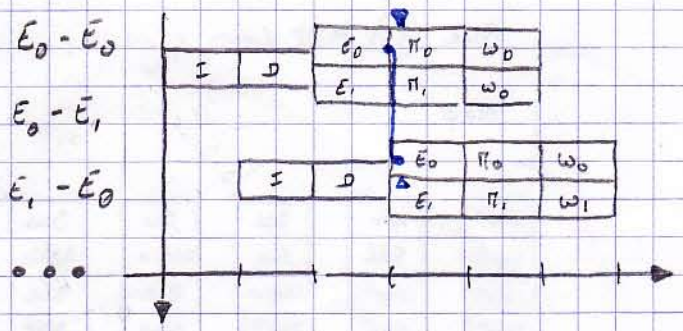
Exo 1 : L'objectif est de comparer 2 réalisations de Mips. La 1ère est la réalisation classique sur pipeline en 5 étages (appelée Mips). La seconde, appelée SS2 est une réalisation superscalaire à 2 pipelines. Il s'agit d'une réalisation sur 5 étages : IFC, DEC, EXE, MEM, WBK.

Q1/ Quels sont les bypass nécessaires à l'execution des instructions dans SS2 ?

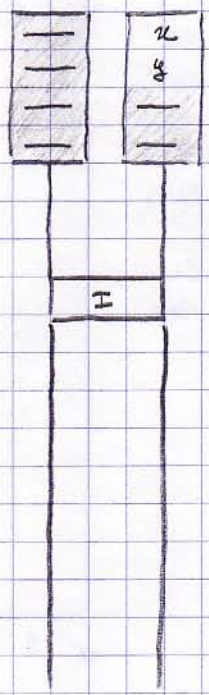
On a 32 bypass au total.

cf Methode du cours.



Q2/ On 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) ?

On a un cas de Stall si :



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

(B) s'il y a un branchement :

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

**/ si la 2ème instruction est un branchement et dépend d'une instruction antérieure (i-2, i-3, ...).

(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 independante de la 1ère 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.

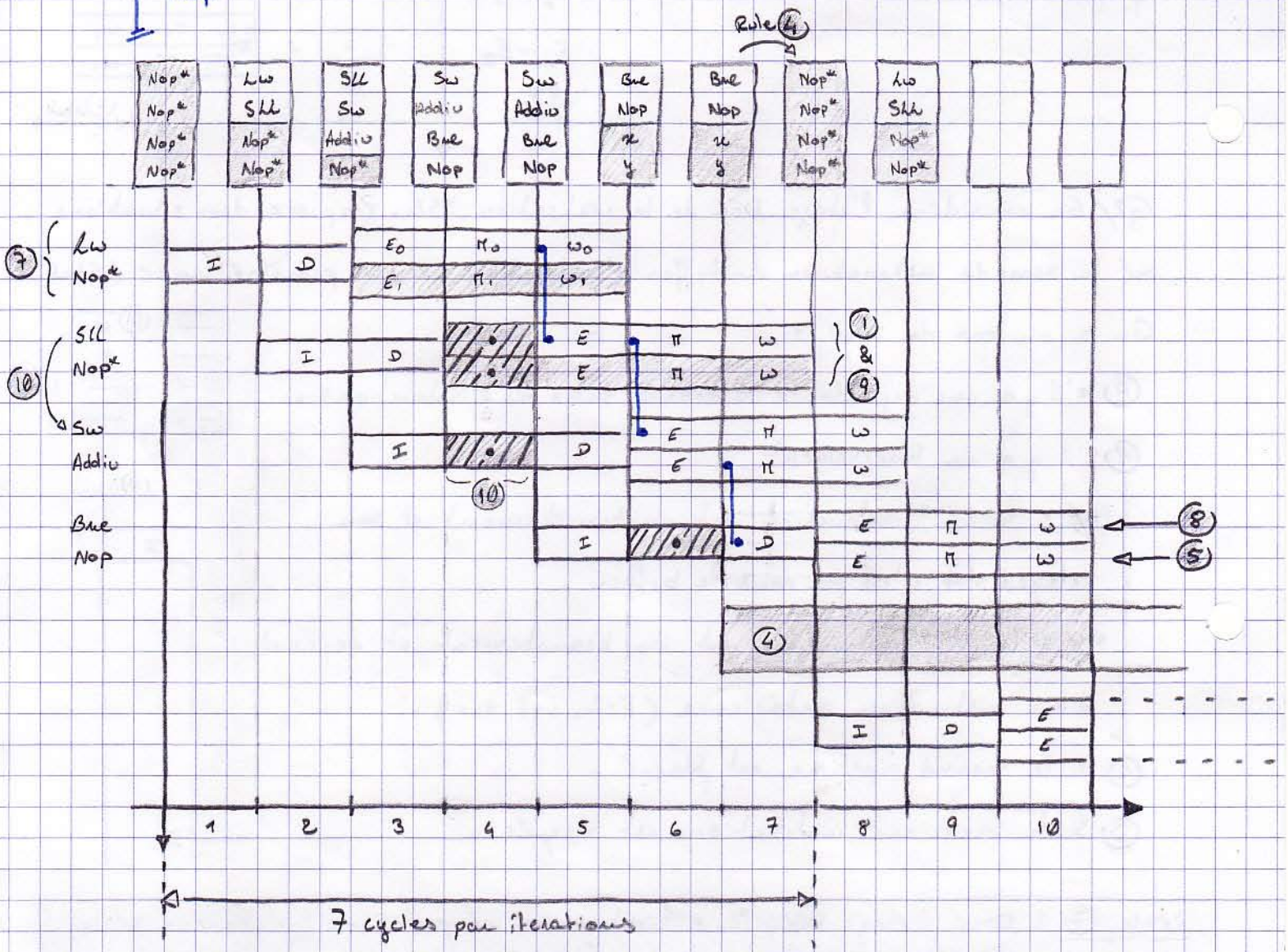
Règle (11) : Une instruction est bloquée dans son pipeline si elle n'a pas les ressources nécessaire à son execution.

