Projet pratique : Partie 1 Circuits numériques

1. Introduction

Ce projet vise à simuler des circuits combinatoires (décodeur, multiplexeur, comparateur) et à développer un programme simple pour l'émulateur 8086. À l'aide de SimulIDE, nous avons conçu et testé les circuits pour comprendre les principes de la logique combinatoire. Ensuite, un programme a été réalisé pour observer l'état des registres du microprocesseur 8086. Ce rapport détaille les étapes, les résultats et les apprentissages issus de ces simulations.

2. Circuits combinatoires

a. Décodeur 2x4

Objectif : Comprendre le fonctionnement des circuits combinatoires qui effectuent des opérations de décodage et analyser comment les portes logiques sont utilisées pour obtenir des sorties spécifiques.

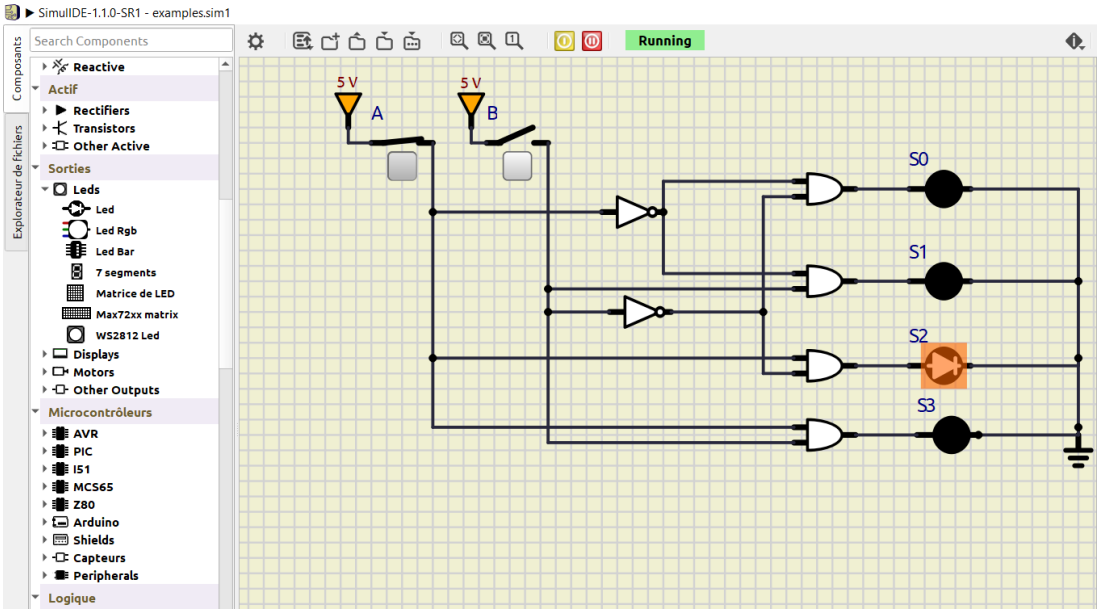


*Figure 1:Décodeur 2*4*

Le circuit du décodeur a été construit en utilisant les composants suivants dans SimulIDE :

- **Entrées** : Deux interrupteurs représentant A et B.
- **Portes logiques** : Des portes NOT & AND
- **Sorties** : Quatre LEDs affichant les états des sorties.

Après avoir connecté les composants et lancé la simulation les résultats sont présentés dans le tableau ci-dessous :

A	B	S0	S1	S2	S3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Equations de sorties : $S_0 = \overline{A} \cdot \overline{B}, \quad S_1 = \overline{A} \cdot B, \quad S_2 = A \cdot \overline{B}, \quad S_3 = A \cdot B$

b. Multiplexeur 8x1

L'objectif : Comprendre comment un multiplexeur permet de gérer plusieurs signaux d'entrée et de transmettre un signal unique à la sortie en fonction des combinaisons des lignes de sélection.

