

Exe 1 : L'objectif est de comparer 2 réalisations de Π_{ips} . La 1^{ère} est la réalisation classique sur pipeline ou 5 étages (appelée Π_{ips}).

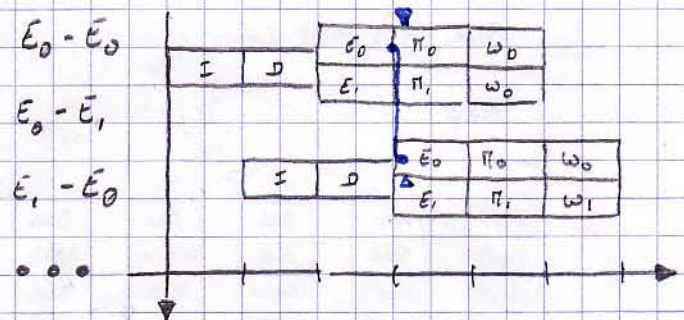
La seconde, appelée **SS2**, est une réalisation superscalaire à 2 pipelines.

Il s'agit d'une réalisation sur 5 étages : IF, DEC, EXE, MET, WBK.

G1/ Quels sont les bypass nécessaires à l'exécution des instructions dans SS2 ?

Gm a 32 bypass au total.

cf Méthode du cours.



G2/ Gm considère l'étage DEC de la réalisation SS2. Proposez deux situations où la seconde instruction du buffer d'instruction ne peut pas être lancée (Stall) ?

Gm a un cas de Stall si :

(A) s'il y a une dépendance de donnée entre les 2 instructions.

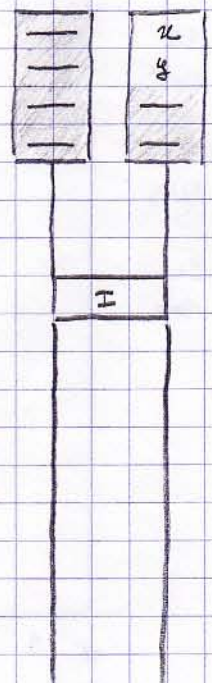
(B) s'il y a un branchement :

* / si la 2^{ème} instruction est un branchement et son delayed slot n'est pas dans le buffer.

** / si la 2^{ème} instruction est un branchement et dépend d'une instruction antérieure ($i-2, i-3, \dots$).

(C) si le second pipeline est bloqué.

(D) si la première instruction est bloquée.



Règle 7 : Dans l'étage DEC, la 2nd instruction ne peut être lancée que si elle est indépendante de la 1^{ère} instruction.

Règle 8 : Une instruction de branchement ne peut être lancée que si son delayed slot est dans le buffer.

Règle 10 : Une instruction est bloquée dans son pipeline si l'instruction qui est dans l'étage suivant est bloquée.

Règle 11 : Une instruction est bloquée dans son pipeline si elle n'a pas les ressources nécessaires à son exécution.

Q3/ Analysez l'exécution de cette boucle dans SS2 en supposant que l'adresse du début de la boucle est alignée.

Loop:

Lw R9, 0(R4)

Sll R9, R9, 1

Sw R9, 0(R4)

