UCC (Unité de Commande et de Contrôle). La conception d'un processeur (parce qu'ici il représente la quasi-totalité de l'architecture) toujours suit ces trois étapes, l'UAL détermine les opérations arithmétiques et logiques que le processeur peut effectuer, le datapath est un réseau de chemins dans lequel chaque instruction prend son propre chemin, c'est la métaphore d'un réseau de chemin de fer dans lequel les trains changent le chemin par des aiguilleurs, c'est les Multiplexeurs qui vont faire ce travail d'aiguillage des chemins dans le processeur. L'UCC son rôle est de reconnaître l'instruction lors de sont entrée dans le processeur (on appelle ça le décodage de l'instruction) puis de lui tracer le chemin approprié dans le datapath en manœuvrant les Multiplexeurs et les contrôles des différents composants sur le chemin.

2.1. L'UAL

Pour cette année l'UAL se base sur une UAL TTL très connue, la 74181. C'est une UAL à 4 bits qui comporte en tout 16 fonctions, pour la réalisation physique il faut se référencer au datasheet du circuit 74181 ou sur internet pour bien comprendre son mode de fonctionnement. Pour la réalisation sur simulateur deux implémentations sont envisageable, soit de reprendre le circuit de l'UAL 74181 à base de portes logiques, son schéma est disponible aussi sur internet ou sur son datasheet, et de le reproduire porte par porte sur Logisim, ou-bien de créer une nouvelle UAL simplifier en se basant sur la figure de conception générique d'un UAL en bas, dans laquelle l'UAL est présentée sous forme de plusieurs circuits combinatoires réalisant pour chacun une opération logique ou arithmétique unique, tous les circuit reçoivent les 2 opérandes A et B sur 4 bits (n=4 pour notre cas) et font tous leurs calculs en parallèle et retournent leurs résultats au Multiplexeur, un seul résultat est choisi par le Multiplexeur parmi les 8 proposés, le choix du Multiplexeur pour laisser passer le résultat d'un circuit précis est pris en rapport avec le nombre encodé en binaire sur m bits (dans notre cas m=3 puisqu'on a réellement besoin en tout que de 5 opérations) fourni sur l'entré *Fonctions* du Multiplexeur. Le tableau en dessous résume l'encodage binaire sur 3 bits de l'entrée *Fonctions* de l'UAL. Il est aussi à noter que l'ordre décrit des 5 opérations sur la table normalement pour 8 opérations n'est pas fixe et peut être permuté, et l'ajout d'autres opérations est aussi possible, quoique les 5 opérations listés sont indispensable pour le fonctionnement de CHUMP.

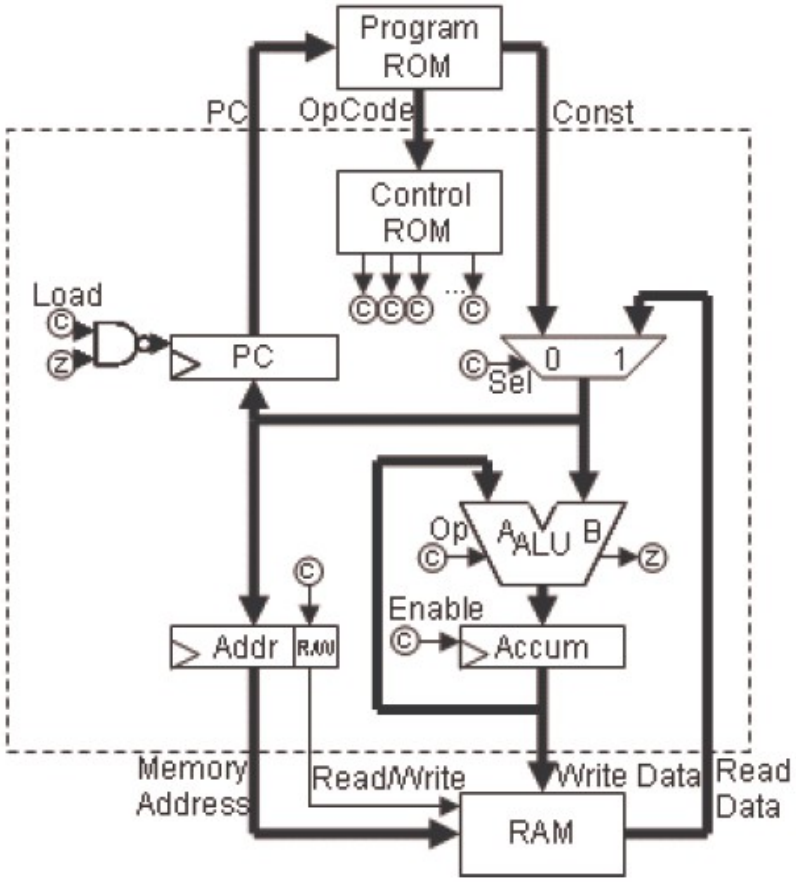
F	F2	F1	F0	Sortie de l'UAL
0	0	0	0	A
1	0	0	1	B
2	0	1	0	A+B
3	0	1	1	A-B
4	1	0	0	0
5	1	0	1	...
6	1	1	0	...
7	1	1	1	…



