MINI-PROJET AVEC L'ARCHITECTURE Y86

Nathan VANBESELAERE, Sacha DUPERRET

30 avril 2023

Résumé

Rapport détaillant le travail effectué en binôme pour adapter l'architecture Y86 aux consignes.

Table des matières

1 Exercice 1

1.1 Question 1

Nous supprimons l'instruction :

```
intsig MRMOVL                    'instructionSet.get("mrmovl").icode'
```

Nous modifions également l'instruction set pour que RMMOVL ait un icode = 4 et ifun = 0, et MRMOVL un icode = 4 avec un ifun = 1. Le code source Y86 compile bien, le code hexadécimal est bien de 40 et 41 pour RMMOVL (ifun = 0) et RMMOVL (ifun = 1) respectivement.

1.2 Question 2

Nous supprimons l'ensemble des occurences de MRMOVL. Dans les cas où MRMOVL était dissocié de RMMOVL, nous ajoutons l'instruction :

```
|| icode == RMMOVL && ifun == 1
```

permettant ainsi d'exécuter correctement les instructions demandées.

2 Exercice 2

2.1 Question 1

Nous modifions le instruction set en ajoutant STRGL avec un icode = 14 et un ifun = 0. Nous testons cette nouvelle instruction en utilisant le code joint au projet (Ex1-Q1).

2.2 Question 2

Nous ajoutons l'instruction

```
intsig STRGL                    'instructionSet.get("strgl").icode'
```

permettant de donner un icode à l'instruction STRGL. Nous testons le code avec les même instructions que précédemment. Les valeurs des signaux et les opérations réalisées sont conformes à nos attentes.

2.3 Question 3

Nous ajoutons l'instruction

```
intsig STOSL                    'instructionSet.get("strgl").icode'
```

permettant de donner un icode à l'instruction STOSL. Suivant la même technique que pour la question 2 de l'execrcice 1, nous factorisons les cas commun à STRGL ifun == 0 || infun == 1.

2.4 Question 4

Nous codons un clone de strcpy en y86. Nous la testons dans le simulateur, avec le code joint à ce rapport.

3 Exercice 3

3.1 Question 1

Nous ajoutons un icode de 15 avec ifun de 4 pour le code LOOP. Le code compile avec cette nouvelle instruction.

3.2 Question 2

Nous déclarons les signaux LOOP et RECX. Sous la forme :

```
intsig LOOP                    'instructionSet.get("loop").icode'
intsig RECX                    'registers.ecx'
```

Dans new_pc nous modifions la #Taken branch : Use immediate value.

```
icode in { JXX, LOOP } && Bch : valC;
```

Nous testons nos modifications avec ce code :

```
    .pos 0
irmovl t, %esi
irmovl r, %edi
mrmovl s, %ecx

boucle: lodsl %eax
        stosl %eax
        loop boucle
        halt

.pos 0x100
s:  .long 5
t:  .long 2
    .long 3
    .long 5
    .long 7
    .long 11
r:
```

Le code compile et s'exécute sans erreur.

4 Exercice 4

4.1 Question 1

Nous ....

4.2 Question 2

4.3 Question 3