Architecture des ordinateurs - Architecture MIPS

Didier Teifreto

Université de Franche Comté

19 octobre 2015

Instructions implémentées

1. **Instructions de chargement rangement** : lw et sw .
2. **Instructions arithmétiques et logiques** : add , sub , and , nor , or ,xor , addi , ori , andi , lui
3. **Instructions de branchement** : bne

lw et sw : lw *rt* , *Imm16* (*rs*)

add , sub , and , or ,xor .... : add *rd* , *rs* , *rt*

addi , ori , andi ... : addi *rt* , *rs* ,*Imm16*

bne : bne *rs* , *rt* ,*Imm16*

Rappel : Codage instructions MIPS

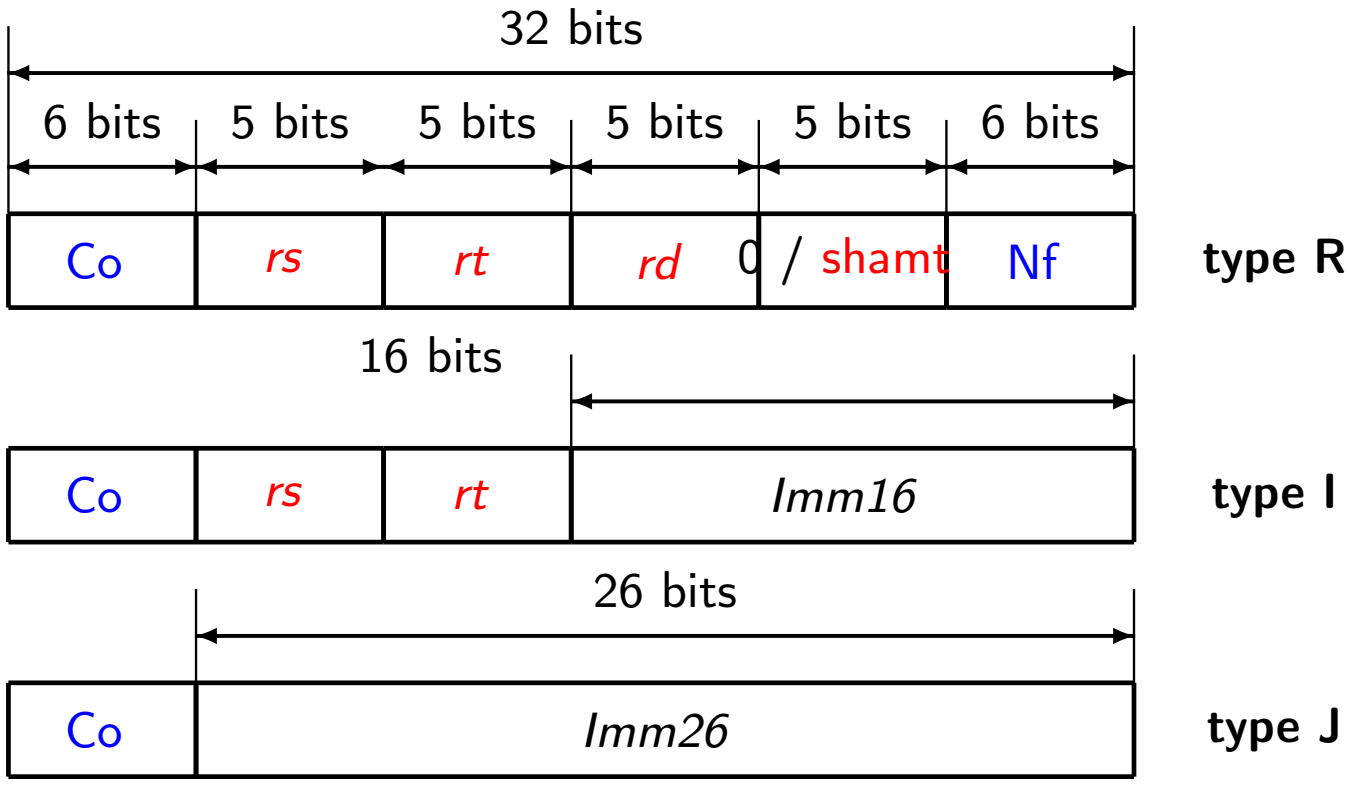


Figure : Codage des instructions

Type J non implémenté.

Décomposition instruction 32 bits

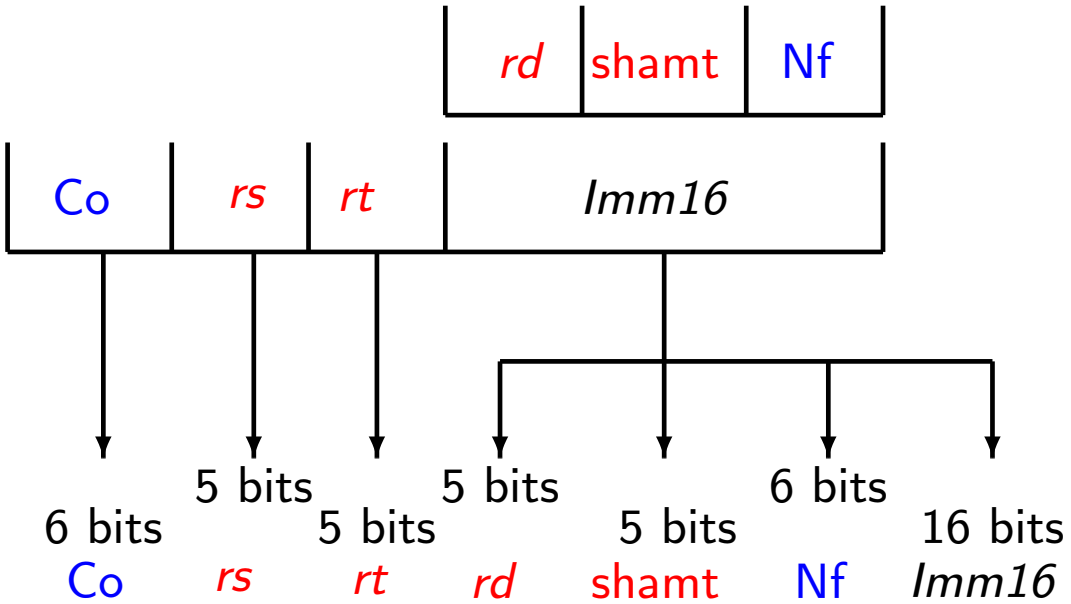


Figure : Décomposition instruction

- Pour décoder l'instruction nous devons fabriquer ces signaux.
- Si l'instruction du type R, le signal *Imm16* inutilisé.
- Si l'instruction du type I, *rd* , shamt et Nf inutilisés.

Traitement des instructions du type R - 1

1. Charger l'instruction et calculer l'adresse de la suivante *cp* +4
2. Décomposer l'instruction en différent champs (Co *rs* ...)
3. Charger les valeurs des opérandes depuis les registres *rs* et *rt* .
4. En fonction de Co et de Nf déterminer l'opération à effectuée.
5. Calculer le résultat *rs* opération *rt*
6. ranger le résultat dans le registre ayant pour numéro *rd* .

Traitement des instructions du type R - Exemple

Soit l'opération d'addition add *$t0* , *$t1* , *$t2*

1. Charger l'instruction : **0x**012a 4020 et calculer l'adresse de la suivante : **0x**0040 0004 (si instruction en **0x**0040 0000)
2. Charger les valeurs des opérandes depuis les registres *rs* et *rt* . Les numéros de *rs* = 9 et de *rt* = 10
3. En fonction de Co et de Nf déterminer l'opération à effectuée. (Co =0 et Nf =20)
4. Calculer le résultat *$t1* + *$t2*
5. ranger le résultat dans le registre ayant pour numéro *rd* =8

Traitement des instructions arithmétique et logique du type I - 1

1. Charger l'instruction et calculer l'adresse de la suivante *cp* +4
2. Décomposer l'instruction en différent champs (Co *rs* ...)
3. Charger les valeurs des opérandes depuis le registre *rs* .
4. En fonction de Co déterminer l'opération à effectuée.
5. Calculer le résultat *rs* opération *Imm16*
   - Extension non signée pour opérations logiques
   - Extension signée pour opérations arithmétiques
6. ranger le résultat dans le registre ayant pour numéro *rt* .