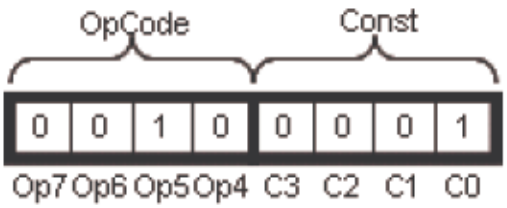
Bien sur il est possible de faire de l'optimisation pour réduire le nombre de portes, Ce n'est pas obligatoire mais vous pouvez vous inspirer de l'UAL du SAP-1 ou de Mic-2 de l'année dernière ou de l'UAL du livre *Digital Design and Computer Architecture* (section 5.2.4 page 248) pour faire de l'optimisation.

2.2. Le Datapath

Comme signalé auparavant le datapath est un réseau interconnecté de chemins, chaque instruction parcours son propres chemin dans ce réseau. Le plus souvent sur un chemin on trouve de la RAM, des registres, des circuits combinatoires (entre autres l'UAL), des Multiplexeurs, des tristate buffers...etc. Le chemin du processeur est décrit dans la figure d'en bas copiée directement du papier **A Simple and Affordable TTL Processor for the Classroom**. Le datapath pour ce processeur est assez simple, il s'appuie sur une architecture RISC (Reduced Instruction Set Computer) avec un nombre d'instructions très réduit, Ce qui permet au processeur CHUMP d'exécuter une instruction dans sa totalité en un seul cycle d'horloge. Contrairement au SAP-1 et au Mic-2 dans lesquels l'instruction est découpée en plusieurs morceaux appelés micro-instructions, et chaque morceaux est exécuté sur un cycle d'horloge.



Le datapath dans le processeur CHUMP commence par le Program ROM, une mémoire de 16 octets qui détient le programme à exécuter, chaque instruction se trouve dans une cellule mémoire de 8 bits, elle se compose de deux parties: un OpCode indiquant quelle opération le processeur doit effectuer suivi d'une valeur constante/ immédiate de 4 bits utilisée comme valeur de donnée, ou-bien comme adresse dans la RAM ou-même comme adresse qui pointe dans la ROM des programmes. La représentation binaire d'une instruction est comme sur le schéma en bas. Par conséquent, le CHUMP ne peut manipuler que des valeurs de données de 4 bits, représentant des nombres dans la plage étroite de 0 à 15 (ou -8 à 7 en complément-à-2). De même, il ne peut accéder qu'à 16 emplacements en RAM/ROM. Enfin, l'utilisation d'un PC (Program Counter) de 4 bits signifie que les programmes ne peuvent pas dépasser 16 lignes. Le jeu d'instructions de CHUMP est de 7 instructions différentes, chaque instruction peut prendre comme opérande 2 variantes ; une immédiate se trouvant sur la même instruction, ou une valeur de la mémoire RAM, ce qui revient au final à 14 instructions.



Sur le diagramme du datapath chaque composant représente un circuits TTL unique, qui sont reliés par des bus de 4 bits représentés par des flèches épaisses. Le nombre de registres utilisés à l'intérieur du CPU est 2 de 4 bits, Le registre Accum, le et le registre Addr. Le registre Accum est appelé accumulateur en raison de la façon dont il fonctionne, c'est un registre impliqué dans toutes les opérations arithmétiques du processeur, il est toujours relié au premier opérande de l'UAL, contrairement au 2-ième opérande qui peut prendre selon l'instruction soit l'immédiate, soit une valeur de la mémoire, déterminé par un Multiplexeur. Le deuxième registre Addr, c'est un registre qui détient l'adresse de la cellule pointée dans la RAM. Plus un compteur PC de 4 bits (Program Counter) pointant sur l'instruction en court d'exécution dans la ROM. Le Multiplexeur 2-1 (4 bits) est utilisé pour choisir entre l'immédiate ou l'information dans la RAM, La sortie du Multiplexeur peut être utilisée comme opérande B de l'UAL, ou comme une adresse dans le PC pour les instructions de Jump, ou encore comme adresse de RAM dans le registre Addr. Plus au final une RAM de 16 cellules avec un mot mémoire de 4 bits de taille.

