RAPPORT
MINI-PROJET
COMMUNICATION BAS-NIVEAU

Réalisé par :

AHOUSSI NATHAN

SOMMAIRES

INTRODUCTION

Le circuit projet.circ est celui d'un microprocesseur qui a les caractéristiques suivantes :

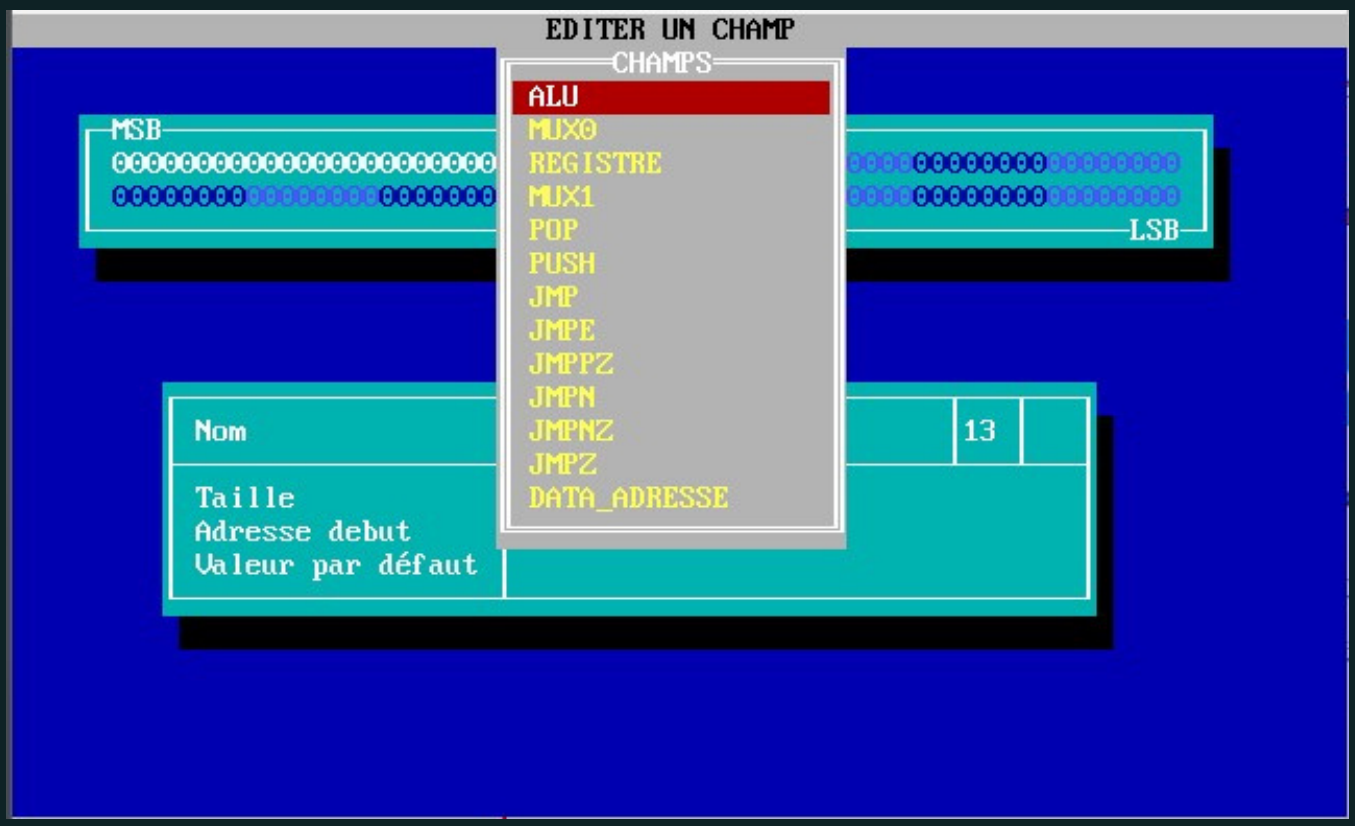
- **Instructions sur 32 bits**
- **Code opératoire sur 16 bits. Adresses ou données sur 16 bits**
- **Les données sortantes de l'ALU peuvent être stockées dans un des deux registres 16 bits (Registre A ou Registre B) ou dans une pile LIFO. Elles peuvent également servir pour sélectionner une adresse de la mémoire des données.**
- **L'entrée I0 de l'UAL peut prendre ses données soit à partir du Registre A, soit directement depuis le champ DATA ou Adresse du code de l'instruction et ceci en fonction de la valeur du bit MUX0.**
- **L'entrée I1 de l'UAL peut prendre ses données soit à partir du Registre B, soit à partir du Registre C en fonction de la valeur du bit MUX1.**

Ce circuit comporte 2 mémoires : une pour les instructions et une autre pour les données. Il est fait de telle sorte qu'il supporte un adressage indirect à partir des registres A ou B, c'est-à-dire qu'il est possible de charger dans le registre C une donnée (de la mémoire des données) dont l'adresse se trouve dans le registre A ou B. Il supporte également l'adressage direct, c'est-à-dire qu'il est possible de charger dans le registre C une donnée (de la mémoire des données) dont l'adresse se trouve directement dans le champ Adresse de l'instruction.

Il comporte également une pile LIFO (Last In First Out) qui permet le stockage temporaire de données grâce à la borne Push et leur récupération grâce à la borne Pop. Dans le cas d'une instruction PUSH, la donnée sur 16 bits à empiler doit être présente sur l'entrée Data in. Dans le cas d'une instruction POP, la donnée sur 16 bits à dépiler, peut être récupérée sur la sortie Data out. Deux indicateurs Empty et Full passent à 1 respectivement lorsque la pile est vide ou lorsque la pile est pleine.