Traitement des instructions arithmétique et logique du type I - exemple

Soit l'opération d'addition addi *$t2* , *$t1* ,**0x**1234

1. Charger l'instruction : **0x**212a 1234 et calculer l'adresse de la suivante : **0x**0040 0008
2. Charger les valeurs des opérandes depuis le registre *rs* . Le numéro de *rs* = 9
3. En fonction de Co déterminer l'opération à effectuée. (Co =8)
4. Calculer le résultat *$t1* + **0x**0000 1234
5. ranger le résultat dans le registre ayant pour numéro *rt* =10

Traitement des instructions de chargement I - 1

1. Charger l'instruction et calculer l'adresse de la suivante *cp* +4
2. Décomposer l'instruction en différent champs (Co *rs* ...)
3. Charger les valeurs des opérandes depuis le registre *rs*
4. En fonction de Co et de Nf déterminer l'opération à effectuée.
5. Calculer l'adresse mémoire de la donnée à lire *rs* + *Imm16*
6. ranger le résultat dans le registre ayant pour numéro *rt* la valeur mémoire d'adresse *rs* + *Imm16*

Traitement des instructions de chargement I - exemple

Soit l'opération de chargement lw *$t2* ,*Imm16* (regt1)

1. Charger l'instruction : **0x**8d2a 1234 et calculer l'adresse de la suivante : **0x**0040 000C
2. Charger les valeurs des opérandes depuis le registre *rs* . Le numéro de *rs* = 9
3. En fonction de Co déterminer l'opération à effectuée. (Co =23)
4. Calculer l'adresse mémoire *$t1* + **0x**0000 1234
5. ranger le le contenu de l'adresse mémoire dans le registre ayant pour numéro *rt* =10

Traitement des instructions de rangement I - 1

1. Charger l'instruction et calculer l'adresse de la suivante *cp* +4
2. Décomposer l'instruction en différent champs (Co *rs* ...)
3. Charger les valeurs des opérandes depuis les registres *rs* et *rt*
4. En fonction de Co déterminer l'opération à effectuée.
5. Calculer l'adresse mémoire de la donnée à lire *rs* + *Imm16*
6. ranger le contenu du registre *rt* dans valeur mémoire d'adresse *rs* + *Imm16*

Traitement des instructions de rangement I - exemple

Soit l'opération de rangement sw *$t2* ,Imms(*$t1*)

1. Charger l'instruction : **0x**ad2a 1234 et calculer l'adresse de la suivante : **0x**0040 0010
2. Charger les valeurs des opérandes depuis les registres *rs* . Les numéros de *rs* = 9 et *rt* =10
3. En fonction de Co et de Nf déterminer l'opération à effectuée. (Co =2b)
4. Calculer l'adresse mémoire *$t1* + **0x**0000 1234
5. ranger le contenu du registre *rt* dans la mémoire d'adresse *rs* + *Imm16*

Traitement des instructions de branchement I - 1

1. Charger l'instruction et calculer l'adresse de la suivante cp +4
2. Décomposer l'instruction en différent champs (Co rs ...)
3. Charger les valeurs des opérandes depuis les registres rs et rt
4. En fonction de Co et de Nf déterminer l'opération à effectuée.
5. Calculer l'adresse instruction cible cp +4 + ($Imm16$ << 2)
6. Si $rs \neq rt$ alors $cp \leftarrow cp$ +4 + ($Imm16$ << 2) sinon $cp \leftarrow cp + 4$

Traitement des instructions de branchement I - exemple

Soit l'opération de rangement bne *$t1* , *$t2* ,*Imm16*

1. Charger l'instruction : **0x**152a 1234 et calculer l'adresse de la suivante : **0x**0040 0018
2. Charger les valeurs des opérandes depuis les registres *rs* . Les numéros de *rs* = 9 et *rt* =10
3. En fonction de Co et de Nf déterminer l'opération à effectuée. (Co =5)
4. Calculer l'adresse instruction cible *$t1* + (**0x**0000 1234 << 2) = **0x**0040 48EC
5. Si *rs* ≠ *rt* alors *cp* ← **0x**0040 48EC sinon *cp* ← **0x**0040 0018

Registre 32 bits

- Data : Donnée à écrire dans registre
- Output : Donnée contenue dans le registre
- *en* enable : si 0 le signal d'horloge n'est pas pris en compte
- 0 : permettent de positionner la sortie à 0
- ^ : signal d'horloge (front montant : 0 à 1, ou descendant : 1 à 0)
- Possibilités de paramétrer le fonctionnement de la bascule : niveau, front montant, front descendant

Registres architecture mips

Utilisations des registres par les instructions implémentées :

- 3 registres : 2 registres de données et un résultat
- Immédiate : 1 registre de donnée et un résultat
- Chargement et rangement : 1 registre donnée et un résultat
- Branchement : 2 registres de données

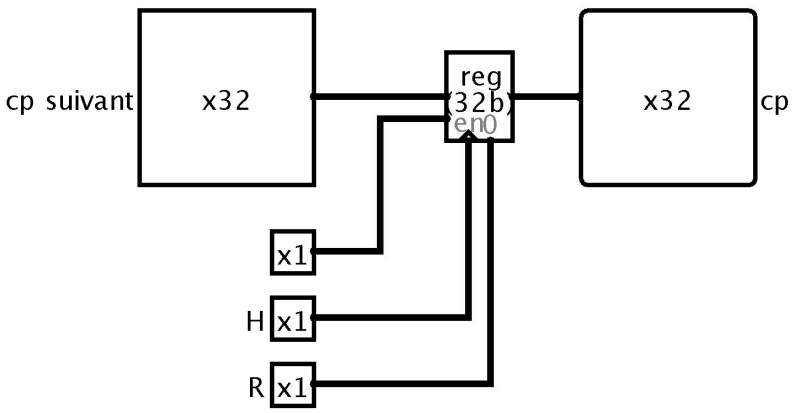
Dans le pire des cas, nous avons besoin de

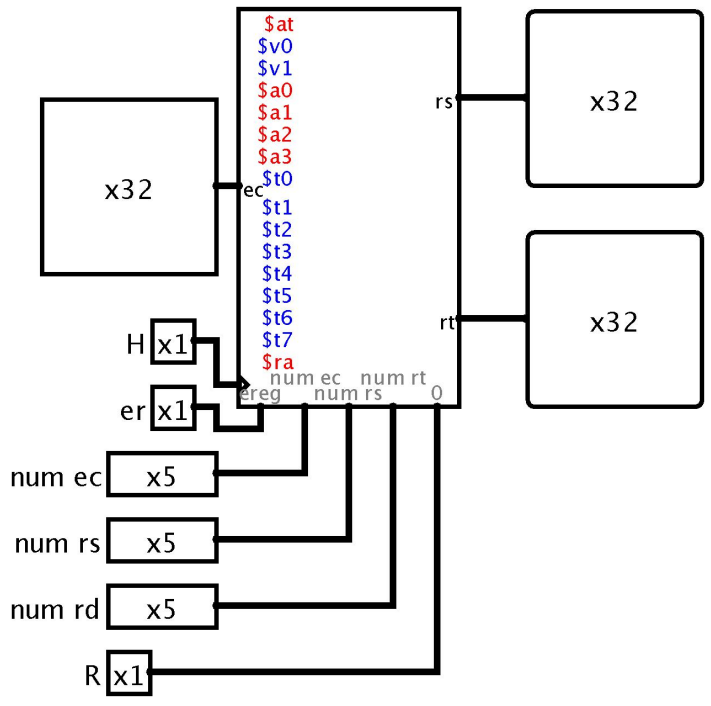
- 2 valeurs contenues dans les registres pour les données : 2 ports en lecture
- 1 valeur résultat à écrire dans un registre : 1 port en écriture

Banc de registres - suite

- Entrées :
  - Numéro des trois registres (2 pour lecture, 1 en écriture)
  - Signal d'écriture du registre destination
  - Signaux d'horloge et de remise à 0
  - Chargement au front montant du signal d'horloge $h \uparrow$
- Sorties :
  - Valeur de deux registres opérandes *rs* et *rt*

