TDs - Architecture des ordinateurs Semestre 6

7 Février, 2024

Louis Thevenet

Table des matières

1. TD1

Exemple :

```
resultat := variable1 + variable2;
```

%r	
%r2	38
%r3	39
%r4	37
%r5	12
%r6	27
%r7	39

```
set variable1 %r2
set variable2 %r2
set resultat %r3

load(%r2), %r5
load(%r3), %r6
add %r5, %r6, %r7

store %r7, [%r4]
```

1.1. Module registre

Définition 1.1.1:

```
module registre(rst, clk, areg[3..0], breg[3..0], dreg[3.0], dataIn[31..
0] : a[31..0], b[31..0], ir[31..0])
```

areg numéro du registre qu'on souhaite lire sur la sortie `a`
breg numéro du registre qu'on souhaite lire sur la sortie `b`
dreg numéro du registre dans lequel on souhaite écrire l'entrée `dataIn`
IR servira pour accéder directement au code de l'instruction courante sans passer par `a` ou `b`

Exercise 1.1.1:

```
module registres(rst, clk, areg[3..0], breg[3..0], dreg[3..0], datain[31..
0] : a[31..0], b[31..0],pc[31..0], ir[31..0])

// constantes
r0[31..0] = "00000000000000000000000000000000"
r1[31..0] = "00000000000000000000000000000001"

// où on va écrire
decoder4to16(dreg[3..0] : dsel[15..0])

// écriture
reg32_D(rst, clk, dsel[2], datain[31..0] : r2[31..0])
reg32_D(rst, clk, dsel[3], datain[31..0] : r3[31..0])
reg32_D(rst, clk, dsel[4], datain[31..0] : r4[31..0])
reg32_D(rst, clk, dsel[5], datain[31..0] : r5[31..0])
reg32_D(rst, clk, dsel[6], datain[31..0] : r6[31..0])
reg32_D(rst, clk, dsel[7], datain[31..0] : r7[31..0])

reg32_D(rst, clk, dsel[12], datain[31..0] : r12[31..0])
reg32_D(rst, clk, dsel[13], datain[31..0] : r13[31..0])
reg32_D(rst, clk, dsel[14], datain[31..0] : r14[31..0])
reg32_D(rst, clk, dsel[15], datain[31..0] : r15[31..0])


// qu'est-ce qu'on met dans a ?
decoder4to16(areg[3..0] : asel[15..0])
a[31..0] = r0[31..0] * asel[0]
            + r1[31..0] * asel[1]
            + r2[31..0] * asel[2]
            + r3[31..0] * asel[3]
            + r4[31..0] * asel[4]
            + r5[31..0] * asel[5]
            + r6[31..0] * asel[6]
            + r7[31..0] * asel[7]

            + r12[31..0] * asel[12]
            + r13[31..0] * asel[13]
            + r14[31..0] * asel[14]
            + r15[31..0] * asel[15]

decoder4to16(breg[3..0] : bsel[15..0])
b[31..0] = r0[31..0] * bsel[0]
            + r1[31..0] * bsel[1]
            + r2[31..0] * bsel[2]
            + r3[31..0] * bsel[3]
            + r4[31..0] * bsel[4]
            + r5[31..0] * bsel[5]
            + r6[31..0] * bsel[6]
            + r7[31..0] * bsel[7]

            + r12[31..0] * bsel[12]
            + r13[31..0] * bsel[13]
            + r14[31..0] * bsel[14]
            + r15[31..0] * bsel[15]


pc[31..0] = r14[31..0]
ir[31..0] = r15[31..0]
end module
```

1.2. Module UAL (Unité arithmétique et logique)

Exercise 1.2.1:

Opérations :
sigext 1100 (signe extension)
add 0000
sub 0001

```
module ual(a[31..0], b[31..0], cmd[3..0] : s[31..0], N, Z, V, C)

addsub32(a[31..0], b[31..0], cmd[0] : saddsub[31..0], V, C)

sext[23..0] = a[23..0]
sext[31..24] = a[23] *"11111111"

s[31..0] = saddsub[31..0] * /cmd[3] * /cmd[2] * /cmd[1]
            + sext[31..0] * cmd[3] * cmd[2] * /cmd[1] * /cmd[0]

Z = "tous les bits à 0"
N = /s[31]
end module
```

1.3. Les instructions et le séquenceur

```
module sequenceur(rst, clk, ir[31..0], N, Z, V, C : fetch, decode, pcplus1,
areg[3..0], breg[3..0], dreg[3..0], ualcmd[3..0], dbusin[1..0], write, setflags)
```

```
areg[3..0] = fetch * "1110"
                + decode2pcplus1 * ir[23..20]
                + pcplus1 * "1110" // l'adresse de PC

breg[3..0] = fetch * "0000"
                + decode2pcplus1 * ir[19..16]
                + pcplus1 * "0001" // l'adresse de 1

dreg[3..0] = fetch * "1111"
                + decode2pcplus1 * ir[27..24]
                + pcplus1 * "1110" // l'adresse de PC pour l'addition

ualcmd[3..0] = fetch * "0000"
                + decode2pcplus1 * ir[31..28]
                + pcplus1 * "0000"

dbusin[1..0] = fetch * "10"
            + decode2pcplus1 * "01"
            + pcplus1 * "01"

write = fetch * "0"
            + decode2pcplus1 * "0"
            + pcplus1 * "0"

setflags = decode2pcplus1
end module
```