ÉCOLE CENTRALE LYON

Fonctionnement interne d'un microprocesseur : chemin de données, chemin de contrôle et programmation bas-niveau.

L'objectif de ce TP est la compréhension et la pratique à bas niveau d'abstraction du fonctionnement d'un microprocesseur. A l'issue de ce TP, l'étudiant aura appréhendé de manière fine le chemin de données et le chemin de contrôle d'un microprocesseur. Ayant établi des programmes en langage assembleur puis en binaire, il aura acquis une maîtrise du microprocesseur à bas niveau d'abstraction.
Le rôle de l'étudiant sera :

1) de suppléer au *compilateur* en traduisant des instructions exprimées en langage haut niveau (addition de deux nombres,…) en mnémonique assembleur puis
2) de suppléer à l'*assembleur* en effectuant la traduction entre les mnémoniques assembleur et le code machine binaire nécessaire à l'exécution du code.

L'architecture du microprocesseur utilisée est celle utilisée en séance du TD7. Il est à noter que si, cette architecture reste volontairement simple, elle n'est pas simpliste et donne les bases à la compréhension de tout microprocesseur moderne, aussi complexe soit-il.

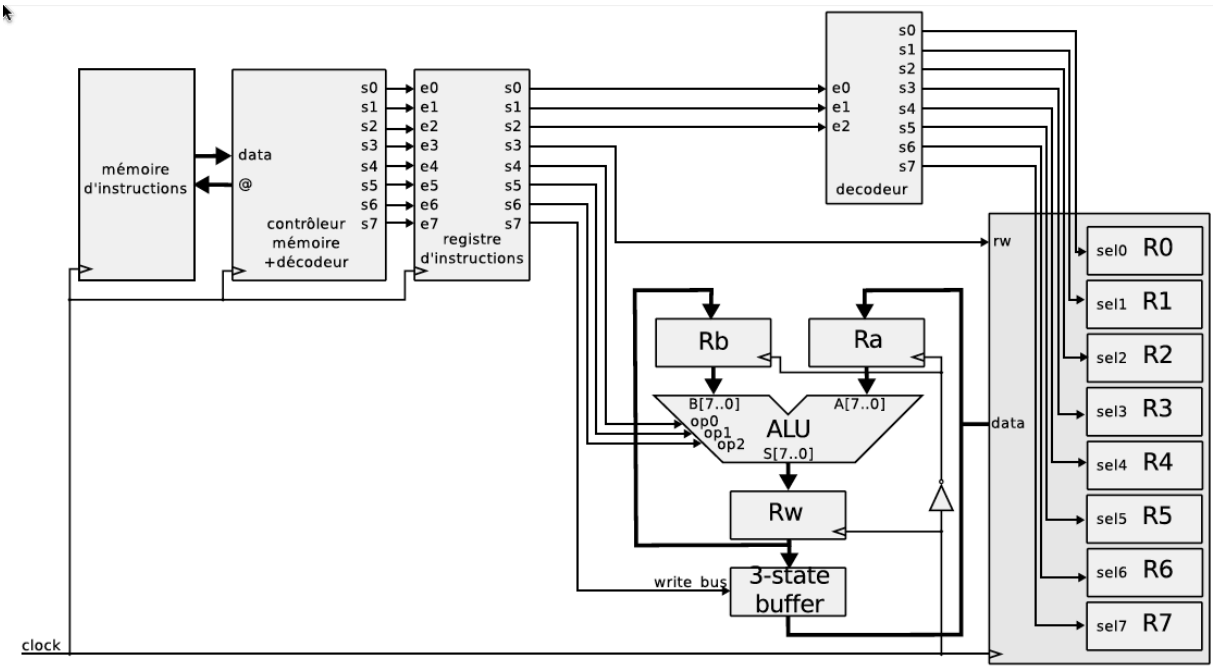

Figure 1 : Architecture du microprocesseur.