Registre 32 bits et Banc de registres





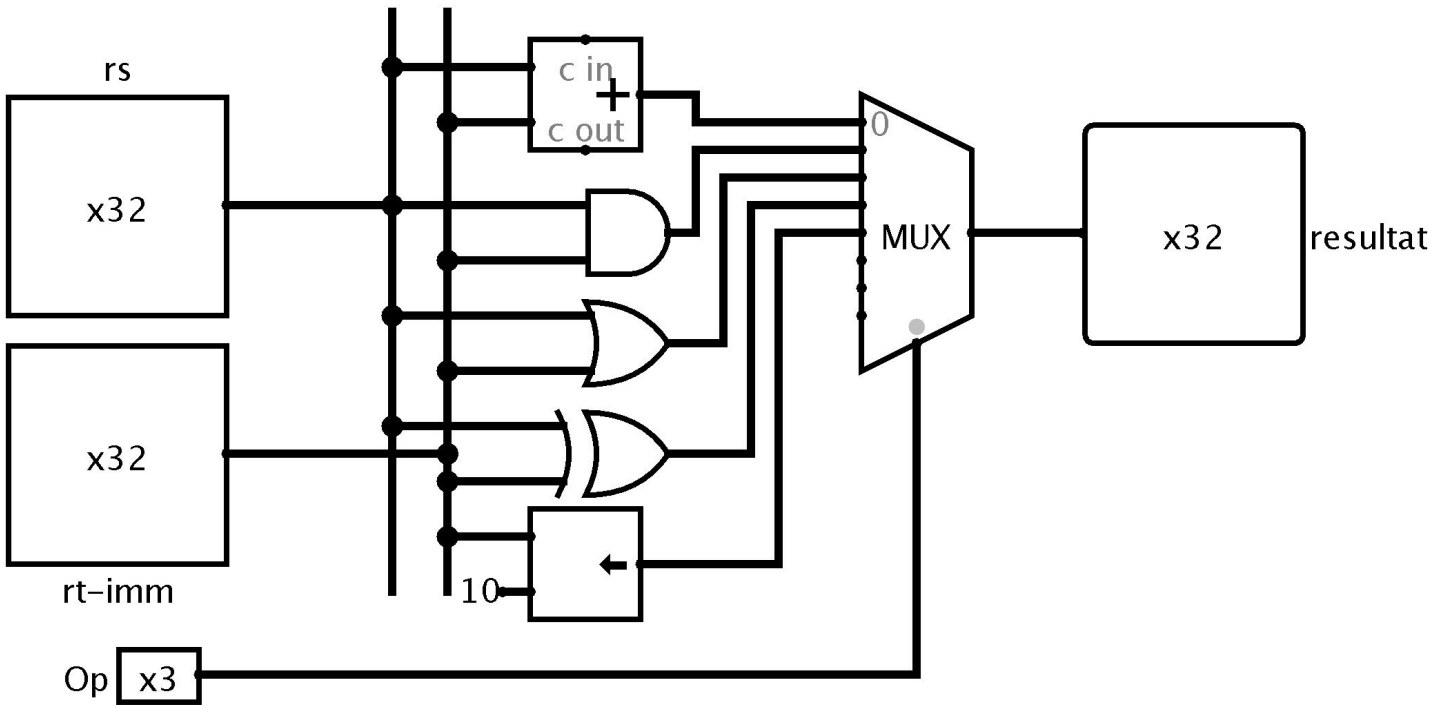
Unité arithmétique et logique UAL (ALU) - 1

- Circuit de calcul du résultat de l'instruction
- Une opération parmi un ensemble possible
- Signal Op permet de sélectionner une opération
- Signal A et B non interchangeable.

Ordre aléatoire pour le projet....

Op	Opération
0	A & B
1	A | B
2	A ^ B
3	A + B
4	B << 16

Unité arithmétique et logique UAL (ALU) - 2



Mémoire de données et d'instructions

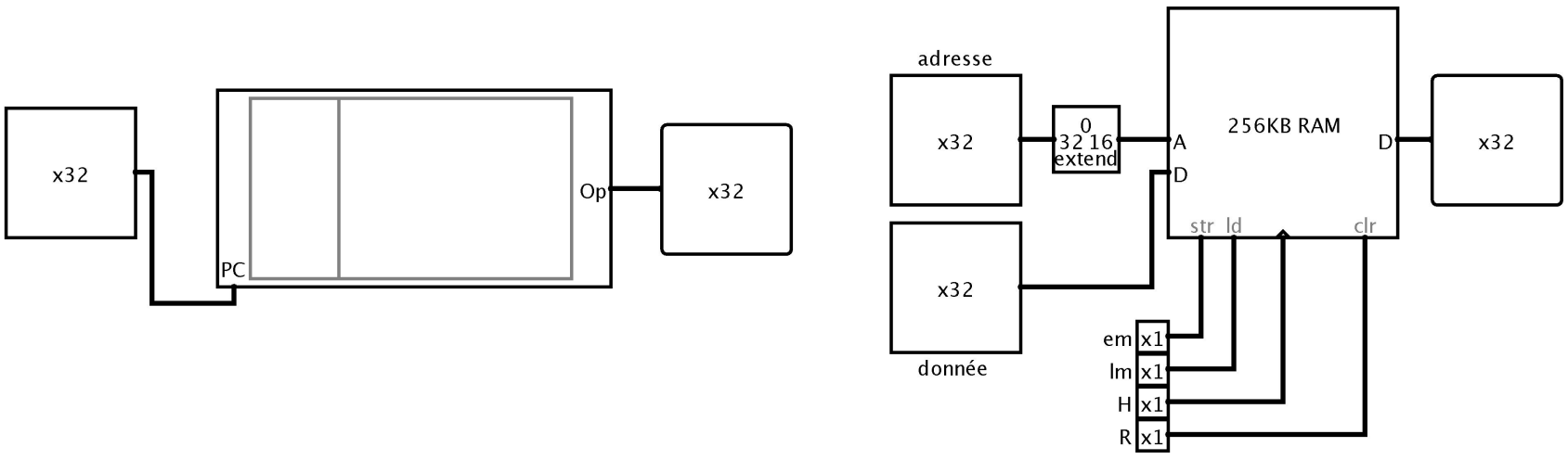
Donnée

- Bus adresse 16 bits 64Ko x 4 = 256 K Octets
- Mémoire 32 bits
- signal pour valider l'écriture mémoire au signal d'horloge ↑

Instruction

- Bus adresse 32 bits
- Données 32 bits
- Mémoire lecture seule (ROM) persistante
- Doit contenir code sans étiquette (nous devons spécifier la distance de branchement (en nombre d'octets, pas d'étiquette)

Mémoire de données et d'instructions



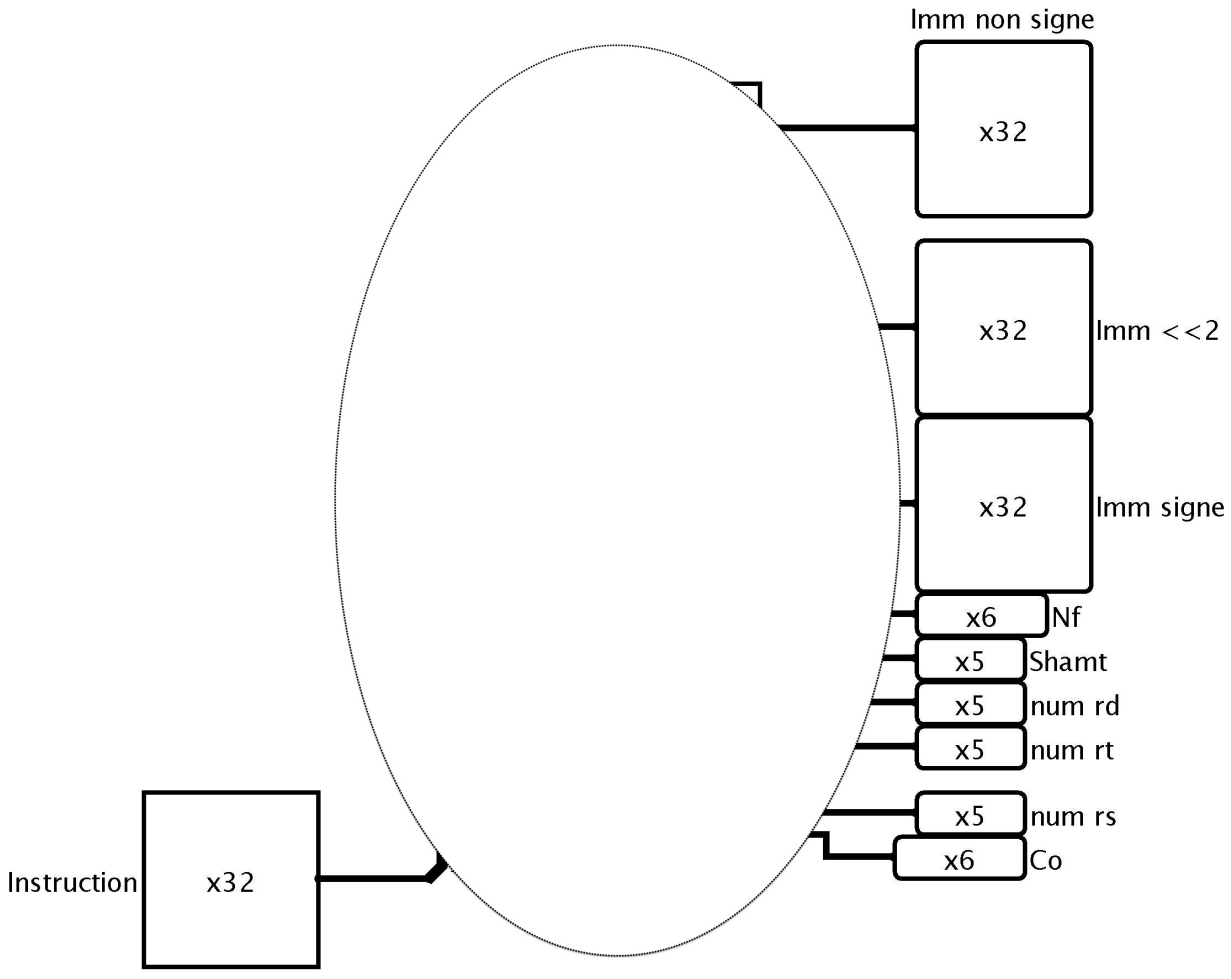
Adresse instruction - registre *cp* et décodage