2.3. Le jeu d'instructions (Instructions Set)

Le jeu d'instructions CHUMP comprend sept opérations clés, chacune se déclinant en deux versions: immédiate et mémoire. Par exemple, il existe une instruction ADD pour ajouter une constante à l'accumulateur et une autre ADD pour ajouter une valeur de la mémoire dans l'accumulateur. La partie immédiate de 4 bits de l'instruction est ignorée par les sept commandes de mémoire. Le tableau suivant décrit les sept instructions immédiates et les 7 instructions mémoire. La différence entre Les instructions mémoires et les instructions immédiat est dans le bit Op4 du poids faible de l'OpCode de l'instruction (voir le diagramme de la représentation binaire de l'instruction).

Instruction	Type	OpCode	Fonctionnement	Description
LOAD	Immédiate	000|0	Accum ← imm ; PC++ ;	Charger l'accumulateur avec une immédiate ou une valeur mémoire.
	Mémoire	000|1	Accum ← Mem[Addr] ; PC++ ;	
ADD	Immédiate	001|0	Accum ← Accum + imm ; PC++ ;	Ajouter une immédiate ou une valeur mémoire à l'accumulateur.
	Mémoire	001|1	Accum ← Accum + Mem[Addr]; PC++ ;	
SUB	Immédiate	010|0	Accum ← Accum - imm ; PC++ ;	Soustraire une immédiate ou une valeur mémoire à l'accumulateur.
	Mémoire	010|1	Accum ← Accum - Mem[Addr]; PC++ ;	
STORETO	Immédiate	011|0	Mem[Addr] ← Accum; PC++ ;	Sauvegarder la valeur de l'accumulateur dans la mémoire.
	Mémoire	011|1	Mem[Addr] ← Accum ; PC++ ;	
READ	Immédiate	100|0	Addr ← imm ; PC++ ;	Charger l'adresse mémoire avec une immédiate ou une valeur mémoire.
	Mémoire	100|1	Addr ← Mem[Addr] ; PC++ ;	

GOTO	Immédiate	101|0	PC ← imm ;	Sauter à l'instruction avec l'adresse dans l'immédiate ou une valeur mémoire.
	Mémoire	101|1	PC ← Mem[Addr] ;	
IFZERO	Immédiate	110|0	If (Accum == 0) PC ← imm ; else PC++ ;	Sauter à l'instruction avec l'adresse dans l'immédiate ou une valeur mémoire, si l'accumulateur est égal à zéro.
	Mémoire	110|1	If (Accum == 0) PC ← Mem[Addr] ; else PC++ ;	

Remarque : dans le tableau précédent *imm* désigne l'immédiate à l'intérieur de l'instruction, et Mem[Addr] la cellule mémoire avec l'adresse dans Addr.

Exemple : On peut voir sur le listing en bas d'un aperçu d'un petit programme en langage machine et en assembleur, le programme est très simple, il permet d'incrémenter +1 infiniment la cellule mémoire dans la RAM avec l'adresse 2.

```
0:  10000010    READ 2
1:  00010000    LOAD
2:  00100001    ADD 1
3:  01100010    STORETO 2
4:  10100000    GOTO 0
```

2.4. L'Unité de Commande (UC)

L'unité de commande est l'unité responsable de décoder l'OpCode de l'instruction et déclencher les commandes adéquates pour les différents composants du datapath, et permettre ainsi à l'instruction de suivre son chemin approprié dans le datapath pour l'exécutée. Sachant que CHUMP est un processeur RISC, l'instruction est à exécuter sur un seul cycle d'horloge, un circuit combinatoire est utilisé pour faire office de unité de commande, ce dernier en capturant le OpCode de l'instruction sur 4 bits permet de reproduire instantanément les signaux de commande pour chaque composant du datapath. Le circuit combinatoire sera implémenté dans CHUMP par une ROM (plus de détailles dans la section 4, **Conseils et directives** pour comment implémenter un circuit combinatoire en utilisant une ROM). La ROM en question est représentée dans le schéma du datapath par Control ROM, elle reçoit en entrée les 4 bits de l'OpCode et produits en sortie les signaux de commande.

Les composants impliqués par le contrôle de l'UC sont ceux avec le signe © sur le schéma du datapath : Le Multiplexeur, L'UAL, L'accumulateur, la RAM et le PC. Le Multiplexeur réellement reçoit la commande de sélection directement de l'OpCode sans passer par la ROM, c'est le bit Op4 avec le poids le plus faible de l'OpCode de l'instruction qui va décider si faut prendre l'information de l'immédiate ou de la RAM. L'UAL 74181 a besoin de 6 bits pour exercer une fonction, néanmoins il faut trouver le moyen de n'utiliser que 5 bits, en raison de la limitation de la ROM qui ne possède que 8 bits de mot mémoire en sortie, les 3 bits restant sont utilisés par les autres composants. La RAM avec 1 bit pour la lecture/écriture, et le PC avec 1 bit pour choisir entre incrémenter le PC ou charger une nouvelle adresse, et l'accumulateur avec 1 bit pour sa commande d'écriture. Au total ça fait 8 bits pour coïncider avec la taille de sortie de la ROM.

