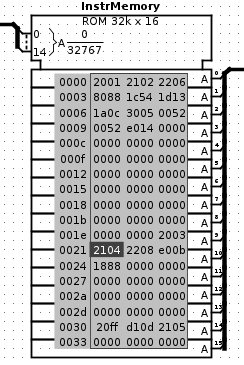
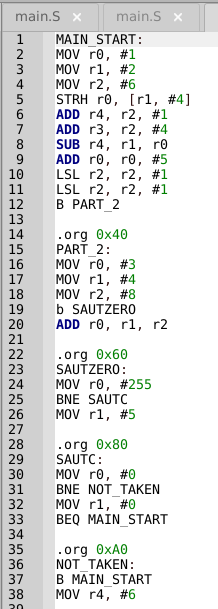
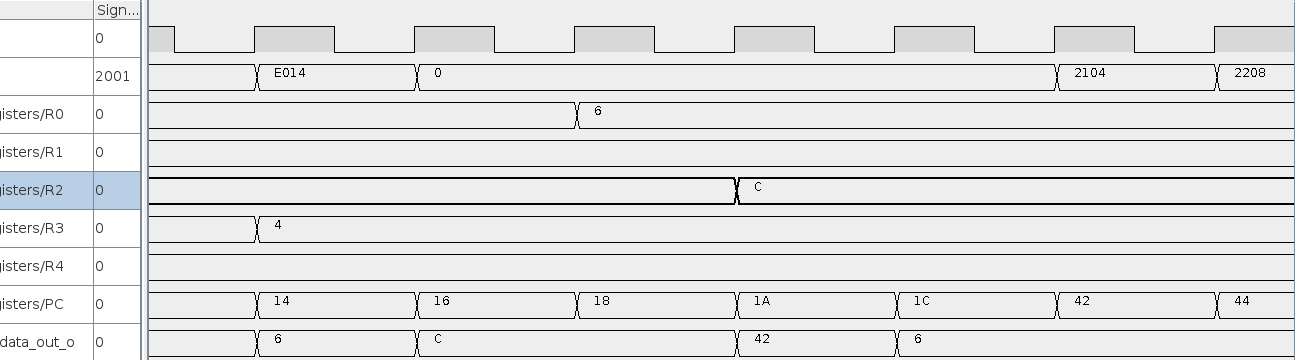
# Partie 1





Circuit faux



F

D

E

M

W

Inst = E014

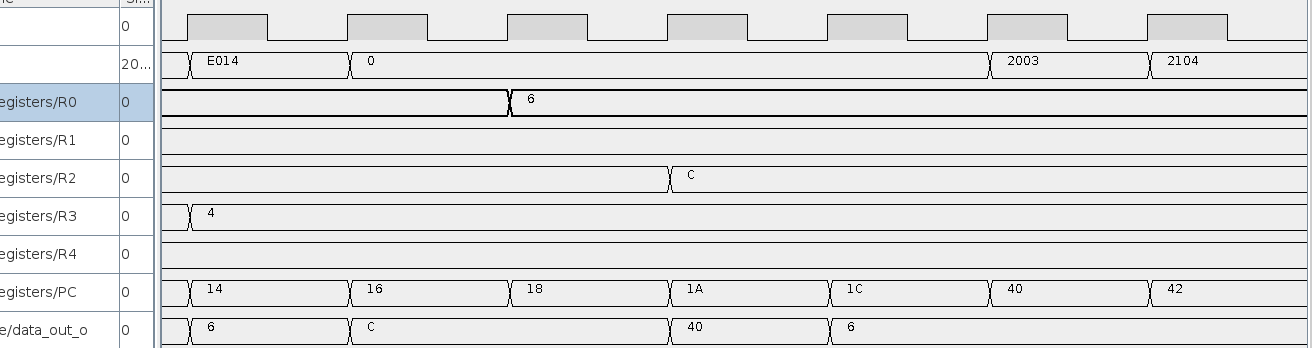
**Formule de Saut:**

**Formule de saut utilisée labo sans pipeline:**

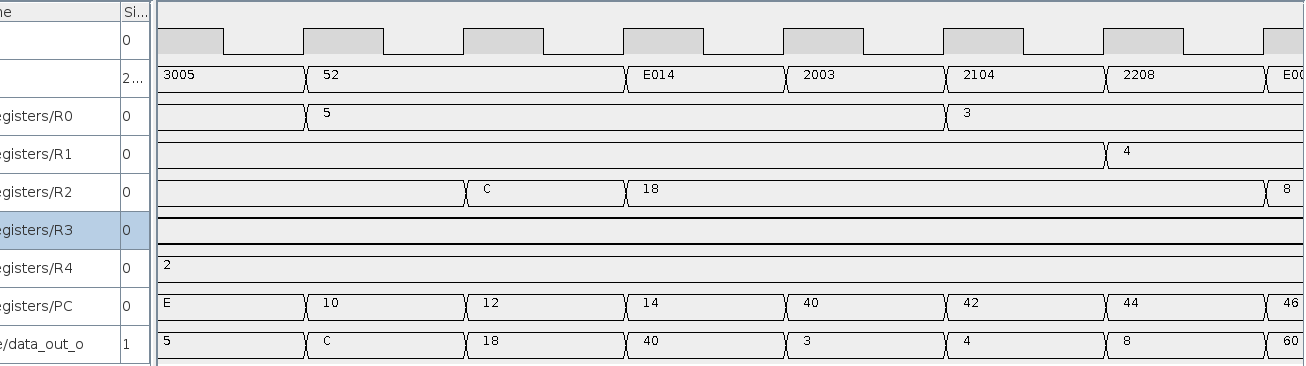
**Formule de saut utilisée labo pipeline:**