- Valeur codée sur 32 bits
- Première instruction du programme est en 0 (mais pas en **0x**0040 0000 comme MARS)
- Changement de valeur au front descendant du signal $h\downarrow$
- Calcul de *cp* +4

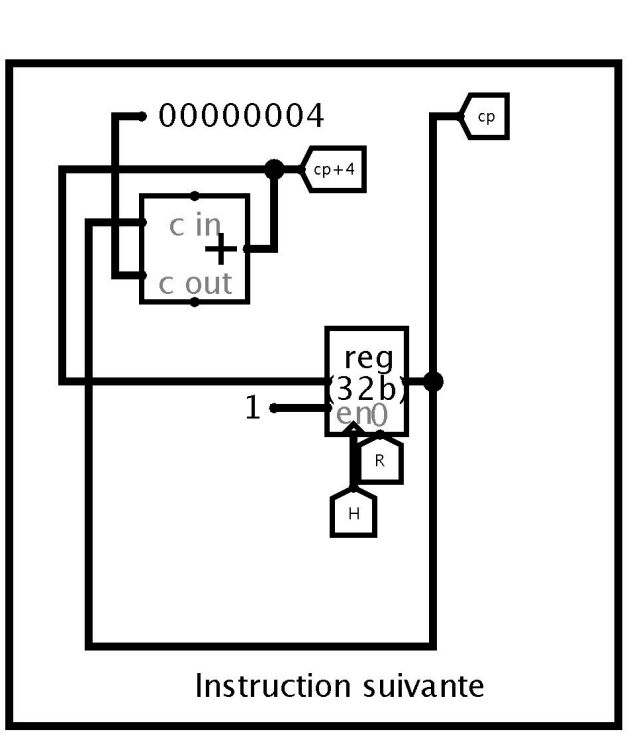
Décodage

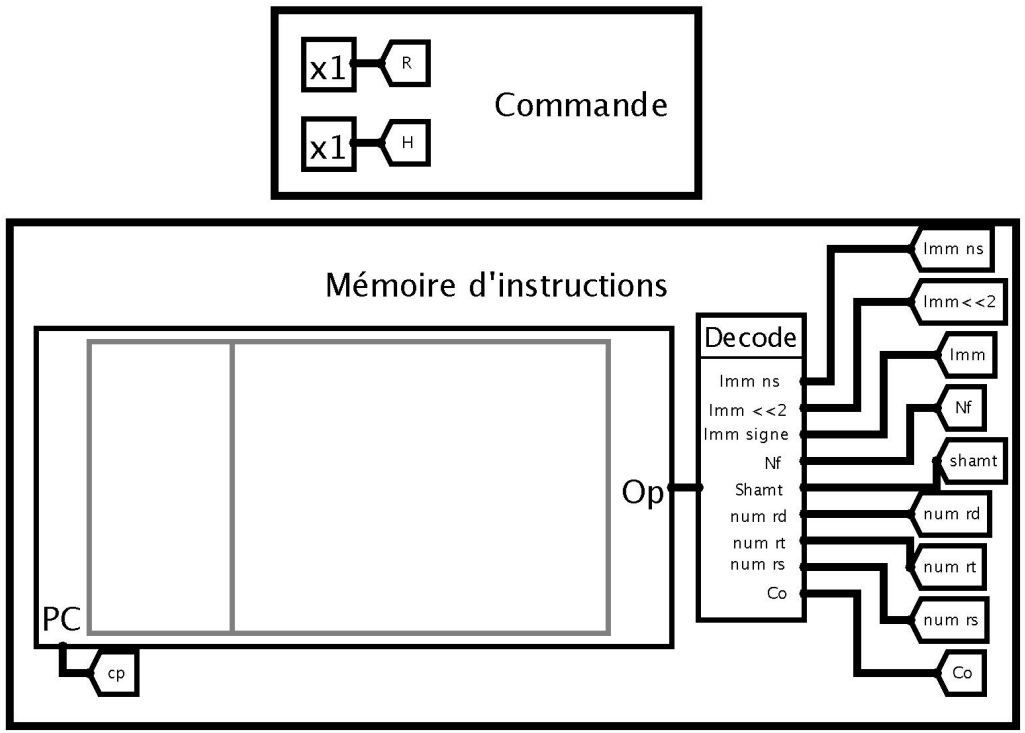
- Sortir les différents champs de l'instructions
- Interpréter les valeurs en fonction du codage