Les signaux de commandes pour chaque instruction sont listés sur le tableau suivant :

Instructions	ALU (5bit)	Accumulateur (1bit)	RAM (1bit)	JMP(1bit)
LOAD	B	écrire	lire	PC++
ADD	A+B	écrire	lire	PC++
SUB	A-B	écrire	lire	PC++
STORETO	X	pas d'écriture	écrire	PC++
READ	X	pas d'écriture	lire	PC++
GOTO	0	pas d'écriture	lire	écrire si Z
IFZERO	A	pas d'écriture	lire	écrire si Z

Remarque 1: La valeur X dans le tableau indique que peut importe la valeur transmise a l'UAL son résultat ne sera pas pris en compte.

Remarque 2: Le signal Z est un signal sur 1 bit en provenance de l'UAL, indiquant si le résultat en sortie de l'UAL est égale à zéro ou pas.

Il faut noter sur le diagramme du datapath l'implémentation du signal de commande pour le compteur PC, son signal est généré à travers une porte NAND, son rôle est de produire la valeur 0, donc un Jump, que si le signal Z et le signal JMP sont égaux à 1, sinon c'est la valeur 1, la valeur 1 indique au PC de faire une incrémentation PC++, et la valeur 0 (le Jump) de lui faire charger une nouvelle adresse en provenance de la sortie du Mux 2-1. En réalité, cette porte permet l'implémentation du mécanisme qui permet de faire un Jump par les 2 instructions GOTO et IFZERO. Pour l'instruction GOTO le saut est inconditionnel, et le Z doit toujours être 1, ainsi L'UAL doit exercer la fonction qui fait sortir la valeur 0 (sur l'UAL 74181, ça doit être la fonction logique 1, voir le datasheet pour plus d'explication). Par-contre pour l'instruction IFZERO, l'UAL doit faire passé l'opérande A venant de l'accumulateur pour savoir si c'est égale à zéro ou pas, donc exercer la fonction A (ça doit être la fonction $\overline{A}$ pour la 74181, voir le datasheet).

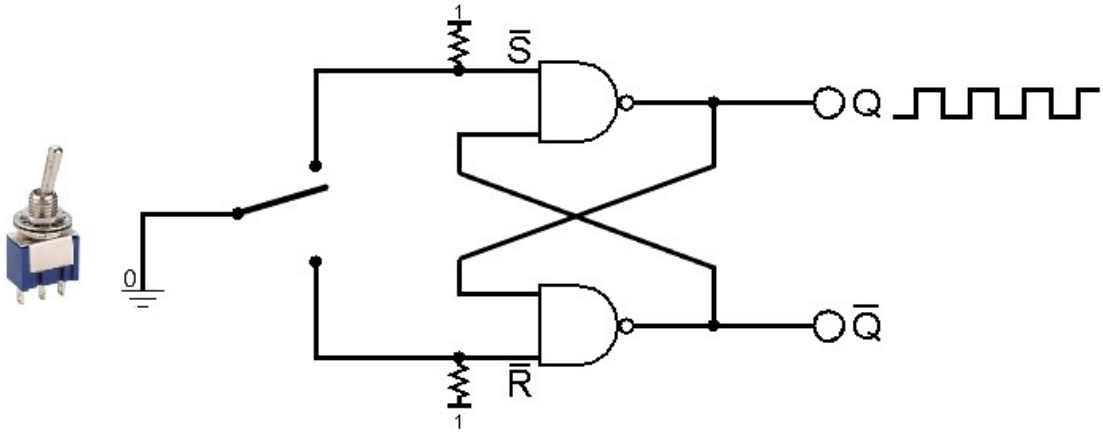
Une remarque aussi importante concernant le processeur CHUMP, c'est que le registre Addr n'a pas de signal de contrôle dédié, ce qui implique que c'est un registre qui change de valeur lors de chaque front montant d'horloge, donc pour chaque instruction. Cette information est importante a prendre en compte lors de la lecture et l'écriture dans la RAM, il faut toujours vérifier ce qui a été mis dans ce registre dans l'instruction précédente, rappelons que ça valeur vient directement de la sortie du Multiplexeur, donc de l'immédiate ou de la RAM. Cette constatation n'est réellement vraie que pour la lecture de la RAM, par conséquent une instruction LOAD doit toujours être précédée par une instruction READ. Par-contre une amélioration a été mise en place pour éviter d'utiliser 2 instructions pour l'écriture sur la RAM, c'est de faire passer le signal d'écriture de la RAM par le registre Addr (visible sur le schéma du datapath), et ainsi éviter d'ajouter une instruction READ avant l'instruction STORETO, l'écriture réellement se fait sur le front montant qui suit l'instruction STORETO. Le registre Addr doit contenir au moins 5 bits de mémoire pour contenir les 4 bits d'adresse plus le signal d'écriture de la RAM.