Q3/ Analysez l'execution 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 contenus du buffer est invalidé lorsque l'on execute un branchement pris (réussit). De manière générale, ce contenus est invalidé lorsque l'execution du branchement contredit la prédiction.

Règle ⑤ : Il y a un delayed slot après un branchement. Donc lorsque l'on execute un branchement réussit, on n'invalide pas son delayed slot.



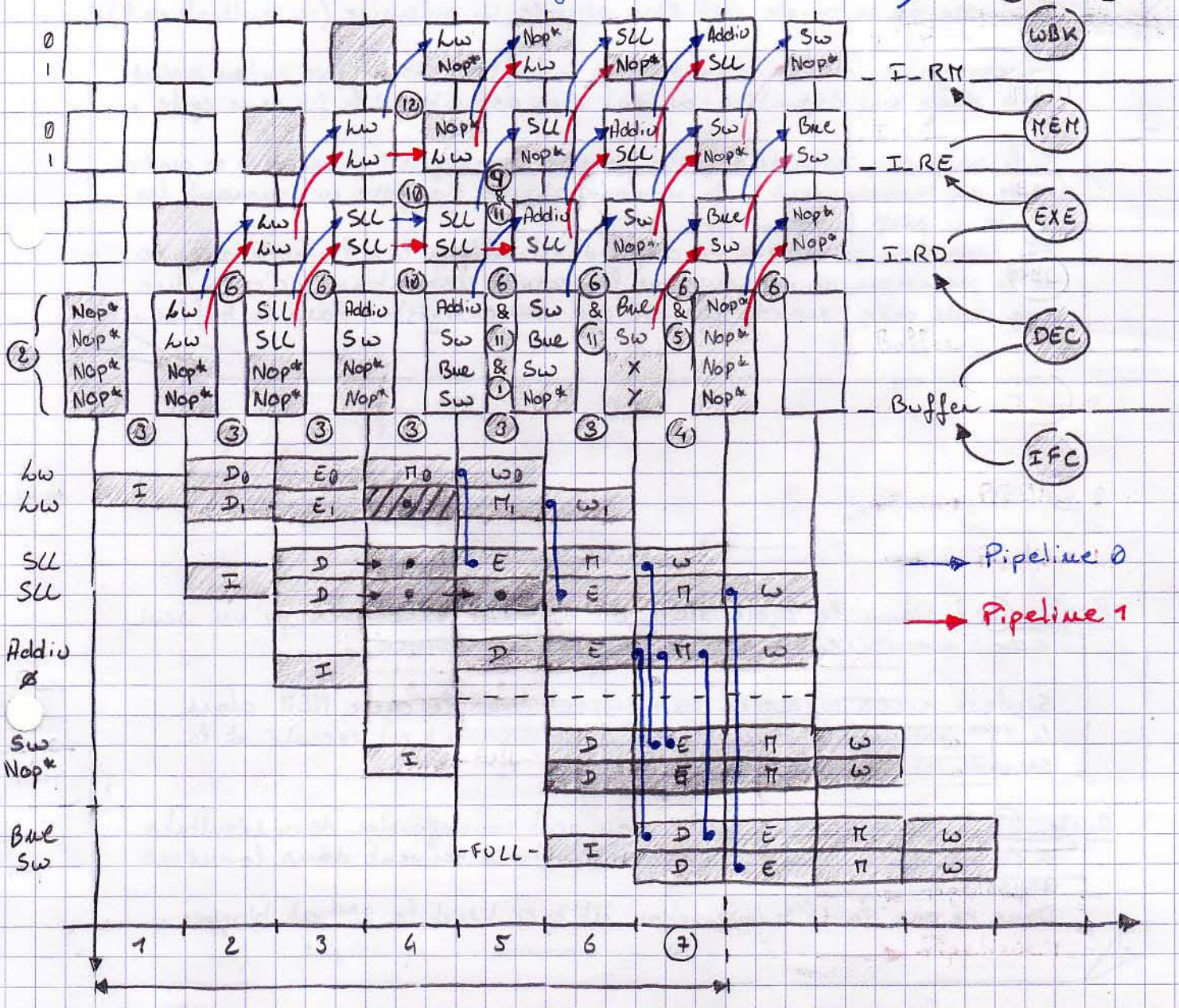
Règle ① : Les instruction sont exécutés dans l'ordre du programme : une instruction ne peut pas executer un étage quelconque avant une instruction qui la précède.