Décodage des instructions



Compteur de programme, mémoire d'instructions et décodages des instructions

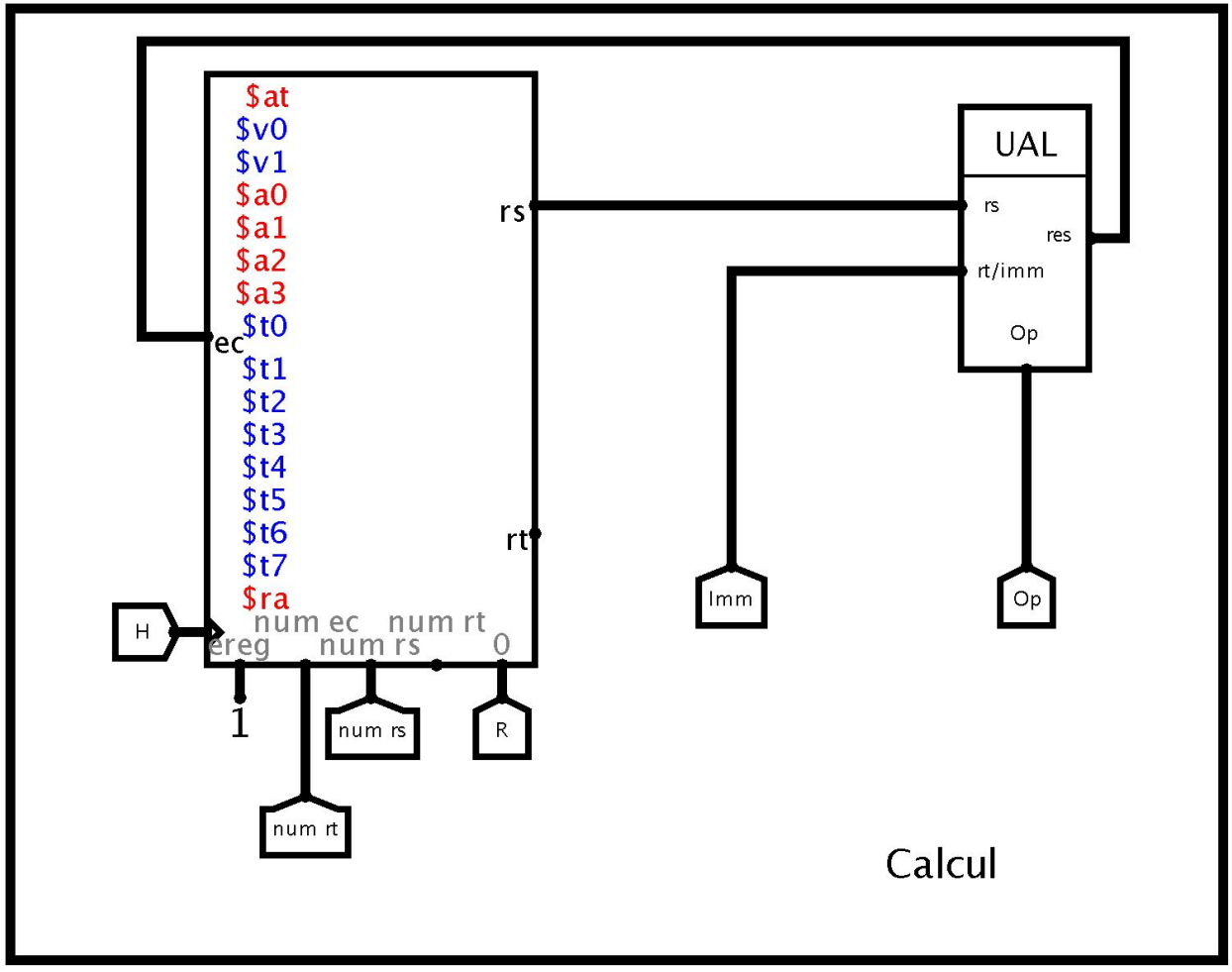




Instruction arithmétiques logiques immédiate

- Nous regroupons les schémas précédents
- L'adresse contenue dans le registre *cp* est envoyée à la mémoire d'instructions, *cp* +4 est calculé
- L'opérande A est dans le registre de numéro *rs* , l'opérande B est la valeur immédiate *Imm16*
- Le registre résultat est contenu dans registre de numéro registre *rt* .
- La donnée est stockée dans le banc de regsitre au front montant du signal d'horloge
- Nous passons à l'instruction suivante au front descendant du signal d'horloge

Instruction arithmétiques logiques immédiate -2

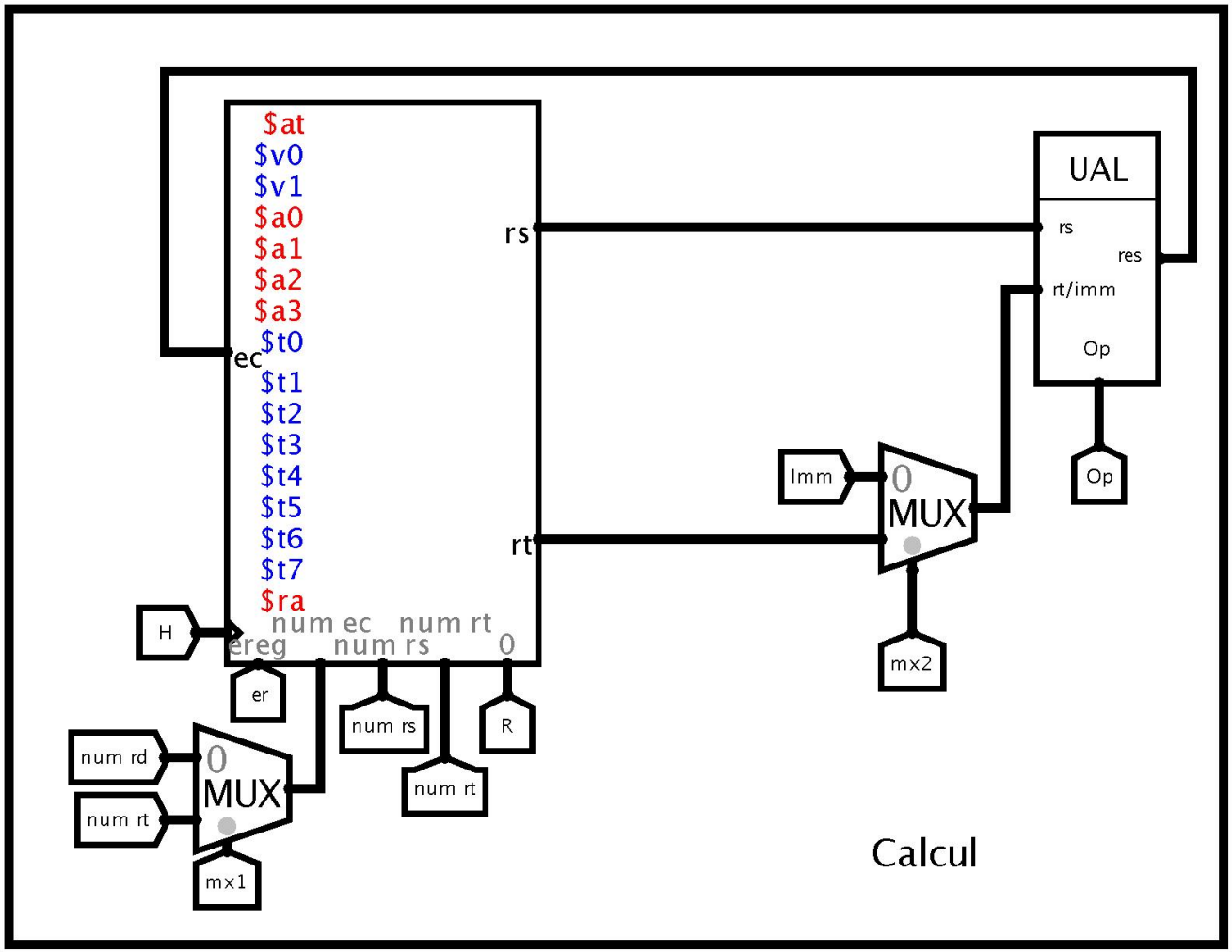


Instruction arithmétiques logiques 3 registres

- Les numéros des registres sources *rs* et *rt* sont envoyés au banc de registres.
- Les valeurs des registres sources *rs* et *rt* sont envoyées à l'UAL qui effectue le calcul
- Le résultat est stocké dans le registre de numéro *rd*

- Nous devons ajouter un multiplexeur pour sélectionner la valeur immédiate ou contenu du registre *rs*
- Nous devons ajouter un multiplexeur pour sélectionner le registre résultat *rt* ou *rd*

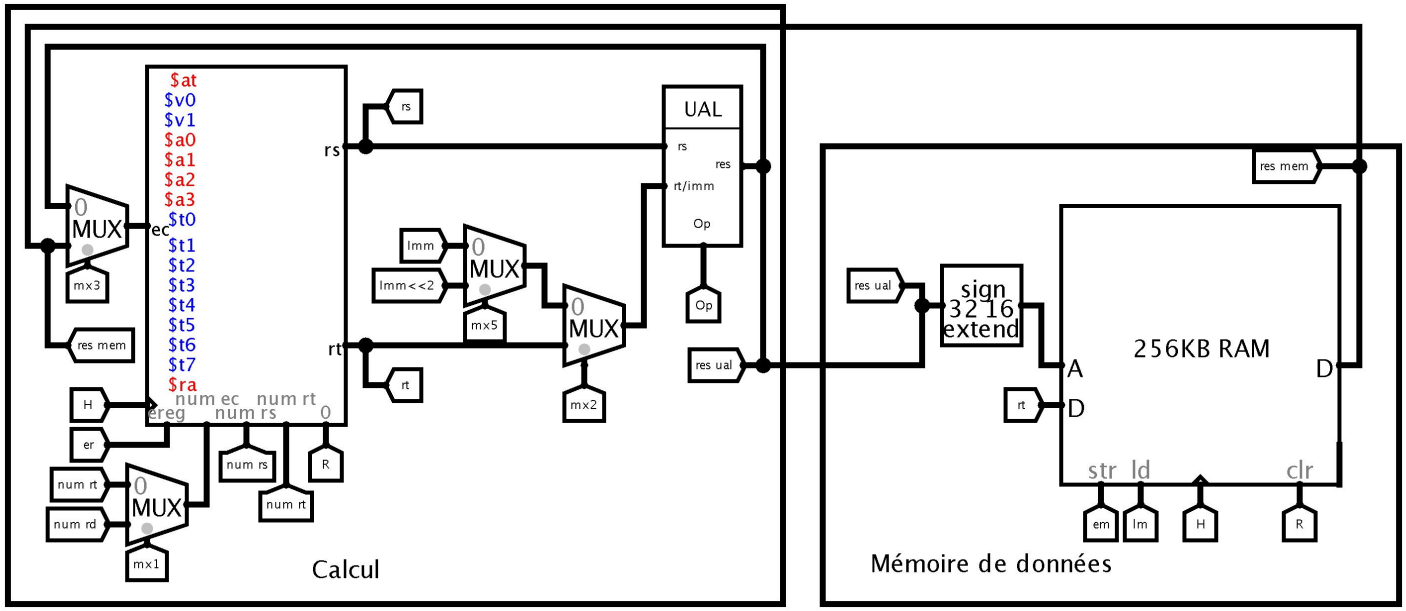
Instruction arithmétiques logiques 3 registres - 2