2.5. L'implémentation physique de CHUMP

L'implémentation physique de CHUMP ne nécessite pas plus de 10 circuits TTL, elle est assez simple, la liste des composants est sur le listing en bas, c'est un peu différent de ceux utilisés dans le papier de *Dave Feinberg* en raison de leurs disponibilités locales. Plus de détails dans la manière de les utiliser dans la section 4 Conseils et directives.

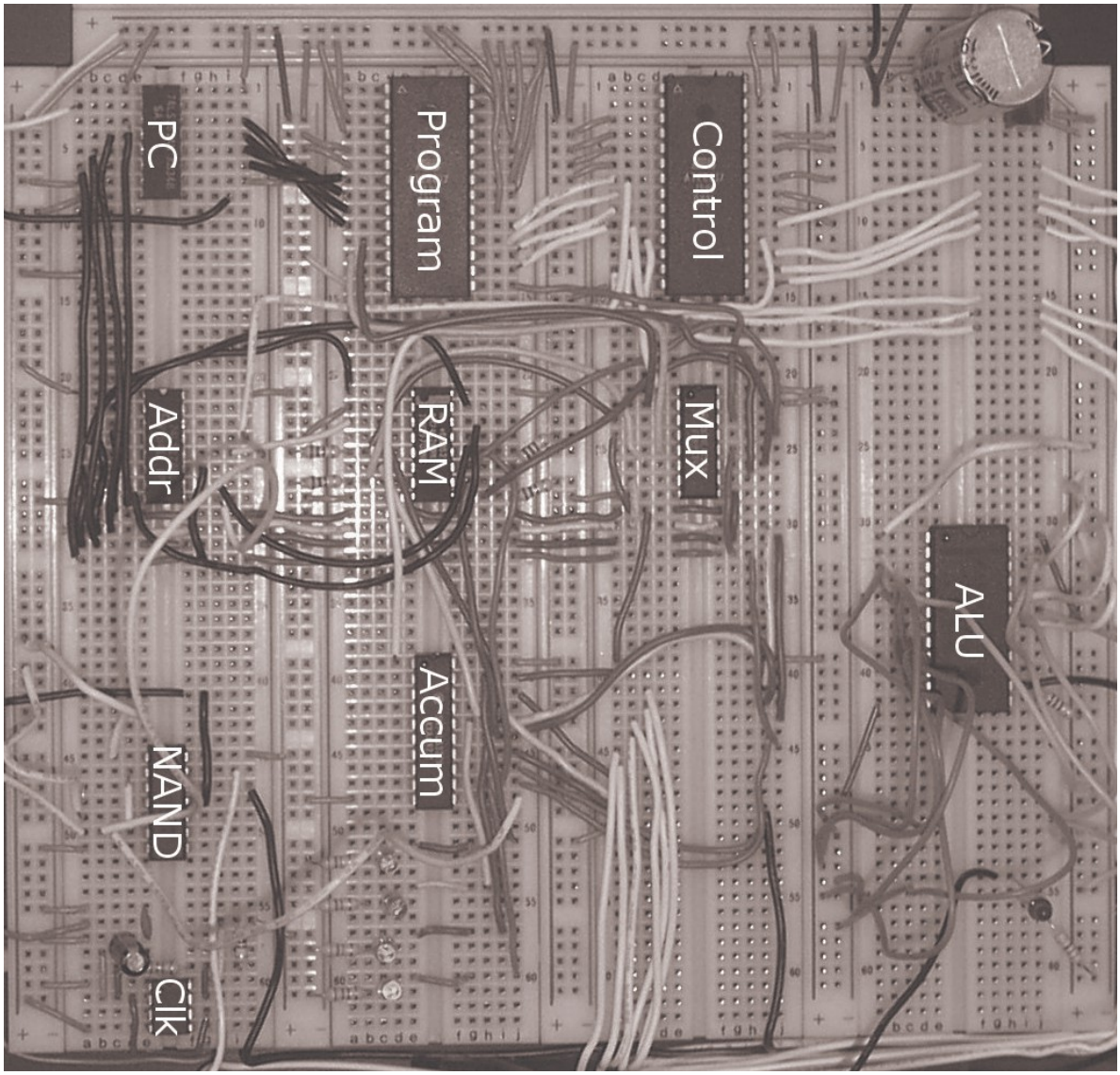
circuit	Description	Usage
7400	4 portes NAND	Horloge et signal du PC
74157	Mux 2-1 (4 bits)	Multiplexeur
74161	Compteur 4 bits	PC
74273	Registre 8 bits	Registre Addr
74181	UAL 4 bits	UAL
74377/74173	Registre 8 bits/4 bits	Accumulateur
Series HM61xx/HM62xx	RAM	RAM de données
Series AT28xxx/AT27xxx	ROM	Les 2 ROM