Addiu R4, R4, 4

Bne R4, R10, Loop

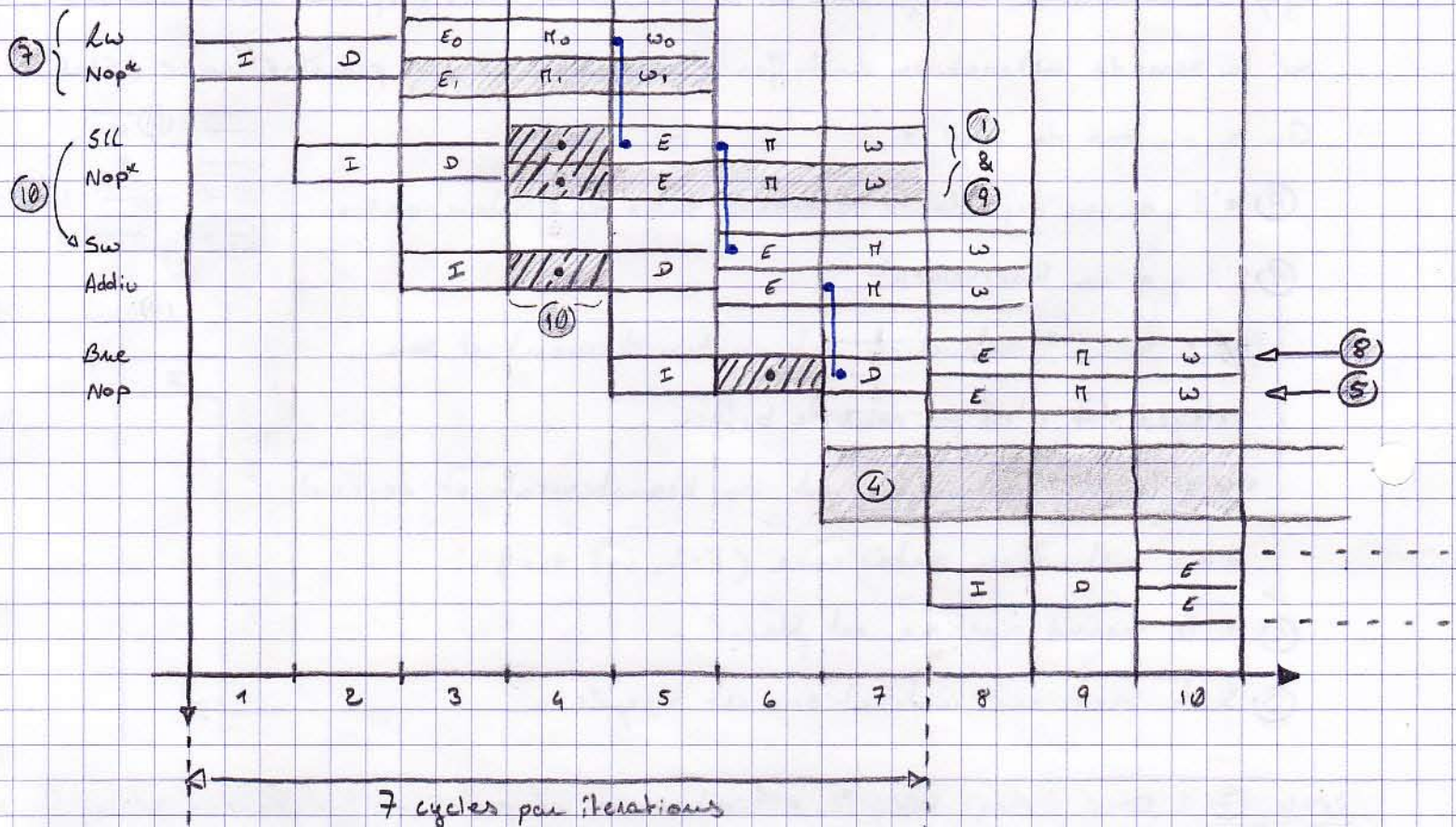
Nop

Règle (4) : le contenu du buffer est invalidé lorsque l'on exécute un branchement pris (réussit). De manière générale, ce contenu est invalidé lorsque l'exécution du branchement contredit la prédiction.

Règle (5) : Il y a un delayed slot après un branchement. Donc lorsque l'on exécute un branchement réussi, on n'invalidé pas son delayed slot.

Rule (4)

Nop*	Lw	Sll	Sw	Sll	Bne	Bne	Nop*	Lw		
Nop*	Sll	Sw	Addiu	Addiu	Nop	Nop	Nop*	Sll		
Nop*	Nop*	Addiu	Bne	Bne	N	N	Nop*	Nop*		
Nop*	Nop*	Nop*	Nop	Nop	4	4	Nop*	Nop*		



Règle (1) : Les instructions sont exécutées dans l'ordre du programme : une instruction ne peut pas exécuter un étage quelconque avant une instruction qui la précède.

Règle (9) : Une instruction avance toujours dans le pipeline dans lequel elle a été lancée. Elle ne peut pas changer de pipeline.