Chargement et rangement - 1

- L'adresse de la donnée est calculée par l'UAL (opération +) est envoyée à la mémoire.
- Nous ajoutons un multiplexeur qui sélectionne la valeur écrite dans le banc de registre en *rt* . Soit la sortie de l'UAL, soit la valeur lue en mémoire (chargement lw)
- La valeur contenue dans le registre *rt* est envoyée à la mémoire pour écriture (rangement sw)

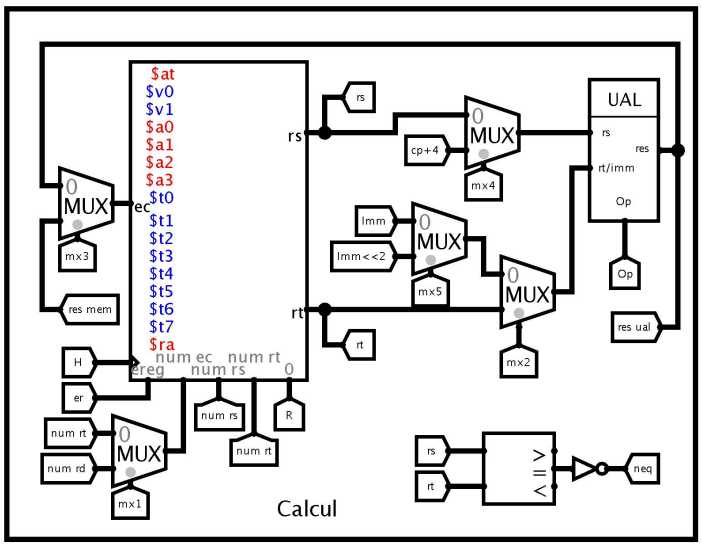
Chargement-rangement - 2

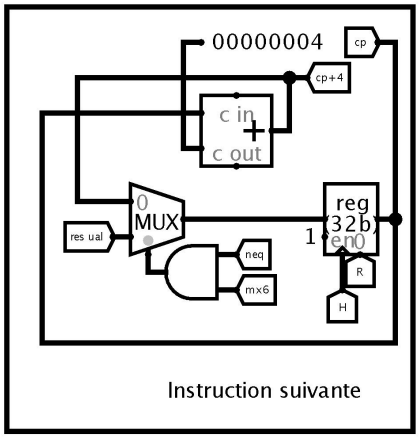


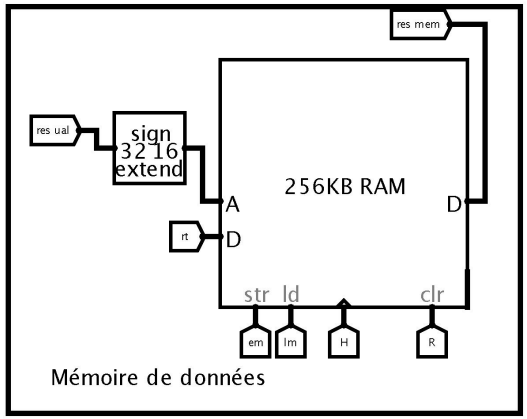
Branchement conditionnel - 1

- L'UAL effectue le calcul de la comparaison entre les valeurs contenues dans les registres *rs* et *rt* (*rs* - *rt*)=0 (Op=6, opération -)
- Nous devons calcule l'adresse de l'instruction cible (*cp* +4) + (*Imm16* << 2)
- Nous ajoutons un multiplexeur pour sélectionner l'adresse de l'instruction suivante ou cible en fonction du résultat de la comparaison.