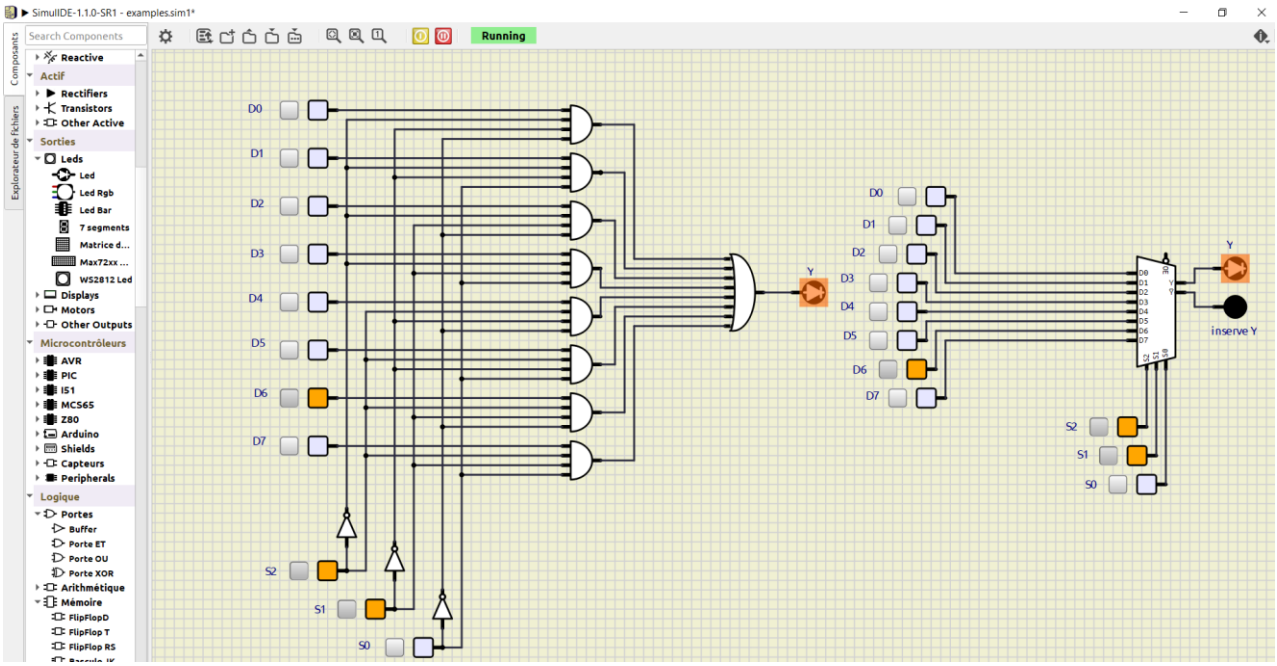


Figure 2:Multiplexeur 8x1 et son logigramme

Le circuit du multiplexeur (MUX) 8x1 a été construit en utilisant les composants suivants dans SimulIDE :

- **Entrées** : 8 interrupteurs (D0 à D7).
- **Lignes de selection**: Trois interrupters (S0, S1, S2).
- **Portes logiques** : NOT, AND, et OR pour implémenter la formule logique.
- **LED** : Pour visualiser l'état de la sortie Y.

Étapes de simulation :

1. Configurer les interrupteurs pour représenter les entrées (D0 à D7) et les lignes de sélection (S0, S1, S2).
2. Construire les portes logiques selon la formule logique :

$$Y = (\overline{S_2} \cdot \overline{S_1} \cdot \overline{S_0} \cdot D_0) + (\overline{S_2} \cdot \overline{S_1} \cdot S_0 \cdot D_1) + (\overline{S_2} \cdot S_1 \cdot \overline{S_0} \cdot D_2) + (\overline{S_2} \cdot S_1 \cdot S_0 \cdot D_3)$$
$$+(S_2 \cdot \overline{S_1} \cdot \overline{S_0} \cdot D_4) + (S_2 \cdot \overline{S_1} \cdot S_0 \cdot D_5) + (S_2 \cdot S_1 \cdot \overline{S_0} \cdot D_6) + (S_2 \cdot S_1 \cdot S_0 \cdot D_7)$$

3. Relier les sorties des portes AND à une porte OR unique, connectée à la LED.
4. Tester les différentes combinaisons des lignes de sélection et vérifier la sortie.