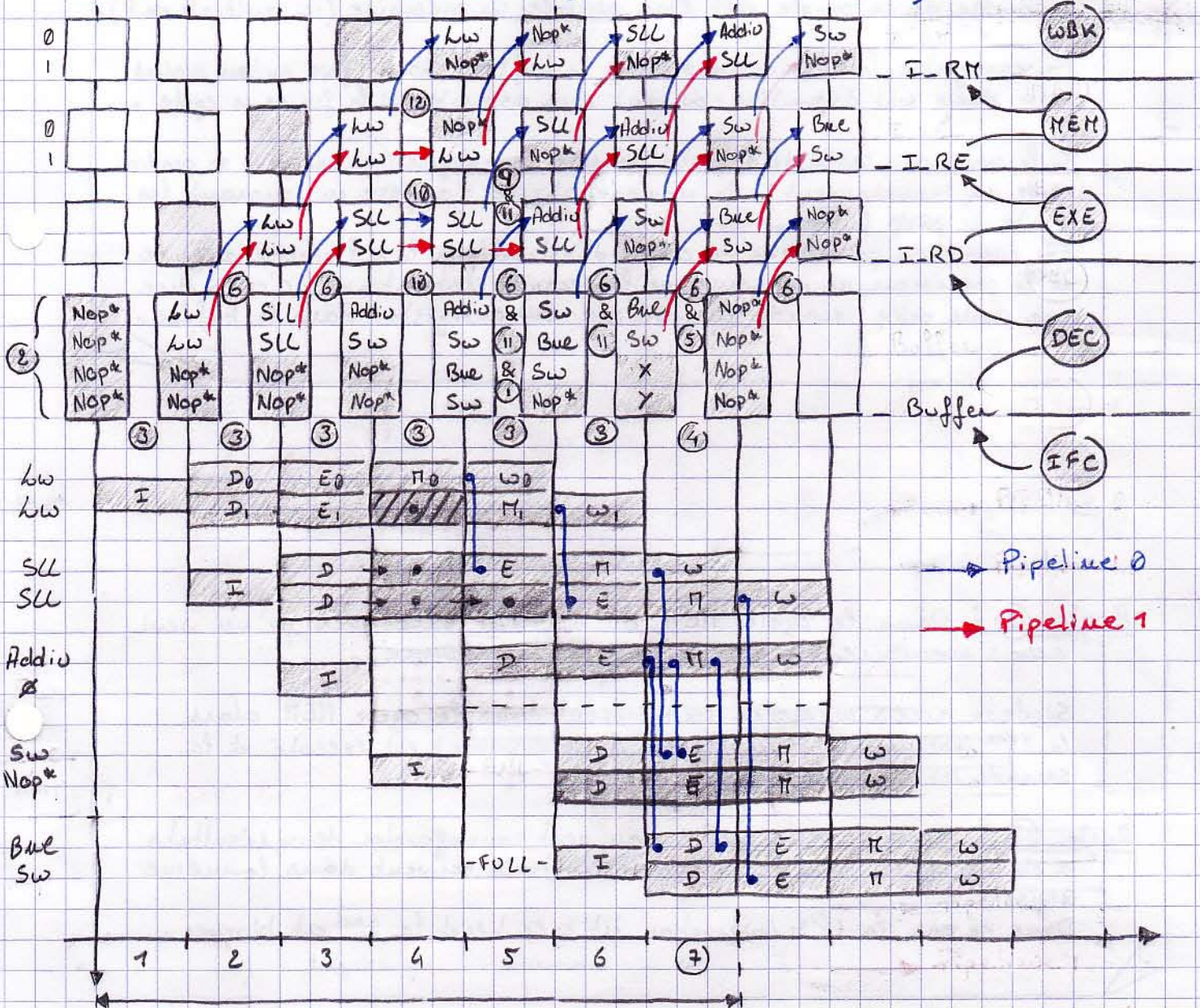
Exe 1 : Suite

Gg/ On déroule la boucle de la question précédente et on l'optimise légèrement ; analysez l'exécution de cette boucle dans SSE en supposant que l'adresse du début de la boucle est alignée.

Loop:

```

lw R9, 0(R4)
lw R19, 4(R4)
sll R9, R9, 1
sll R19, R19, 1
addiu R4, R4, 4
sw R9, -4(R4)
bne R4, R10, loop
sw R19, 0(R4)
    
```



Règle (6) : A chaque cycle, dans le cycle DEC, les deux 1^{ère} instructions du buffer sont décodées. Si les conditions d'exécution de ces instructions sont réunies alors, elles sont autorisées à passer au cycle EXE. On dit qu'elles sont lancées. Pour lancer une instruction, on autorise son écriture dans les registres I-RD : I-RD0 pour le pipeline 0 et I-RD1 pour le pipeline 1.

Règle (2) : Un superscalaire à deux pipeline possède un buffer d'instructions de 4 places.
Les places vides du buffer d'instructions sont remplies avec des Nop*.

