R204 – Projet Communication Bas Niveau

PAYOL Albert

BEN OUIRANE TAYCIR

Table des matières

Taycir Ben Ouirane
Albert Payol

1. Introduction

Ce rapport présente le travail réalisé dans le cadre d'un projet en binôme visant à développer un algorithme en utilisant un ensemble d'instructions spécifiques conçues à l'aide du logiciel MAG. Notre objectif principal était de concevoir un ensemble d'instructions et de champs permettant d'effectuer des opérations bas niveau sur une architecture de processeur hypothétique.

Le domaine du bas niveau en informatique est crucial pour la compréhension et la manipulation directe du matériel informatique. Il implique des opérations proches du langage machine, offrant un contrôle précis sur les composants matériels tels que les registres, la mémoire et les opérations d'entrée/sortie.

Le logiciel MAG (Microprogrammable Architecture Generator) est un outil puissant utilisé par les concepteurs de processeurs pour simuler, tester et développer des architectures de processeurs personnalisées. Il permet aux ingénieurs de définir des jeux d'instructions sur mesure, des champs de contrôle et des opérations spécifiques pour créer des processeurs adaptés à des applications spécifiques.

2. Création des Champs

Nous avons initié le projet en concevant plusieurs champs dans le logiciel MAG. Voici une liste des champs créés avec leurs attributs :

- **JMP**
- **MUX1**
- **MUX0**
- **MUX2**
- **DEC**
- **ALU**
- **DATA**
- **JMPZ**
- **JMPNZ**
- **JMPN**
- **JMPPZ**

Ces champs sont organisés sur 32 bits, où les 16 premiers bits représentent les données, et les 16 bits suivants sont réservés pour les différents champs.

Taycir Ben Ouirane
Albert Payol

3. Conception des Instructions

	JMP	JMPZ	JMPNZ	JMPN	JMPPZ	MUX1 bits 1	MUX1 BITS 0	MUX0	mux2	dec bit 2	dec bit 1	dec bit 0	alu bit 3	alu bit 2	alu bit 1	alu bit 0	DATA / Adress
LOAD_A #val	0	0	0	0	0	xx	X	0	0	0	0	1	0	0	0	1	VVVV
LOAD_SI #val	0	0	0	0	0	xx	X	0	0	0	0	1	0	0	0	1	VVVV
LOAD_A_DI	0	0	0	0	0	1	0	X	X	0	0	1	0	0	1	0	xxxx
LOAD_A_ADR_SI	0	0	0	0	0	1	1	X	0	0	0	1	0	0	1	0	xxxx
LOAD_B_ADR_DI	0	0	0	0	0	1	1	x	1	0	1	0	0	0	1	0	xxxx
LOAD_DI_ADR_SI	0	0	0	0	0	1	1	X	0	1	0	0	0	0	1	0	xxxx
INC_SI	0	0	0	0	0	0	1	X	X	0	1	1	0	1	0	1	xxxx
DEC_DI	0	0	0	0	0	1	0	X	X	1	0	0	0	1	1	0	XXXX
CMP_SI_A	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	XXXX
CMP_B_A	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	XXXX
JMP	1	0	0	0	0	XX	X	0	0	0	0	0	0	0	0	0	AAAA
JMPNZ	0	0	1	0	0	XX	X	0	0	0	0	0	0	0	0	0	AAAA
JMPPZ	0	0	0	0	1	XX	X	0	0	0	0	0	0	0	0	0	AAAA
LOAD_A_SI	0	0	0	0	0	0	1	x	0	0	0	1	0	0	1	0	xxxx
LOAD_B_ADR_SI	0	0	0	0	0	1	1	x	0	0	1	0	0	0	1	0	xxxx
LOAD_ADR_SI_B	0	0	0	0	0	0	0	x	0	1	0	1	0	0	1	0	xxxx
CMP_DI_A	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1	xxxx
CMP_B #valeur	0	0	0	0	0	0	0	0	x	0	0	0	1	0	1	1	VVVV
SUB_B #valeur	0	0	0	0	0	0	0	0	x	0	1	0	1	0	0	0	VVVV
JMPN	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	xxxx

Nous avons élaboré les instructions en utilisant un tableau Excel pour spécifier les différents champs et leurs valeurs associées. Voici un résumé des principales instructions et leurs fonctionnalités :

- **LOAD_A #VAL**: Charge une valeur immédiate dans le registre A en utilisant le décodeur et l'ALU.
- **LOAD_SI #VAL**: Charge une valeur immédiate dans le registre SI.
- **LOAD_A_DI**: Stocke l'adresse de DI dans le registre A.
- **INC** et **DI**: Permettent de récupérer la valeur d'une variable/pointeur (SI/DI).
- **CMP_#_#**: Instructions de comparaison utilisant l'ALU pour comparer deux valeurs.
- **JMP...**: Instructions de saut conditionnel utilisant les champs JMP correspondants et l'opérande ADRESSE DE SAUT.
- **LOAD_A_SI**: Récupère la valeur de SI et la stocke dans le registre A.
- **LOAD_B_ADR_SI**: Charge la valeur pointée par SI dans le registre B.
- **LOAD_ADR_SI_B**: Copie le registre B dans la case pointée par SI.

Taycir Ben Ouirane
Albert Payol

4. Implémentation des Programmes

Programme 1 : Vérification du Palindrome

```
        LOADSI #0
        loadaadrsi
        loaddiadrsi

debut   CMPSIA
        JMPPZ Palin


check   incsi
        loadaadrsi
        loadbadrdi
        cmpba
        JMPNZ NoPalin
        decdi
        LOADADI
        jmp debut

Palin   loada #1
        jmp fin

NoPalin loada #-1

fin     jmp fin
```

Ce programme commence par initialiser le pointeur SI à 0, puis utilise une boucle pour comparer les caractères aux extrémités de la chaîne. Si les caractères ne correspondent pas, le mot n'est pas un palindrome. Sinon, le programme continue jusqu'à ce que les pointeurs SI et DI se croisent.

Taycir Ben Ouirane
Albert Payol

Programme 2 : Conversion en Majuscules

```
        LOADSI #0
        LOADDIADRSI

LOOP    INCSI
        LOADASI
        CMPDIA
        JMPN FIN
        LOADBADRSI
        CMPB #97
        JMPPZ MAJ
        JMPN LOOP

MAJ     CMPB #122
        JMPPZ LOOP
        SUBB #32
        LOADADRSIB
        JMP LOOP

FIN     JMP FIN
```

Dans ce programme, chaque caractère de la chaîne est converti en majuscule si nécessaire. Le programme parcourt la chaîne caractère par caractère, vérifie si le caractère est en minuscule, et le convertit en majuscule si nécessaire en ajustant son code ASCII.

5. Conclusion

Ce projet a été une expérience précieuse pour comprendre le fonctionnement interne d'un processeur et la logique sous-jacente à la programmation bas niveau. En concevant des champs, des instructions et des programmes, nous avons acquis une compréhension approfondie des concepts fondamentaux de l'architecture informatique.