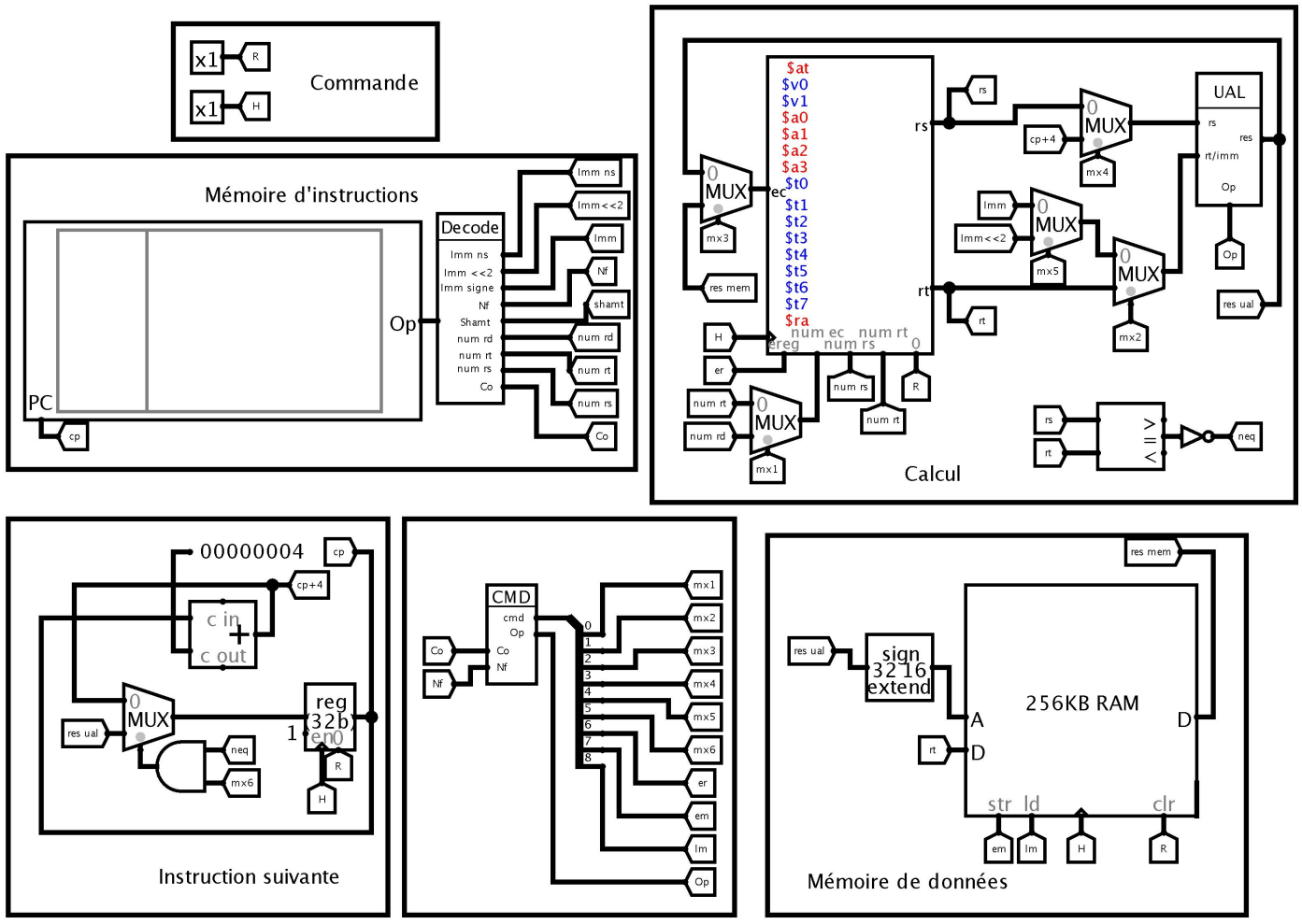
Branchement conditionnel - 2







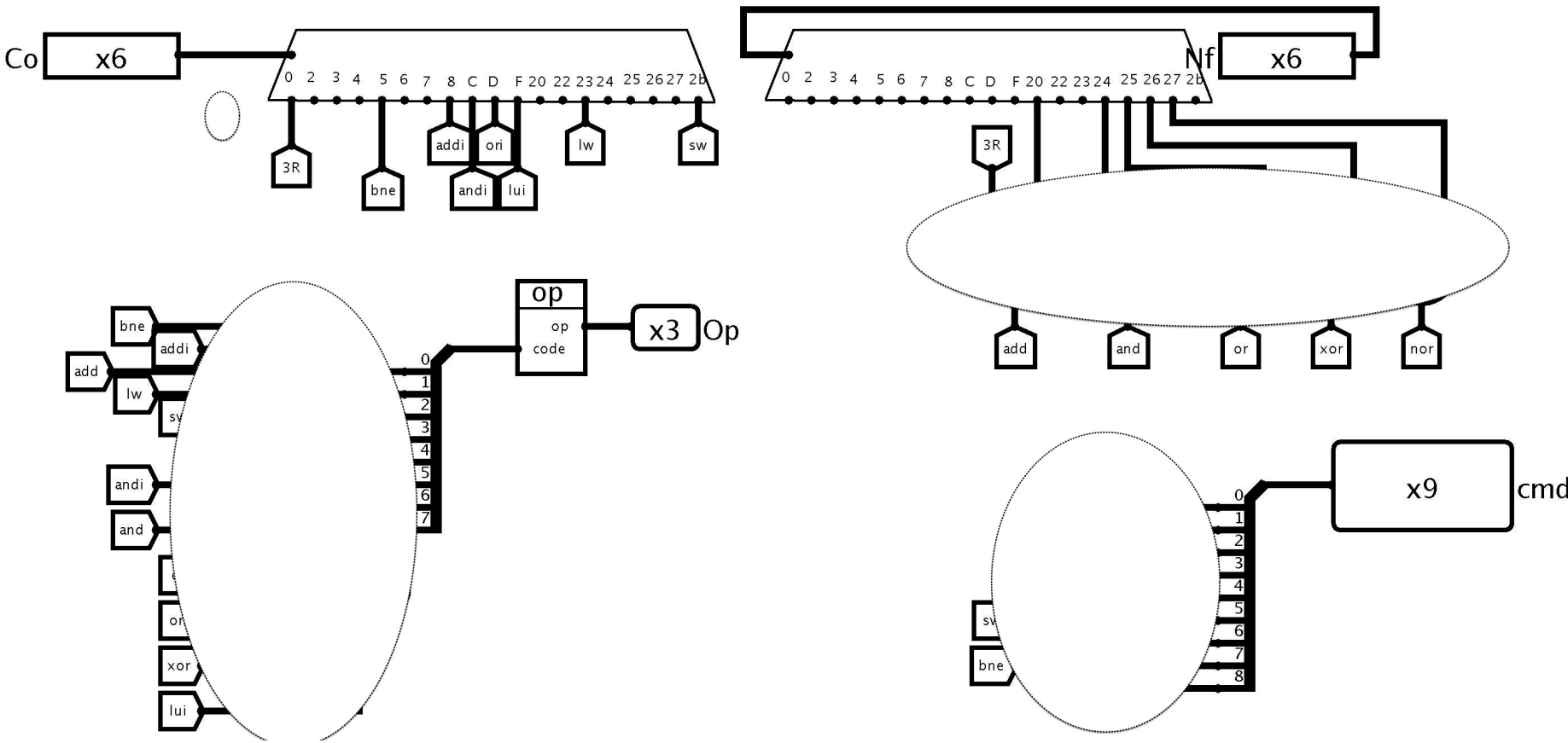
L'ensemble


x1
R
x1
H
Commande
Mémoire d'instructions
PC
cp
Op
Decode
Imm ns
Imm <<2
Imm signe
Nf
Shamt
num rd
num rt
num rs
Co
Imm ns
Imm<<2
Imm
Nf
shamt
num rd
num rt
num rs
Co
$at
$v0
$v1
$a0
$a1
$a2
$a3
$t0
$t1
$t2
$t3
$t4
$t5
$t6
$t7
$ra
num ec
num rt
reg
num rs
0
MUX
mx3
res mem
mx1
mx2
mx4
mx5
mx6
cp+4
Imm
Imm<<2
rs
rt
H
er
num rt
num rd
num rs
R
UAL
rs
res
rt/imm
Op
res ual
Calcul
>
=
<
neq
00000004
cp
cp+4
c in
c out
reg
(32b)
1 en 0
res ual
neq
mx6
R
H
Instruction suivante
CMD
cmd
Op
Co
Nf
mx1
mx2
mx3
mx4
mx5
mx6
er
em
lm
Op
res mem
res ual
sign
32 16
extend
256KB RAM
A
D
rt
D
str
ld
clr
em
lm
H
R
Mémoire de données

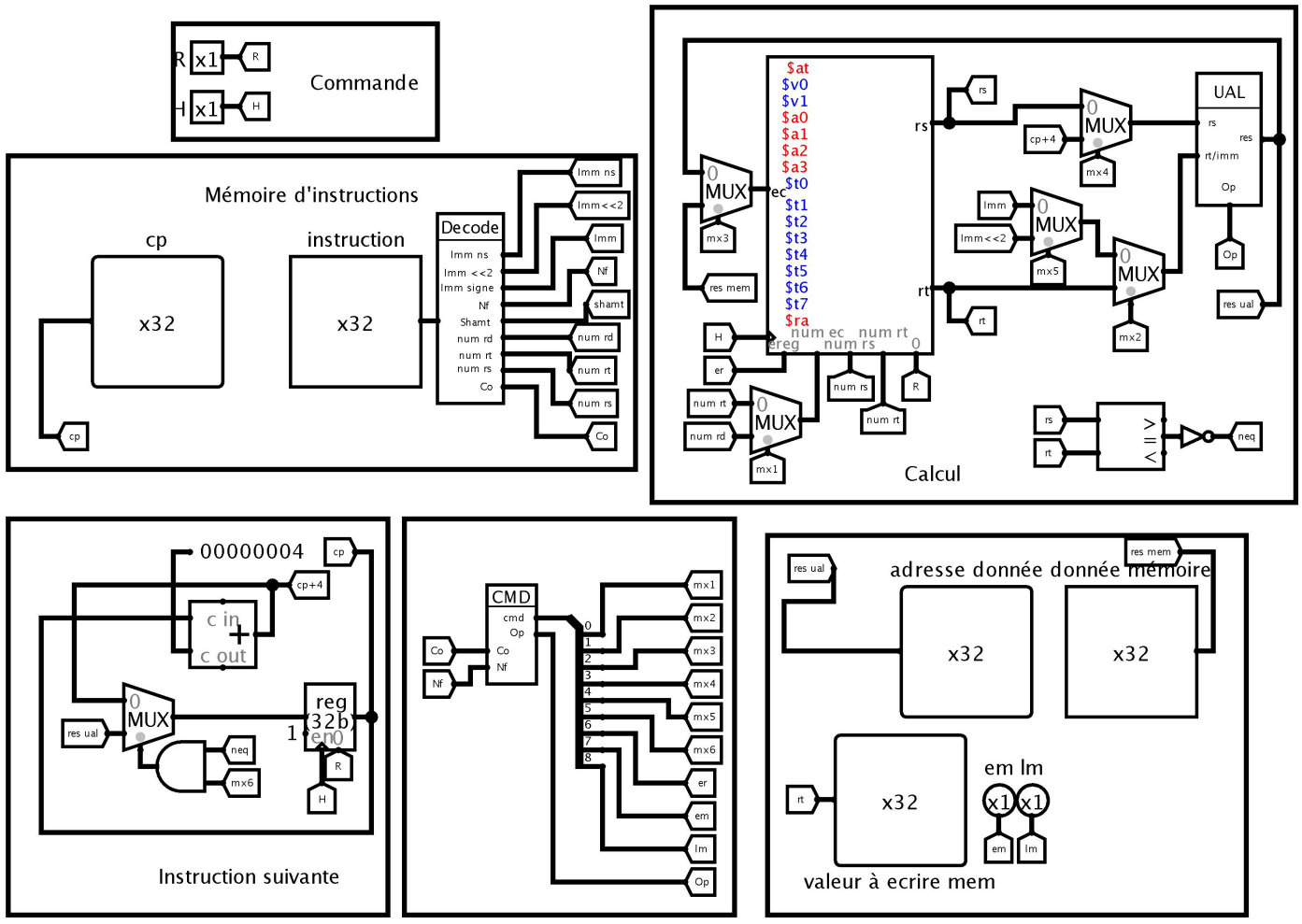
Contrôle

- A partir de Nf et Co nous devons générer toutes les commandes des multiplexeurs et de l'UAL
- Nous devons générer les signaux suivants :
  1. Les commandes des multiplexeurs
  2. Les commandes d'écriture et de lecture des registres et de la mémoire
  3. ...

