> TP Archi rendu 1

Présentation

Le but de ce tp est de simuler un processeur.

Le programme prend un fichier contenant les instructions au format "**adresse : instruction_hex**", il lit ce fichier et décode chacune des instructions pour l'éxécuter. Les instructions décodées sont ensuite sauvegarder dans un fichier.

Programme d'entrée

> format : **HHLL: AA BB CC**

Pour des raisons de lisibilité les addresses sont données sous forme HHLL et non LLHH !

L'éxécution du programme engendrera sur le fichier de sortie : le code assembleur.

Structure du projet

Processeur

La structure du processeur est déclarée dans le fichier **proc.h**, elle est définie dans le fichier **proc.c**.

Elle correspond au schéma dans le sujet de tp, c'est-à dire que les registres, bus, ram, alu, etc... sont représentés par des octets (type -> *byte**).

Les signaux sont représentés par des fonctions tout comme les micro-instructions de l'ALU.

Operations

Les opérations ASSEMBLEUR sont déclarées dans le fichier **operations.h** et sont définies dans le fichier **operations.c**.

Chaque opérations est représentée par une structure et possède une fonction qui lui est propre, son appelle correpond à son exécution.

Le décodage est effectué dans **operations.c**. Il permet de savoir qu'elle opération doit être appeler selon le code d'instruciton donné (c'est le registre IR du processeur).

Parser

Le parser permet de charger le programme donné dans la ram du processeur.

Debug

Fonctions permettant d'arrêter le code à chaque instructions et d'éxécuter une liste de commandes. Le but étant d'éxécuter pas à pas le programme tout en ayant accés aux différentes valeurs du processeurs

(registres, alu, bus, etc...).

Main

Contient l'entré du programme, tout est géré ici.

Utilisation

Mode simple

> Compilation :
>
> - $ **make**

> Lancement du programme :
>
> - $ **./main [-debug] [-v] <fichier_prgm> <fichier_sortie_assm>**
>
> exemple : ./main -v ./my_prgm.txt ./res/out.txt

> Options :
>
> > **-debug** : Permet de décoder pas à pas le programme et d'afficher les variables du processeur.
> > **-v** (verbose) : Affiche plus d'informations dans la console.

Appuyer sur entrer pour éxécuter l'opération suivante.

Le programme se termine automatiquement lorsqu'il lit 3 opérations NOP.

Vous retrouvez le code assembleur du programme dans ./res/out.txt

Voici quelques exemples d'utilisation :

Exécution simple

```
└─$ ./main -v ./prgms/calc_prgm.txt ./res/out.txt
Vérification des arguments... [ok]
Construction du processeur... [ok]
Initialisation des registres... [ok]
Chargement du programme... [ok]
Écriture d'instruction dans la RAM à l'adresse <0200>... [ok]
Sauvegarde de l'adresse de départ <0200> du programme dans PC (PCH <02> PCL <00>)... [ok]
Écriture d'instruction dans la RAM à l'adresse <0202>... [ok]
Écriture d'instruction dans la RAM à l'adresse <0204>... [ok]
Écriture d'instruction dans la RAM à l'adresse <0205>... [ok]
Écriture d'instruction dans la RAM à l'adresse <0206>... [ok]
Écriture d'instruction dans la RAM à l'adresse <ffff>... [END]
Écriture dans la RAM de 5 instructions... [ok]
Déchargement du fichier programme... [ok]
Chargement du fichier de sortie assembleur... [ok]
Chargement de l'instruction... [ ir = 51 ] [ok]
[OP] : MV
Chargement de l'instruction... [ ir = 52 ] [ok]
[OP] : MV
Chargement de l'instruction... [ ir = 62 ] [ok]
[OP] : INC
Chargement de l'instruction... [ ir = 8a ] [ok]
[OP] : ADD
Chargement de l'instruction... [ ir = 41 ] [ok]
[OP] : ST
Chargement de l'instruction... [ ir = 00 ] [ok]
[OP] : NOP
Chargement de l'instruction... [ ir = 00 ] [ok]
[OP] : NOP
Chargement de l'instruction... [ ir = 00 ] [ok]
[OP] : NOP

*****************
*** Registres ***
*****************
R[0] = 00
R[1] = 0c
R[2] = 0a
R[3] = 00
R[4] = 00
R[5] = 00
R[6] = 00
R[7] = 00
[!] Le code assembleur se trouve dans le fichier : << ./res/out.txt >>
```

Fichier de sortie assembleur

```
1   ST R1 0213
2   ADD R1 R2
3   INC R2
4   MV R2 09
5   MV R1 02
6
```

Mode debug

On lance la commande en spécifiant l'option **-debug**.

La liste de toutes les commandes du débuggeur s'affiche, ainsi qu'une invite de commande.

Appuyer simplement sur la touche *"entrée"* pour éxécuter la commande suivante.

Utiliser les commandes indiquer pour afficher des valeurs du processeur.

Vous pouver stopper l'éxécution du programme avec la commande *"quit"* ou *"q"*

*Note : pas de mode "points d'arrêt" comme indiqué dans le sujet **TP1.pdf** car on estime qu'arrêter l'éxécution du code à chaque instructions dispense l'utiilisation de cette fonctionnalité.*

Exécution en mode débogage

```
└─$ ./main -debug ./prgms/calc_prgm.txt ./res/out.txt
Vérification des arguments... [ok]
Écriture d'instruction dans la RAM à l'adresse <0200>... [ok]
Écriture d'instruction dans la RAM à l'adresse <0202>... [ok]
Écriture d'instruction dans la RAM à l'adresse <0204>... [ok]
Écriture d'instruction dans la RAM à l'adresse <0205>... [ok]
Écriture d'instruction dans la RAM à l'adresse <0206>... [ok]
Écriture d'instruction dans la RAM à l'adresse <ffff>... [END]

************************
*** Liste de commandes :
        help : affiche cette liste de commandes
        (entrée) : passe à la commande suivante
        quit | q : quit le programme.
        abus : affiche le bus d'adresse
        al : affiche Address Latch
        pc : affiche PC
        dbus : affiche le bus de données
        dl : affiche DataLatch
        ir : affiche le registre d'instruction ir
        alu : affiche la sortie de l'alu
        x : affiche X (entrée alu)
        y : affiche Y (entrée alu)
        flags : affiche les flags de l'alu

>

[OP] : MV

>

[OP] : MV

>

[OP] : INC

>abus
Bus d'adresse : #0205

>
```

Programmes

*Dans le dossier **./prgms/** !*

my_prgm.txt : Ce premier programme est celui du sujet.

calc_prgm.txt :

1. Met *2* dans *R1*
2. Met *9* dans *R2*
3. Incrémente *R2*
4. Additionne *R1 R2* (résultat dans *R1*)
5. Stocke le résultat à l'adresse **#0213**