⚠ Bien faire la différence avec l'instruction Nop appartenant au programme.

Règle (3) : Dans le cycle IFI, les instructions sont extraites de la mémoire deux par deux pour un superscalaire à deux pipelines.

L'adresse de ce couple doit être alignée en mémoire (ici multiple de 8).

On enregistre les instructions dans le buffer chaque fois qu'au moins deux places sont disponibles - ou vont être disponibles à la fin de ce cycle.

Si le processeur tente de lire une adresse non-alignée (ce cas peut se produire après un branchement), la mémoire aligne l'adresse en ignorant les 3 bits de poids faibles.

Elle envoie au processeur un couple d'instructions lu à l'adresse alignée et le processeur ne conserve que la seconde instruction - c'est à dire que seule cette seconde instruction est enregistrée dans le buffer.

Règle (2) : Dans le cycle MEM, on ne peut effectuer qu'un seul accès mémoire au maximum (load ou Store).

Si deux accès mémoire sont présent dans le cycle MEM alors, la 1^{ère} instruction (dans l'ordre du programme) est exécutée et la seconde est bloquée (stall).

Règle (3) : Dans le cycle WBK, on peut sauvegarder deux résultats à chaque cycles, sauf si les deux instructions écrivent dans le même registre.

Dans ce cas, la 1^{ère} instruction est exécutée et la 2nd est bloquée.

Exe 2 : La fonction `gethight` calcule le nombre d'elements dans une liste.
Un element de cette liste est une structure de donnée nommée `chain`.
Une structure `chain` est composée de 2 pointeurs : le premier pointe vers l'adresse de la structure `chain` suivante et l'autre vers l'adresse de la donnée.

```
int gethight( struct chain *pt ) {
    int i = 0;
    while( pt != NULL ) {
        i++;
        pt = pt->NEXT;
    }
    return i;
}
```

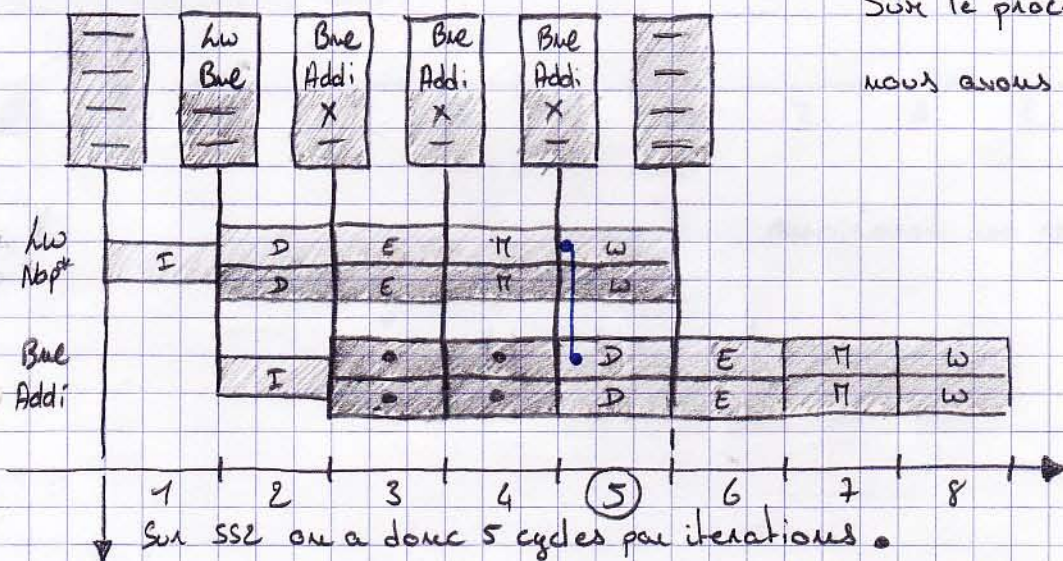
```
struct chain {
    struct chain *NEXT;
    void *data;
};
```

La boucle principale de cette fonction s'écrit en assembleur (on suppose que R8 contient l'adresse de la chaîne `pt`) :