Travail préliminaire:
Identifiez sur la carte les différents éléments correspondant aux blocs de l'architecture de la Figure 1.

Le jeu d'instruction du microprocesseur est le suivant:

$\overline{WriteBus}$	OP_2	OP_1	OP_0	$R\overline{W}$	R_2	R_1	R_0
0 Écriture **1** Haute impédance	**000** *ADDI* **001** *SUB* **010** *MULT* **011** *W=A (NOP)* **100** *DIV* **101** $\bar{A}$ **110** *A+B+1* **111** *Réservé*			Sens de communication avec les registres R_0 à R_7 **1** lecture **0** écriture	**000** Registre 0 **001** Registre 1 **010** Registre 2 **011** Registre 3 **100** Registre 4 **101** Registre 5 **110** Registre 6 **111** Registre 7		

Donnez la correspondance entre les mnémoniques assembleur et le code binaire (on utilisera XXX pour désigner un registre R_x).
STR(R_x) :
LD(R_x) :
ADD(R_x) :
SUB(R_x) :
MULT(R_x) :
DIV(R_x) :
INV(R_x) :
ADDINC(R_x) :

Partie 1 : Système de génération d'horloge :

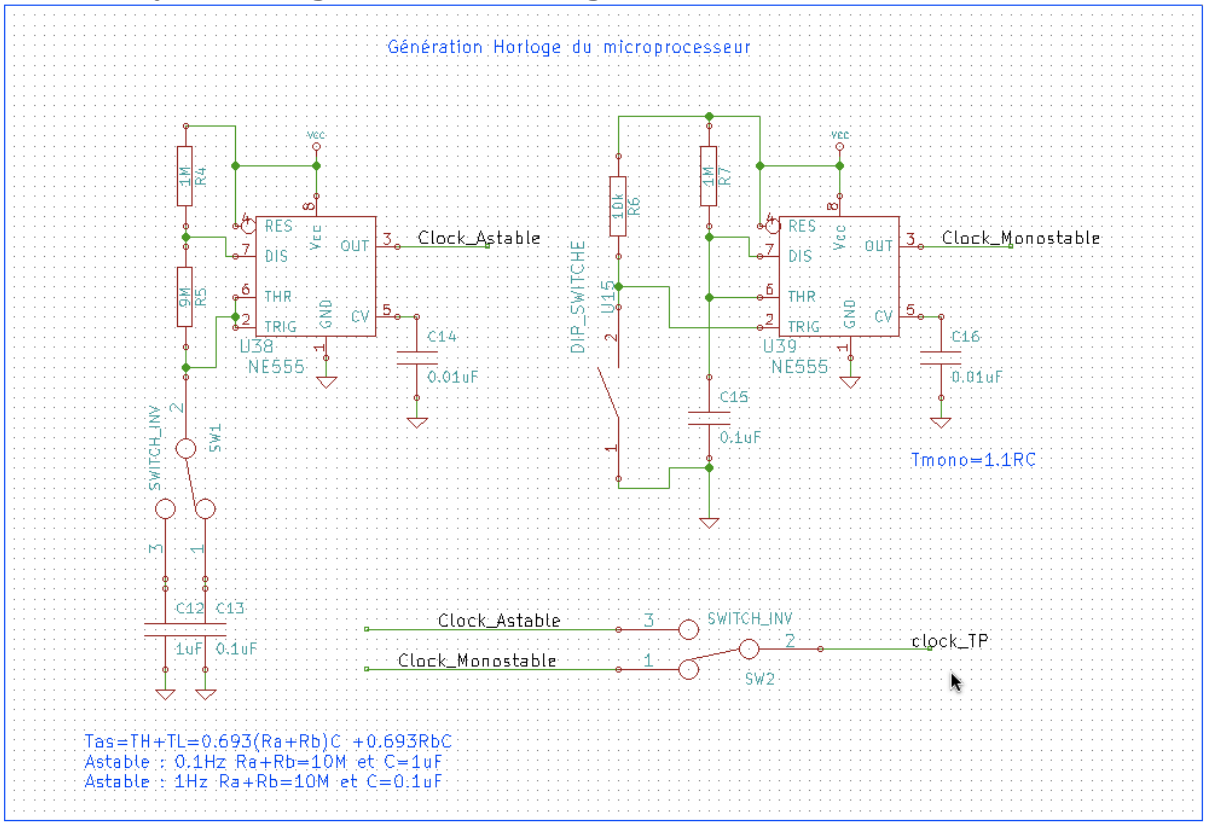

Figure 2: Système de génération de l'horloge du système.

Le système de génération d'horloge utilise deux NE555. Déterminez les configurations des deux NE555. Expliquez le fonctionnement de ce système et les différents modes de fonctionnement à l'aide du schéma électrique précédent et de la carte de TP

Partie 2 : Analyse du programme par défaut.
Chargez le programme suivant en hexa.
BD 30 BD 31 BD 32
Convertissez ce code en langage assembleur et interprétez le programme.
Exécutez le programme en mode automatique et en mode pas à pas. Expliquez le fonctionnement de ce programme. Tracer le diagramme temporel de l'état du bus, des registres A, B et W.

Exécutez maintenant le suivant:
B9 33 BA 31 BB 32
Exécutez le programme en mode automatique et en mode pas à pas. Expliquez le fonctionnement de ce programme. Tracer le diagramme temporel de l'état du bus, des registres A, B et W.

Partie 3 : Implémentation de l'addition
On cherche à additionner deux valeurs (décimales) 89 et 23 : vous utiliserez les interrupteurs pour stocker ces deux valeurs dans les registres R0 et R1. Le résultat sera stocké dans le registre R2. Écrivez le programme en mnémoniques assembleur puis écrivez le code binaire correspondant que vous interpréterez en hexadécimal. Implémentez le programme en mémoire instruction. Vérifiez le bon déroulement du programme et tracez le chronogramme des registres A, B et W.