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

Exe 1 : Suite

Q4/ On déroule la boucle de la question precedante et on l'optimise légèrement ; analysez l'execution de cette boucle dans sse en supposant que l'adresse du debut 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 1ères instructions du buffer sont décodées.
Si les conditions d'execution de ces instructions sont réunis 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 différences avec l'instruction Nop appartenant au programme.

Règle (3) : Dans le cycle IFC, 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 disponible à la fin de ce cycle.

Si le processeur tente de lire une adresse non-aligné (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 enregistré dans le buffer.

Règle (12) : 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 1ère instruction (dans l'ordre du programme) est executé et la seconde est bloquée (stall).

Règle (13) : 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 1ère instruction est exécutée et la 2nd est bloquée.

Exe 2 : La fonction GetListLenght 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.

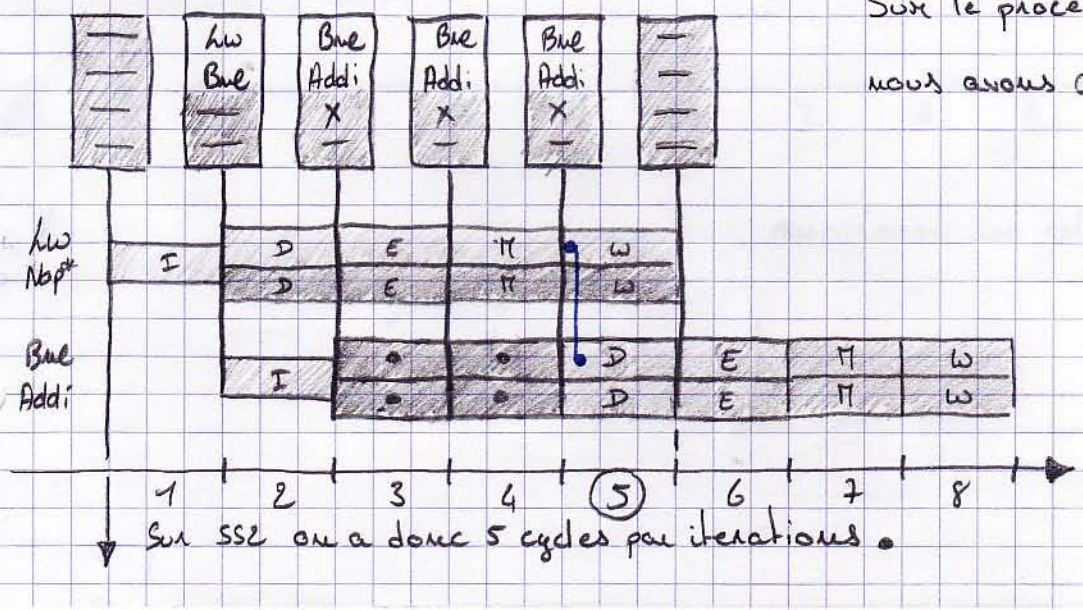
```
int GetListLenght ( struct chain * pt ) {
    int i = 0 ;
    while ( pt != NULL ) {
        i++ ;
        pt = pt -> NEXT ;
        }
    return i ;
    }
```