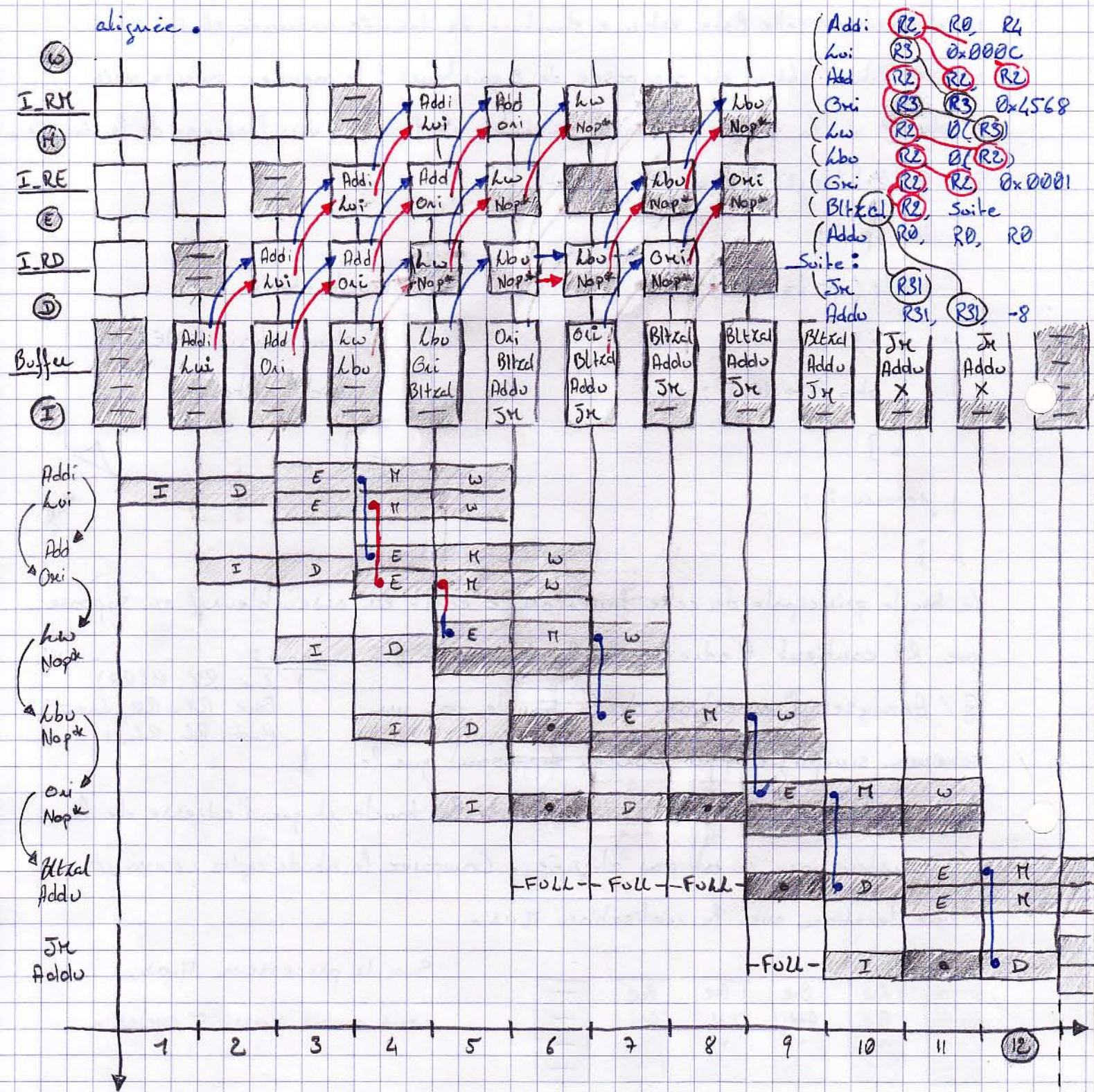
S/ Analysez l'exécution de la boucle par un schéma simplifié dans SS2 en supposant que le buffer d'instruction est vide au début de la boucle et que l'adresse de la 1^{ère} instruction n'est pas alignée. Comparez le nb. de cycles nécessaires à une itération avec la réalisation MIPS.

```
Loop:
    lw R8, 0(R8)
    bne R8, R0, Loop
    addi R2, R2, 1
```

Sur le processeur MIPS, nous avons aussi 5 cycles.



Exe 3 : Analyser, à l'aide d'un schéma simplifié, l'exécution du code suivant pour SS2. On suppose que le buffer d'instructions est vide au début de cette séquence et que la 1^{ère} instruction est placée à une adresse alignée.



⇒ #C = 12 cycles par itérations