S2	S1	S0	Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

Résultats obtenus :
Les tests ont montré que, pour chaque combinaison des lignes de sélection

(S0 , S1 , S2), une seule entrée est transmise à la sortie Y. Les résultats sont résumés dans le tableau suivant :

3. Émulateur 8086

a. Objectif

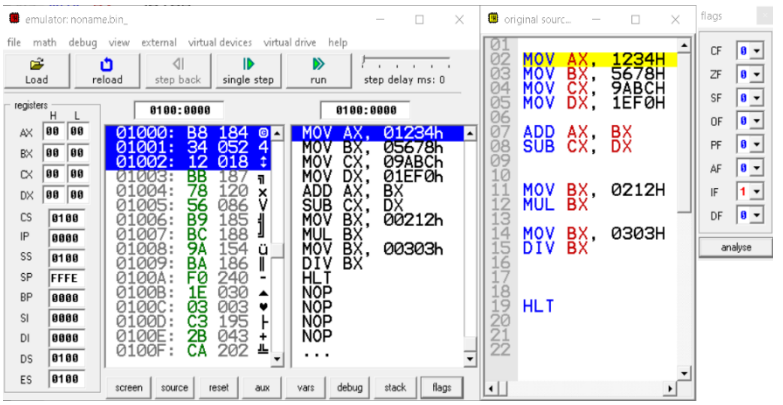
Le but de cette partie est d'écrire un programme en langage assembleur pour exécuter des opérations arithmétiques et logiques tout en observant l'évolution des registres à chaque étape de l'exécution

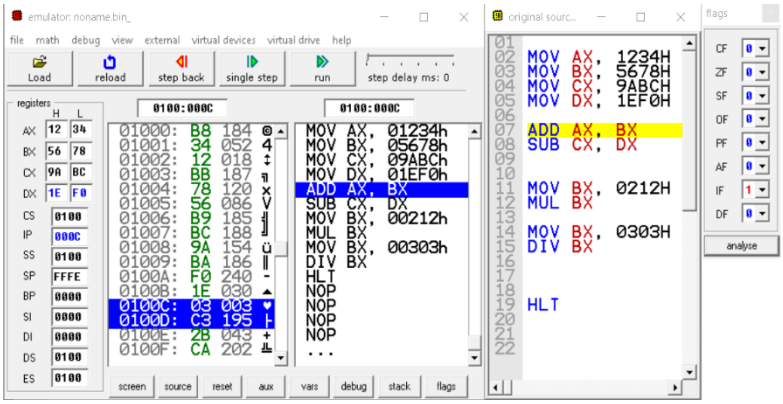
b. Programme utilisée :

Le programme suivant, conçu pour l'émulateur EMU8086, réalise plusieurs opérations sur les registres du microprocesseur

```
01
02 MOV AX, 1234H
03 MOV BX, 5678H
04 MOV CX, 9ABCH
05 MOV DX, 1EF0H
06
07 ADD AX, BX
08 SUB CX, DX
09
10
11 MOV BX, 0212H
12 MUL BX
13
14 MOV BX, 0303H
15 DIV BX
16
17
18 HLT
19
```