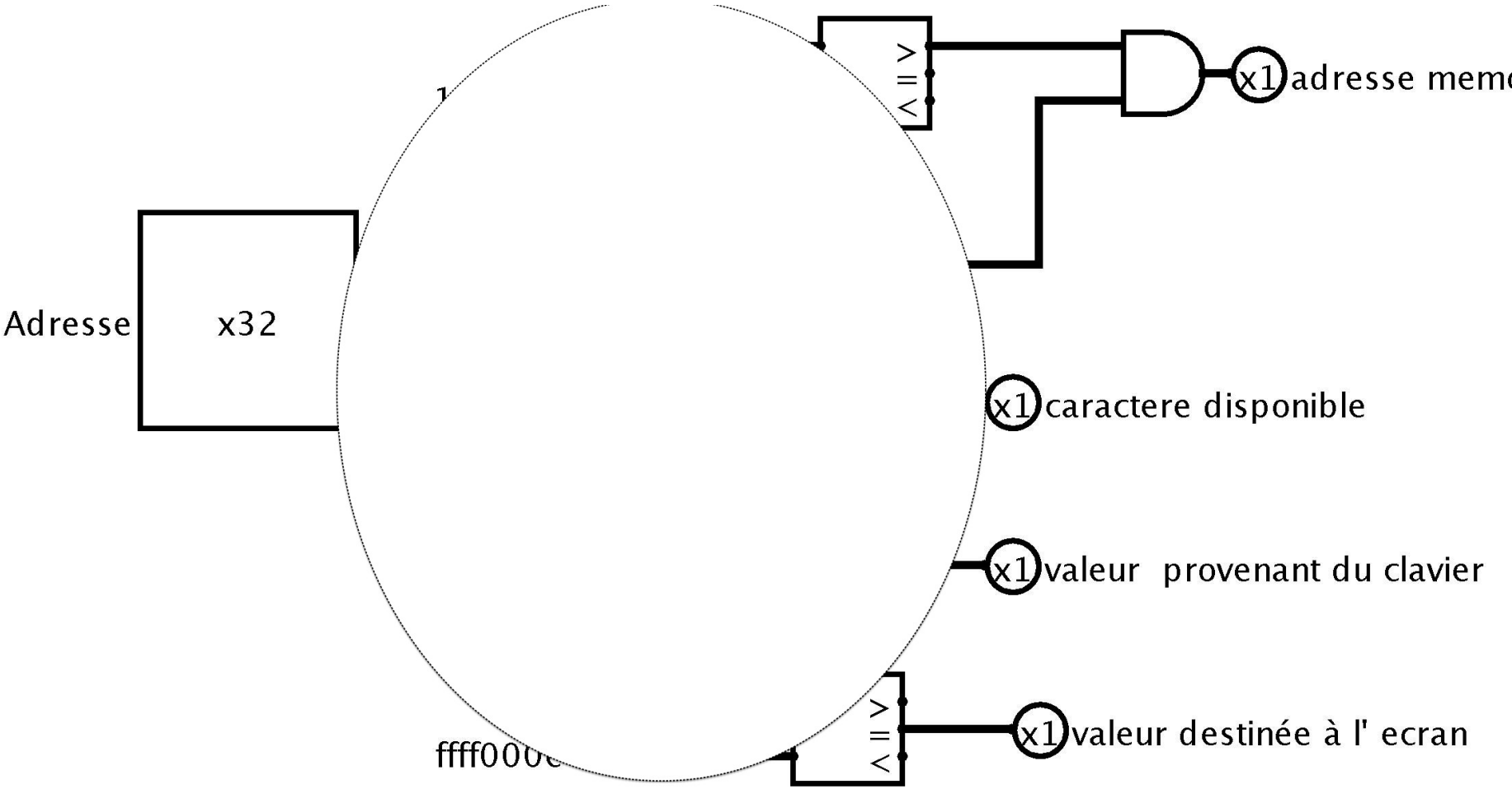
Contrôle - 2


Co
x6
0 2 3 4 5 6 7 8 C D F 20 22 23 24 25 26 27 2b
3R
bne
addi
ori
andi
lui
lw
sw
Nf
x6
add
and
or
xor
nor
op
op
code
x3
Op
bne
addi
add
lw
andi
and
ori
xor
lui
0
1
2
3
4
5
6
7
x9
cmd
sw
bne
8

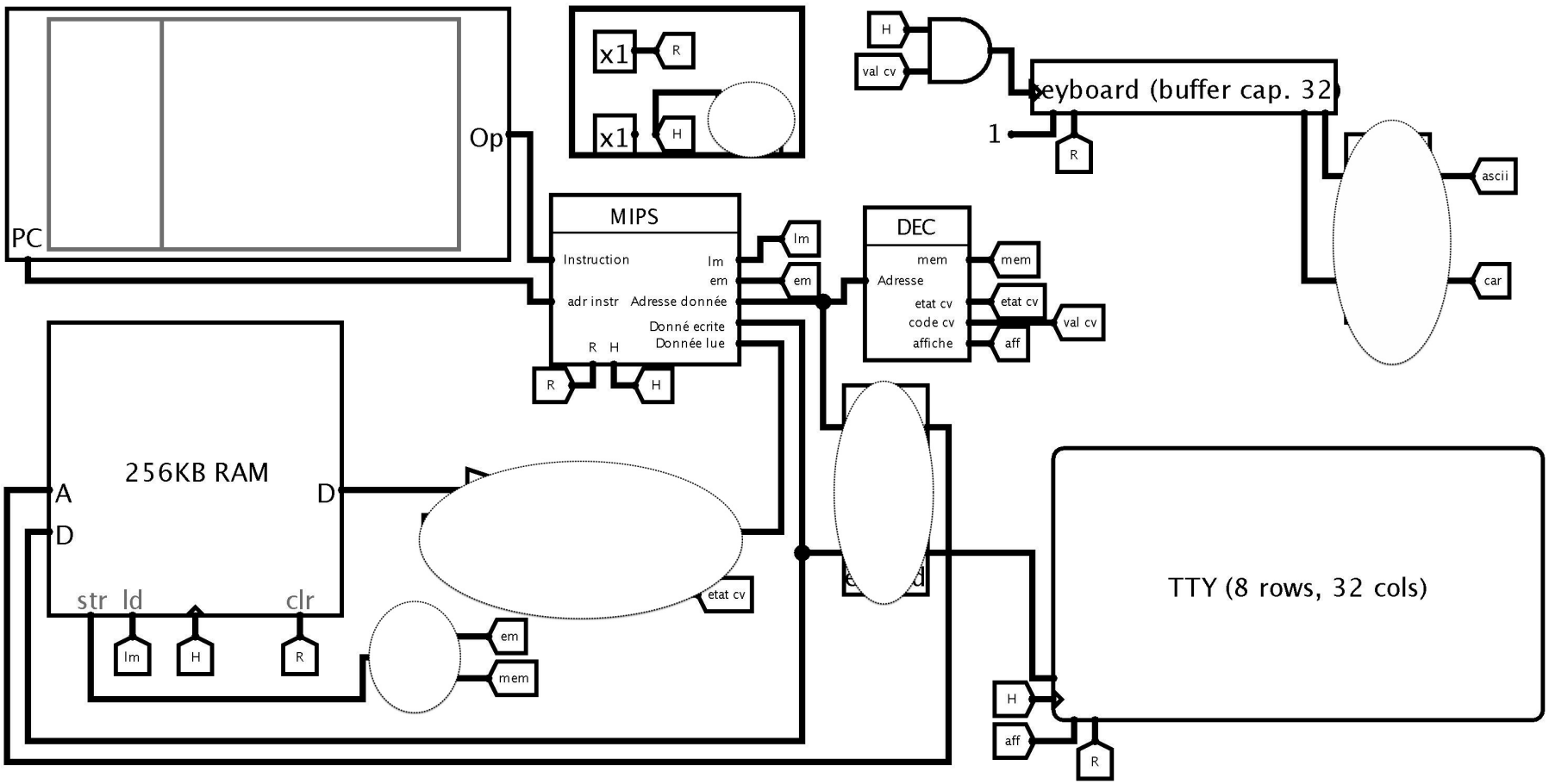
Mips pour les entrées sorties



Décodage d'adresse



Mips et entrées sorties



A faire dans le projet

- Extension non signée opérations logiques
- Adresse mémoire de données en **0x**1001 0000
- shamt pour les décalages
- Branchement autres que bne : bgtz blez beq jr jal