*E014 = 1110 0000 0001 0100*

Circuit juste



Circuit non pipeliné



## Q1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Étapes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Inst1 | F | D | E | M | W |  |  |
| Instr2 |  | F | D | E | M | W |  |
| Instr3 |  |  | F | D | E | M | W |

Quand nous serons à l’étape 3 c’est là qu’on calcule l’adresse de saut pour la 1ère instruction.

Cependant dans le décode on aura déjà le PC + 2 et dans le Fetch PC + 4. Pour faire les calculs d’execute on utilise les valeurs qui sont dans le bloc execute donc ce sera PC +2. Afin que la formule corresponde bien à la formule de saut () on fait donc un ajustement de +1 à l’offset. Ce qui fait donc si o n distribue le 2 dans la parenthèse on obtient bien la formule de saut.

Sans pipeline on a , on a donc un + 2 car le PC cette fois-ci n’est pas incrémenté car sans pipeline on ne commence pas une autre instruction tant que la première n’est pas terminée.

## Q2

Le LR doit être mis à jour uniquement si le saut est validé. Pour valider ce saut il faut attendre que l’instruction précédente termine la partie execute. Entre temps il faut stocker la valeur dans le bloc decode.

## Q3

La sélection des registres ce fait dans le bloc décode alors que l’écriture ne peut être faite qu’après l’execute

## Q4

Les signaux de contrôle de bus ont un MUX en entrée car quoi qu’il arrive une valeur doit être envoyée au prochain bloc pour qu’il sache quoi faire. Les autres signaux n’ont pas forcément besoin d’être envoyé ont peut donc désactiver le registre si on ne souhaite pas écrire.

## Q5

On rajouterait des étapes. Il faudrait donc que même les instructions n’utilisant pas la multiplication soit mise en attente et passe par ces blocs sans rien faire.

# 1.2

## Une image contenant table Description générée automatiquement

## Une image contenant table Description générée automatiquement

## Q1

J’ai vérifié pour le add et pour le and et les valeurs écrites dans les registres semblent correctes.

## Q2

9 cycles

# 1.3 Assembleur : dépendances de données

## Q1

Sans NOP

Une image contenant table

Description générée automatiquement

Avec NOP

Une image contenant texte, reçu, capture d’écran

Description générée automatiquement

Une image contenant table

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

## Q2

3 cycles sont nécessaires.

## Q3

Throughput 4000 inst/s

Latence = (1/f)\* nb clock entre 1er et dernier instruction = 7 ms

# 1.4

## Q1

2 cycles

## Q2

Throughput 4000 inst/s

Latence = (1/f)\* nb clock entre 1er et dernier instruction =

# 2.1

## Q1

2 cycles

## Q2

L’aléa de contrôle c’est en cas de saut conditionnel il faut vérifier si la condition est vérifiée ou non. Afin de savoir s’il faut sauter ou pas. Pour cela il faut faire un temps d’arrêt.

## Q3

C’est quand il y a un saut conditionnel

# Partie 2

# 1.1

## Q1

On fait passer l’adresse du registre de destination dans 3 registres qui seront synchronisés avec l’avancement de la commande dans les blocs.

À chaque sortie de registre on compare l’adresse du registre d avec le registre m et me. À chaque coup de clock on vérifie donc les valeurs des 3 registres de destination des commandes qui sont dans execute, memory et write back.

## Q2

Oui car on ne peut décoder l’instruction que quand les bonnes valeurs sont écrites dans les registres autrement on utilisera des valeurs fausse.

## Q3

On stocke l’adresse du registre de destination et on stocke si on va écrire ou non dans ce registre.

## Q4

On regarde les registres dans lequel on va écrire une valeur et on le compare avec les commandes suivantes les registres dans lesquels on va lire une valeur.

# 1.2

## Q1

Sel operand 1 pour le registre N

Sel operand2 pour le registre m

Et str\_data pour registre MEM.

## Hazard\_detection

## Q1

Decode\_en, execute\_en, mem\_access\_en et reg\_bank\_en

## Q2

Si un aléa est trouvé alors no\_data\_hazard change d’état et il aura une influence sur le control hazard

# 1.3

## Q1

## Q2

L’instruction BL génère deux instructions et il fait un saut mais dans le LR il stocke la valeur de l’instruction suivante.

Il génère un aléa de contrôle car il y a une décision de branchement. Et aléa de donnée à cause des deux instructions générées qui doivent être traitées toutes les deux dans l’execute.