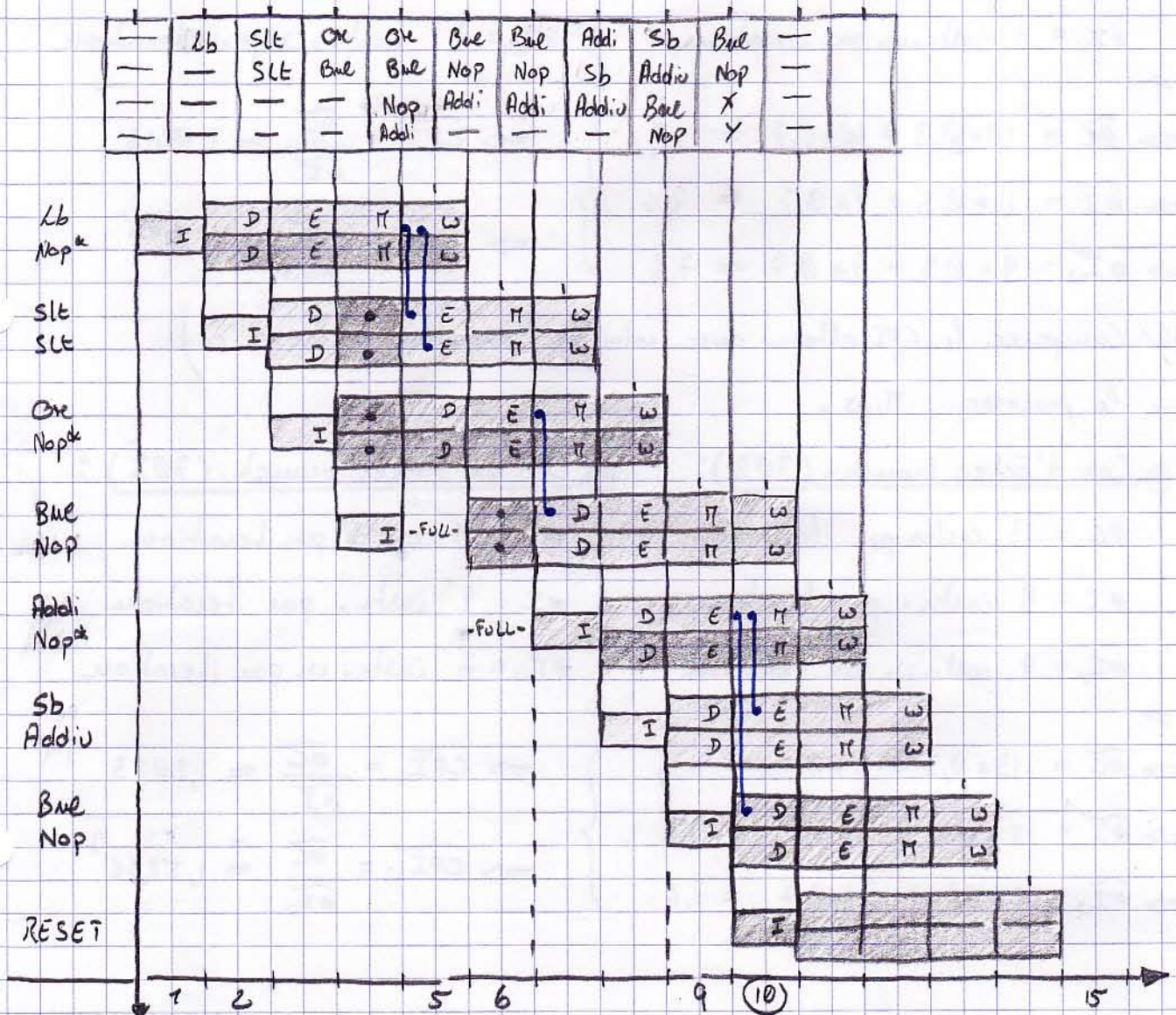
Exe 4 : La fonction strupper transforme une chaine de caractères en majuscules ; la compilation de cette fonction pour un processeur non-pipeliné a produit le code suivant :

```
strupper :
    Addu R2, R0, R4
    Addi R11, R0, 'a'
    Addi R12, R0, 'z'
Loop :
    Lb   R8, 0(R4)
    Slt  R9, R8, R11
    Slt  R10, R12, R8
    Or   R10, R10, R9
    Bne  R10, R0, Endif
    Addi R8, R8, 'A'-'a'
    Sb   R8, 0(R4)
Endif :
    Addiu R4, R4, 1
    Bne  R8, R0, Loop
    Jr R31
```