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

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

```
Loop :
    lw  R8, 0(R8)
    Bne R8, R0, Loop
    Addi R2, R2, 1
```

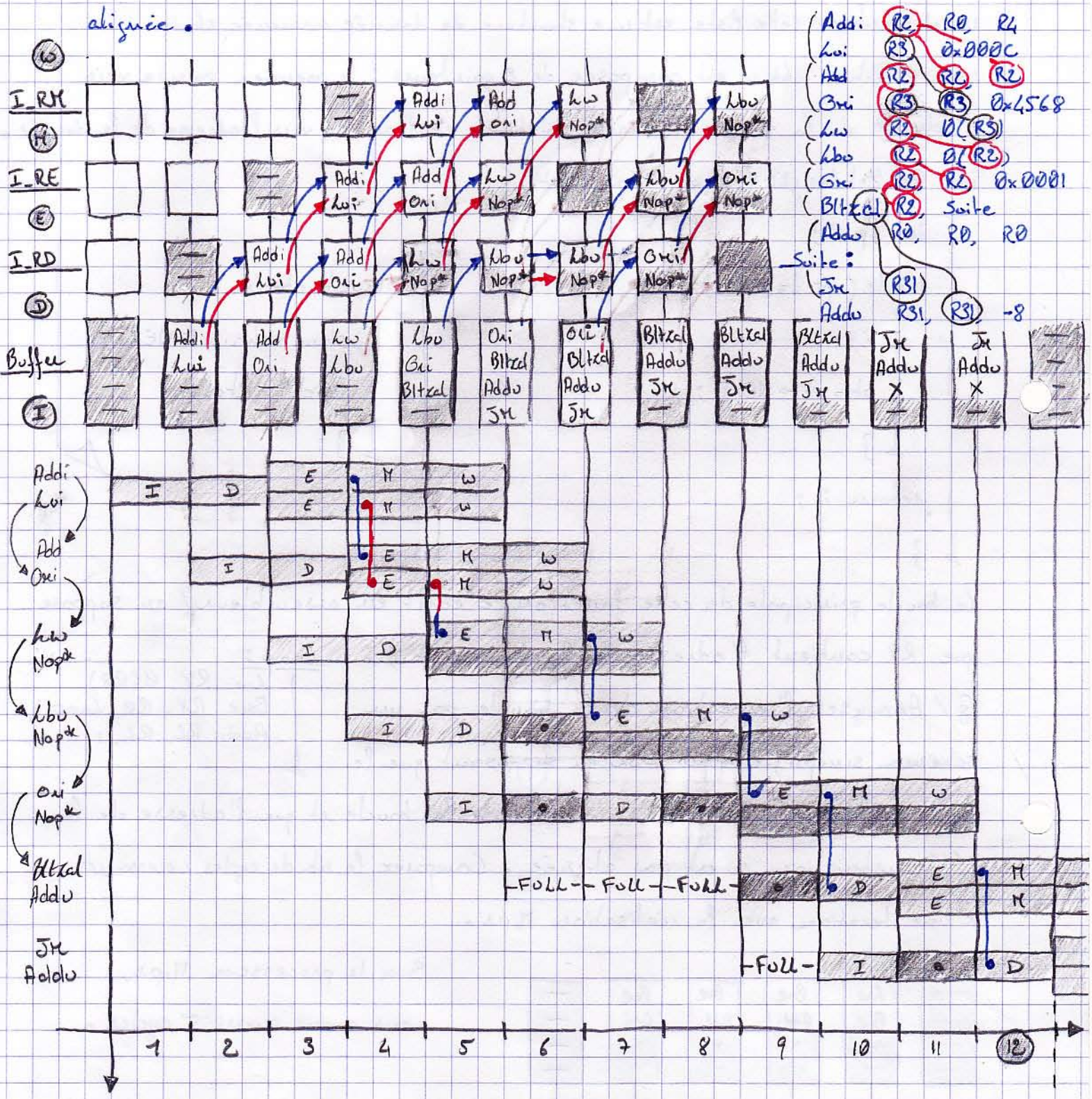
Q/ Analysez l'execution de la boucle par un schéma simplifié dans SS2 en supposant que le buffer d'instruction est vide au debut de la boucle et que l'adresse de la 1ère instruction n'est pas alignée. Comparez le nb. de cycles necessaires à une iteration avec la réalisation Mips.



Sur le processeur Mips, nous avons aussi 5 cycles.

Sur SS2 on a donc 5 cycles par iterations.

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



=> #C = 12 cycles par iterations