Remarque : Le caractère xxx désigne des valeurs différentes de référence pour chaque modèle de RAM ou de ROM. La partie xxx dans la référence d'une RAM/ROM indique généralement la capacité/taille de cette dernière.



Une représentation globale d'une implémentation physique avec les emplacements des composants et les différents bus et fils électrique les reliant, capturée du papier **A Simple and Affordable TTL Processor for the Classroom** est présentée sur l'image en bas. On peut observer que cette implémentation utilise un circuit non mentionné dans la liste pour son horloge, c'est ce le 555, pour notre cas on va utiliser une horloge manuelle plus simple, son signal est générée par un interrupteur (*toggle switch* en anglais), comme indiqué sur l'image en haut, procurant à l'utilisateur la faculté d'alterner manuellement entre 0 te 1 du signal d'horloge. L'utilisation d'une bascule Latch $\overline{R}$ $\overline{S}$ sur le schéma va permettre l'élimination du problème du debouce (rebond en français), ce sont des micro-coupures dans le signal qui existent sur tous les boutons mécaniques et qui peuvent détériorer le signal d'horloge (à chercher plus sur internet). Le signal d'horloge sera généré sur Q. Les 2 portes NAND utilisées pour construire la bascule Latch $\overline{R}$ $\overline{S}$ serons utilisées à partir du circuit 7400, sachant qu'une seule porte jusqu'à maintenant a été

utilisée et il en reste 3 de libre.



3. Méthodes de construction

Pour pouvoir construire une architecture il faut approfondir la compréhension des concepts de la micro-architecture (*Computer Organisation* en englais). Pour se faire on va s'appuyer comme pour les années passées sur un tutoriel vidéo proposant la construction d'une architecture présentée sur le site web ou la chaine Youtube de l'académicien nommé *Ben Eater* (vous pouvez le trouver en cherchant sur google). L'architecture réalisée par *Ben Eater* est inspirée de l'architecture SAP-1 (Simple As Possible 1) du livre **Digital Computer Electronic** d'*Albert Paul Malvino*. Sur la chaine Youtube, l'architecture et sa construction sont expliquées à travers une playlist de 42 vidéos qui détaillent pas à pas les différentes étapes de construction en commençant par la construction des modules de l'architecture d'une façon isolé et indépendante de l'architecture globale, ensuite de faire l'assemblage de ces modules et former une architecture globale et unique, ensuite à la fin de faire de la programmation sur celle-ci en écrivant des programmes basiques pour tester l'architecture et son langage machine. Le tutoriel vise à faire une construction physique électronique de l'architecture, pour notre cas, l'étudiant a le choix de

faire une construction physique de l'architecture de CHUMP, ou de faire plus simple, une simulation de l'architecture sur Logisim. Comme pour chaque année la solution de la simulation de l'architecture est mise à disposition sur mon dépôt *github*. Vous pouvez et il est même conseillé d'utiliser les circuits des années précédentes comme source d'inspiration, ou même de prendre des éléments et les réutiliser pour le projet de cette année. Néanmoins l'utilisation du tutoriel de Ben Eater demande quelques explications et remarques à prendre en considération. On peut les lister comme suite :

1. Il faut savoir que l'explication de l'architecture touche 2 aspects de l'électronique, l'électronique analogique et l'électronique numérique. L'électronique numérique représente la majeure partie du projet et vous avez le devoir de bien maîtriser cette partie qui entre dans votre apprentissage. Par contre l'électronique analogique, vous avez déjà le niveau académique de savoir c'est quoi le voltage, l'ampérage, une alimentation, une résistance, un condensateur et leurs fonctionnement, par contre vous ignorer le fonctionnement d'un transistor en mode switch, le comparateur, et le circuit RC, ainsi que les résistances pull-up et pull-down, dans ce cas vous n'êtes pas obligé de tout comprendre, quoique avec un peu d'effort et de recherche il vous serait assez facile de bien suivre et tout comprendre.

2. Lors de la réalisation sur simulateur, il n'est souvent pas question d'implémenter les parties analogiques, il suffit juste de comprendre le fonctionnement et de trouver une méthode pour la faire fonctionner sur le simulateur, par exemple le circuit 555 générateur d'horloge contient plusieurs éléments analogiques que vous pouvez simplement supplanter par une simple horloge dans le simulateur, malgré qu'il est très bénéfique de comprendre son fonctionnement.