Nota : il pourra être judicieux pour le développement du programme de rentrer une bonne fois pour toute les valeurs dans les registres R0 et R1 et ensuite de travailler uniquement que sur l'addition à partir des registres R0 et R1 auxquels on ne touchera pas pour conserver leur contenu.

Partie 4 : Soustraction.
On va chercher à implémenter l'opération 56-27. On commencera par établir le programme en utilisant l'opération soustraction incluse dans le jeu d'instruction du microprocesseur en utilisant dans l'ordre les registres R0 (56), R1 (27) et R2 (résultat). Écrivez les mnémoniques assembleurs puis le code binaire. Implémentez le programme dans la mémoire instruction et assurez vous du bon déroulement du programme.
Dans un second temps, nous allons instancier la même opération mais en utilisant cette fois-ci le complément à deux. Rappelez le principe de la soustraction en complément à deux. Écrivez le programme en mnémonique assembleur, convertissez en code binaire puis hexadécimal et implémentez-le. Assurez-vous du bon fonctionnement du programme et comparez les résultats obtenus avec le programme précédent.
En particulier, on s'attachera à comparer le nombre de cycles machine nécessaires à l'opération.

Partie 5 : Étude du bus
On va chercher à comprendre le rôle du code 1000 0010. Analysez ce code et explicitez son rôle. Implémentez le, puis expliquez l'utilité de ce code.

Partie 6 : Accumulation.

Le programme suivant va être modifié pour incorporer l'accumulation. On va chercher à accumuler 23 trois fois (23+23+23). Cette étape appelée accumulation et à la base de nombreux processus de calcul des microprocesseurs actuels. Modifiez le programme précédent pour écrire cette nouvelle opération. Écrivez le programme en mnémoniques assembleur puis écrivez le code binaire correspondant que vous interpréterez en hexadécimal. Implémentez le programme en mémoire instruction.
Tracez le chronogramme des registres A, B et W.