La description détaillée des champs que j'ai crée.

Liste des champs que j'ai crée pour le circuit :



Elément du champs DATA :

Le champ DATA est sur 16 bits et représente une adresse mémoire de la RAM de notre circuit. La valeur par défaut est 0000000000000000 (LSB). Il permet de stocker une donnée en entrée ou en sortie du circuit.

Elément du champs des JUMPS :

- **JMPZ : ce champ est sur 1 bit et a une valeur par défaut de 0. Il est utilisé pour indiquer si un saut conditionnel JMPZ doit être effectué lors de l'exécution du programme.**

- **JMPNZ : ce champ est sur 1 bit et a une valeur par défaut de 0. Il est utilisé pour indiquer si un saut conditionnel JMPNZ doit être effectué lors de l'exécution du programme.**

- **JMPN : ce champ est sur 1 bit et a une valeur par défaut de 0. Il est utilisé pour indiquer si un saut conditionnel JMPN doit être effectué lors de l'exécution du programme.**

- **JMPPZ : ce champ est sur 1 bit et a une valeur par défaut de 0. Il est utilisé pour indiquer si un saut conditionnel JMPPZ doit être effectué lors de l'exécution du programme.**

- **JMPE : ce champ est sur 1 bit et a une valeur par défaut de 0. Il est utilisé pour indiquer si un saut conditionnel JMPE doit être effectué lors de l'exécution du programme.**

- **JMP : ce champ est sur 1 bit et a une valeur par défaut de 0. Il est utilisé pour indiquer si un saut inconditionnel JMP doit être effectué lors de l'exécution du programme.**

Elément du champs de PUSH:

Le champ PUSH est sur 1 bit et a une valeur par défaut de 0. Il est utilisé pour indiquer si l'instruction PUSH doit être exécutée lors de l'exécution du programme. C'est à dire empiler un élément sur la pile LIFO.

Elément du champs de POP:

Le champ POP est sur 1 bit et a une valeur par défaut de 0. . Il est utilisé pour indiquer si l'instruction POP doit être exécutée lors de l'exécution du programme. C'est à dire dépiler un élément de la pile LIFO.

Elément du champs de MUX0 :

Le champ MUX0 est sur 1 bit et a une valeur par défaut de 0. Il est utilisé pour indiquer le choix du premier multiplexeur (MUX0) du circuit.

Elément du champs de MUX1 :

Le champ MUX1 est sur 1 bit et a une valeur par défaut de 0. Il est utilisé pour indiquer le choix du deuxième multiplexeur (MUX1) du circuit.

Elément du champs de l'ALU:

Le champ ALU est sur 4 bits et a une valeur par défaut de 0000. Il est utilisé pour indiquer l'opération à effectuer dans l'unité arithmétique et logique (ALU) du circuit.

Explication sur les programmes que j'ai réalisé

Pour le programme permute.asm qui est censé charger la valeur 10=(A)16 dans le registre A et la valeur 11=(B)16 dans le registre B puis permute les valeurs des 2 registres en utilisant la pile LIFO.

```
LOAD_A #10 ; charge la valeur 10 dans le registre A
LOAD_B #11 ; charge la valeur 11 dans le registre B
PUSH_A ; empile la valeur 10 dans la pile LIFO
PUSH_B ; empile la valeur 11 dans la pile LIFO
POP_A ; depile la valeur 11 qui est au sommet de la pile LIFO
    ; et la met dans le registre A
POP_B ; depile la valeur 10 qui est au sommet de la pile LIFO
    ; et la met dans le registre B
```

Pour le programme express.asm qui est censé vérifier si une expression mathématique est correctement parenthésée et écrire à la fin dans le registre A la valeur 12=(C)16 si l'expression est Correctement parenthésée, ou la valeur 14= (E)16 si elle est Erronée.

```
        LOAD_A 0   ;charge l'adresse de la première parenthèse dans le registre A
        PUSH_B   ;empile 0 dans la pile LIFO


        LOAD_C_ADRA ; charge le caractère suivant de l'expression dans le registre C
        CMP_B #28   ; compare B avec le code ASCII de '('
        JMPZ SKIP   ; si le caractère est une parenthèse ouvrante, saute SKIP
boucle  CMP_B #29   ; compare B avec le code ASCII de ')'
        JMPE ERREUR  ; si la pile est vide, saute la boucle ERREUR
        JMPZ ERREUR  ; si le caractère est une parenthèse fermante, saute la boucle ERREUR
        PUSH_B     ; empile 0 dans la pile LIFO


        INC_A      ; avance l'adresse de caractère suivant
SKIP    CMP_B #0    ; compare avec la fin de l'expression
        JMPE END    ; si la fin est atteinte, saute END
        JMP boucle  ; retourne au début de la boucle


ERREUR  LOAD_A #14 ; charge la valeur d'erreur dans le registre A
        JMP FINISH ; saute à la fin du programme


        POP_B     ; dépile le résultat final de la pile LIFO
        CMP_B #0    ; compare avec 0
END     JMPZ CORRECT ; si la pile est vide, la parenthèse est correctement parenthésée
        LOAD_A #14 ; charge la valeur d'erreur dans le registre A
        JMP FINISH ; saute à la fin du programme

CORRECT:
        LOAD_A #12 ; charge la valeur de réussite dans le registre A

FINISH:
        NOP ; ne fait rien
```