3. Le transistor en mode switch dans la vidéo 3 joue le rôle d'un simple interrupteur, qui est activé ou désactivé (ouvert ou fermé) par un signal sur le troisième Pin, pour comprendre plus son fonctionnement vous pouvez faire une simulation sur Logisim, quoique il n a pas d'utilisation réel dans la simulation de notre projet.

4. Le circuit RC (pour Résistance Condensateur) que vous allez le retrouver sur la vidéo 8 et 31 est un circuit qui permet de générer une impulsion électrique de très bref délai (de quelque milliseconde), ça peut être bénéfique pour certaines situations. À noter qu'il n'est pas implémentable sur le simulateur Logisim.

5. Les résistances pull-down et pull-up doivent être implémentées sur le simulateur (vous les trouverez dans le répertoire Wiring). Elles sont utilisées tout au long du projet mais elles sont bien décrites dans la vidéo 34. En bref ces résistances sont des résistances directement branchées sur le GND = 0V (pour la pull-down) ou +5V (pour la pull-up), leurs rôle est de fournir un 0 logique faible ou un 1 logique faible. Ils sont souvent utilisés avec des bus, ou des fils susceptibles de porter la valeur logique Z (valeur flottante), et c'est justement le cas des bus lorsqu'ils sont libres et pas utilisés. La raison d'utiliser des 0 et 1 faibles au lieu de Z c'est qu'il y a des composants qui ne supportent pas une valeur flottante en entrée et présentent un comportement imprévisible ou erroné. Dans ce cas la présence d'une résistance pull-down ou pull-up pousse a superposé la valeur flottante par la valeur logique 0 ou 1 faible, et lorsque sur un fil la valeur est faible elle ne présente pas de danger de conflit (contention) avec une valeur logique normale, car elle laisse place à cette valeur dans le cas où ça devrait y arriver.

4. Conseils et directives

Cette section présente quelques conseils et directives à suivre tout au long du déroulement du projet, qui peuvent vous éclairer, vous aider et vous faciliter le parcours de ce genre de projet :

1. La première chose à prendre en compte, c'est que ce projet par nature est complexe et relativement difficile, qui devrait exiger de l'étudiant beaucoup d'investissement dans son temps et son énergie. C'est pour cette raison que le projet n'est pas obligatoire, et que pour même seulement les 10 premiers étudiants à le rendre serons acceptés, pour ne pas faire sentir à tout le monde la nécessiter de devoir le faire. Si vous pensez vouloir se lancer dans le projet, il faut prendre en considération l'aspect du temps et de l'énergie que vous devriez investir.

2. Malgré le côté long et fastidieux du projet, le bénéfice acquis en apprentissage est énorme, ça vaudrait le coup de le tenter. Même si le simple fait de visionner les vidéos et tester la simulation des années passées et de cette année (qui sera publiée à la fin du délai) reste pédagogiquement très bénéfique.

3. Pour la maîtrise et le contrôle de la complexité de l'architecture, la réalisation sur simulateur demande une utilisation judicieuse de la modularité, ainsi l'utilisation des mécanismes de bibliothèque ou des subcircuits (voir la documentation de Logisim) permettent de ne pas créer plusieurs fois le même composant, en plus de rendre le projet plus organisé et plus facile à construire. Vous pouvez vous inspirer de la réalisation des années passées, et même d'en utiliser des parties s'il conviennent au projet.

4. L'utilisation des composants du simulateur tel que les Splitters et les Tunnels (qui se trouvent dans le répertoire Wiring) permettent de diminuer encore plus la complexité du projet. La première permet de rassembler plusieurs fils sur un seul fil (de couleur noire). La deuxième permet de créer une liaison filaire entre deux points distants sans pour autant dessiner le fil, qui serait dans le cas contraire une opération fastidieuse.

5. Le projet a une vocation de recherche académique personnelle, les principaux références pour le projet sont; les 42 vidéos de Ben Eater, le livre *Digital Design and Computer Architecture 2nd edition* de *David Harris*, le papier **A Simple and Affordable TTL Processor for the Classroom** de *Dave Feinberg*, les projets des années passées, Le guide utilisateur de Logisim (menu Help >User's guide), youtube, et le moteur google. Si vous avez fait le tour complet sur une question ou un point qui vous pose problème, vous pouvez venir me voir et demander conseil, et c'est aussi valable pour la partie électronique du projet.