- **Initialisation des registres**



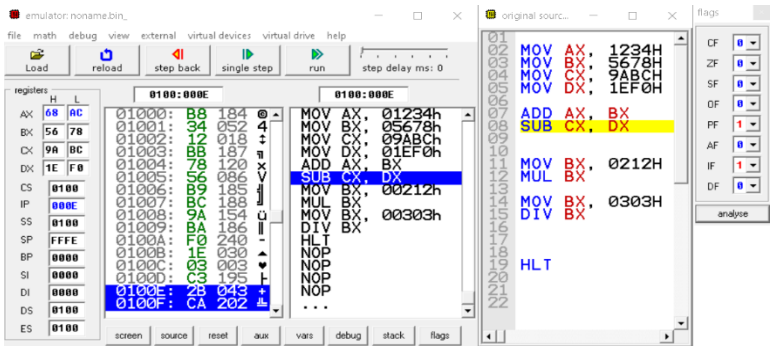


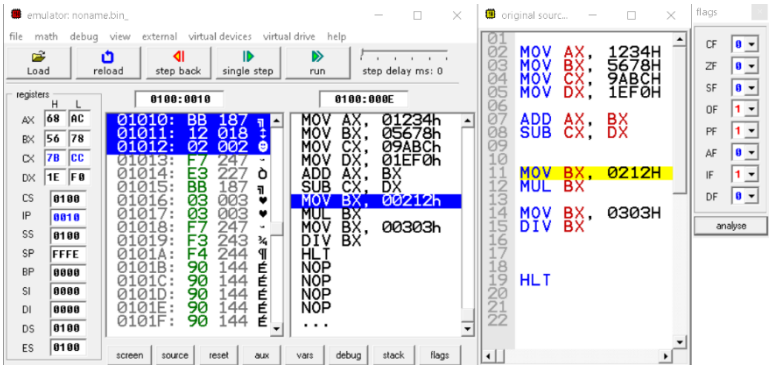
Fonction de l'IP : L'Instruction Pointer (IP) contient l'adresse mémoire de la prochaine instruction à exécuter dans le programme. Son contenu est mis à jour après chaque instruction, soit automatiquement par le processeur

IP : 0000 a l'état initiale

Les registres AX, BX, CX, et DX ont été initialisés avec les valeurs respectives 1234H, 5678H, 9ABCH et 1EF0H et Insctruction ponteur IP :000C. et IF= 1 (Les interruptions sont activées)

- **Addition et Soustraction**

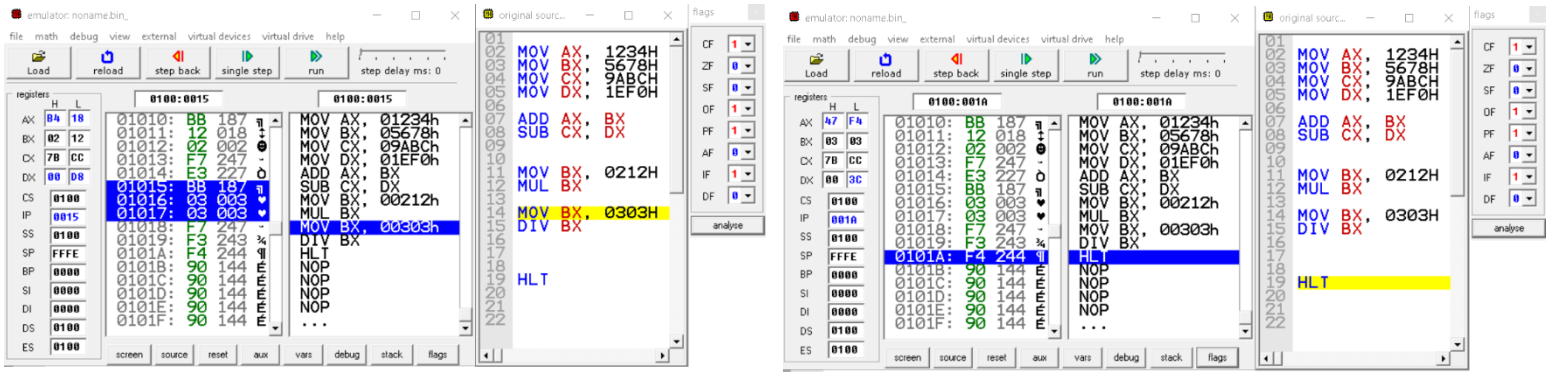




Addition : La valeur de BX (5678) a été ajoutée à celle de AX (1234), et le résultat de l'addition dans AX devrait être 68AC et Insctruction ponteur IP :000E et PF =1

Soustraction : CX(9ABC) a été décrémenté de la valeur de DX (1EF0). Résultat attendu : 7BCC dans BX et Insctruction ponteur IP :0010. et OF= 1

- **Multiplication et Division :**



Multiplication : AX(68AC) a été multiplié par BX(0212). Le produit est stocké dans les registres DX:AX(00D8-B418) et Insctruction pointeur IP :0015 et CF=1

Division : DX:AX(00D8B418) a été divisé par BX(0303). Le quotient est stocké dans AX(47F4), et le reste dans DX(003C). et Insctruction pointeur IP :001A et CF=1

HLT : Fin du programme

4- Conclusion

Ce projet a permis d'explorer deux volets complémentaires : la simulation de circuits combinatoires et la programmation sur un émulateur 8086. Dans la première partie, nous avons conçu et testé des composants numériques comme le décodeur et le multiplexeur, ce qui nous a aidés à mieux comprendre leur fonctionnement et leur utilité dans les systèmes numériques. La seconde partie a consisté à écrire un programme simple pour analyser l'état des registres du microprocesseur 8086, approfondissant ainsi notre connaissance de son architecture.