6. Le projet sur simulateur est relativement complexe et il existe plusieurs façons de l'implémenter, c'est pour cette raison qu'il est pratiquement impossible d'avoir des solutions strictement identiques. Le projet est destiné a être fait individuellement, si d'après jugement personnel je remarque un degré de similitude entre 2 ou plusieurs projets la note sera amputé en rapport avec le degré de similitude de ces projets. Le projet aussi doit suivre la description donnée dans la section 2 et les directives dans cette section, une pénalité est aussi applicable sur les réalisation qui dérive de

l’énoncé du mini-projet.

7. Il est impératif d'implémenter les circuits combinatoires à l’intérieur de l'UAL, du multiplexeur et du décodeur à partir de portes logiques et de ne pas les utiliser directement à partir des bibliothèques standard du simulateur. Il est de même pour les registres et où il faut les construire à partir de flipflops. Sauf exception pour le compteur où on a pas eu assez de temps pour l'étudier.

8. Par contre les RAMs et les ROMs il est préférable d'utiliser ceux fournies par le simulateur que de les créer soi-même en raison de leurs complexités.

9. Dans le projet et comme expliqué dans la vidéo 31 la ROM est utilisée comme une méthode alternative pour implémenter les circuits combinatoires, cette méthode est basée sur le fait que les entrées dans la table de vérité d'un circuit combinatoire représentent la porte l'adresse dans la ROM et les sorties représentent l'information à la sortie de la ROM, en résumé la table de vérité est l'adresse/information à l’intérieur d'une ROM. Pour savoir comment utiliser la RAM et la ROM et aussi les Splitteurs et les Tunnels, il faut s'appuyer sur le guide utilisateur de Logisim.

10. La méthode de construction utilisée par *Ben Eater* est académique plus que professionnelle, c'est pour cette raison que son implémentation est plus orientée pour démontrer les concepts que de faire une implémentation efficace, et ça c'est clairement visible sur les LED de débogage pour indiquer le contenu des registres, de la mémoire ou du bus, et le mécanisme pour faire entrer les valeurs manuellement dans la RAM. Il est préférable de suivre la même logique pour l’implémentation sur le simulateur, c'est à dire d'ajouter des LED indicateurs sur les registres, les bus, et la mémoire, et de faire le mécanisme d'entré manuel sur la RAM.

11. Pour la réalisation physique, il existe des composants mentionnés dans le papier qui ont été remplacer par d'autres composants équivalents, la raison est que ses composants sont introuvable en Algérie, et il était nécessaire de trouver et les remplacer par d'autres avec la même équivalence fonctionnelle.

12. Les ROM et La RAM utilisés dans le papier sont aussi introuvable en Algérie, pour les remplacer il faut chercher un modèle équivalent dans les séries mentionnés plus haut. Toute mémoire, que ça soit RAM ou ROM dans les séries devrait fonctionner, tant-que sa taille/capacité est supérieur ou égale à celle utilisée, et que la cellule mémoire doit aussi être supérieure ou égale à celles mentionnées dans le papier.

5. Clauses et conditions

Le mini-projet doit suivre les causes et conditions suivante pour qu'il soit accepté :

1. L'enseignant chargé du TD/TP doit avoir au préalable annoncé la participation de ses étudiants au projet, si ça n'a pas été annoncé par l'enseignant, ses étudiants ne sont concernés par le projet.

2. L'implémentation du projet doit se faire en monôme, sauf autres indications de votre enseignant de TD/TP.

3. La réalisation du mini-projet vous octroie une note bonus de 10 points additionnels à votre note de TD et TP pour la simulation, et une note de 20 pour le TD et TP pour la réalisation physique. sauf autres indications de votre enseignant de TD/TP.

4. La date butoir pour la remise des mini-projets est à minuit 00:00 avant le premier jour des examens. Une solution pour la simulation sera après publiée dans le dépôt *github*.

5. Seules les 10 premières solutions correctes serons acceptées pour la simulation. Et pas de limite de nombre pour la réalisation physique.

6. La condition pour que le mini-projet soit accepté est qu'il puisse exécuter le programme factoriel et le programme de la suite de Fibonacci avec en résultat pour les 2 algorithmes la plus grande valeur représentable sur 4 bits. Les 2 programmes doivent être inclus avec les fichiers du projet.

7. Ceux qui ont fait une réalisation physique, doivent me contacter et fixer un rendez-vous pour pouvoir faire des testes réels sur l'architecture.

8. La solution doit contenir tous les fichiers .circ de votre projet, plus les fichiers des mémoires ROM, plus les 2 programme en langage machine (les images mémoire de la Program ROM), plus les 2 programme écrit en Assembleur dans un fichier texte.

9. Tous les fichiers doivent être regroupés sur un même fichier compressé et envoyés par émail en fichier joint avec le nom le prénom et le groupe de l’étudiant à l'adresse *kara.abdelaziz@el-kalam.com*.

وفقكم الله