Q1/ Transformer ce code de telle façon qu'il puisse être exécuté sur le processeur SSE.

Dans le corps de la boucle Loop, il suffit de rajouter un delayed slot après chaque branchement.

Q2/ Analyser l'exécution de la boucle sur le processeur SSE à l'aide d'un schéma simplifié. On suppose qu'au début de l'exécution de la boucle, le buffer d'instruction est vide et que l'étiquette Loop est à une adresse non-alignée.



§3/ Calculez le nombre de cycles necessaire à l'execution d'une iteration de la boucle dans le cas où le if (du code src) réussit.

Réussite du if du code source <=> Echec branchement vers Endif.

Nous avons donc 10 cycles par iterations en cas de reussite du "if".

§4/ Calculez le nombre de cycles necessaire à l'execution d'une iteration de la boucle dans le cas où le if (du code src) échoue.

Echec du if du code source <=> Réussite branchement vers Endif.

Nous avons également 10 cycles par iterations en cas d'echec du "if".

§5/ Sachant que 30% des caracteres sont minuscules, calculez le CPI et CPI utile de la boucle.

D'après les questions precedantes, on a :

- Cas d'Echec branch. (30%) :
  - #C = 10 cycles par iteration
  - #I = 11 instr. par iteration
  - $\#I_u$ = 9 instr.u. par iteration
- Cas de Succes branch. (70%) :
  - #C = 10 cycles par iteration
  - #I = 9 instr. par iteration
  - $\#I_u$ = 7 instr.u. par iteration

$\Rightarrow \widetilde{\#C} = 10*0,3 + 10*0,7 = 10$

$\Rightarrow \widetilde{\#I} = 11*0,3 + 9*0,7 = 9,6$

$\Rightarrow \widetilde{\#I_u} = 9*0,3 + 7*0,7 \approx 7,6$

$\Rightarrow \widetilde{CPI} = \frac{\widetilde{\#C}}{\widetilde{\#I}} \approx 1,0416$

$\Rightarrow \widetilde{CPI_u} = \frac{\widetilde{\#C}}{\widetilde{\#I_u}} \approx 1,3157$

§6/ Comparez le CPI obtenu avec celui de l'execution du même code sur le processeur Mips.

3,9 / 3,7

- Cas d'Echec branch. (30%) :
  - #C = 13 cycles par iteration
  - #I = 11 instr. par iteration
  - $\#I_u$ = 9 instr.u. par iteration
- Cas de Succes branch. (70%) :
  - #C = 11 cycles par iteration
  - #I = 9 instr. par iteration
  - $\#I_u$ = 7 instr.u. par iteration

$\Rightarrow \widetilde{\#C} = 13*0,3 + 11*0,7 = 11,6$

$\Rightarrow \widetilde{\#I} = 11*0,3 + 9*0,7 = 9,6$

$\Rightarrow \widetilde{\#I_u} = 9*0,3 + 7*0,7 = 7,6$

$\Rightarrow \widetilde{CPI} = \frac{\widetilde{\#C}}{\widetilde{\#I}} \approx 1,2083$

$\Rightarrow \widetilde{CPI_u} = \frac{\widetilde{\#C}}{\widetilde{\#I_u}} \approx 1,5236$

Exe 5 : On considère le programme suivant :

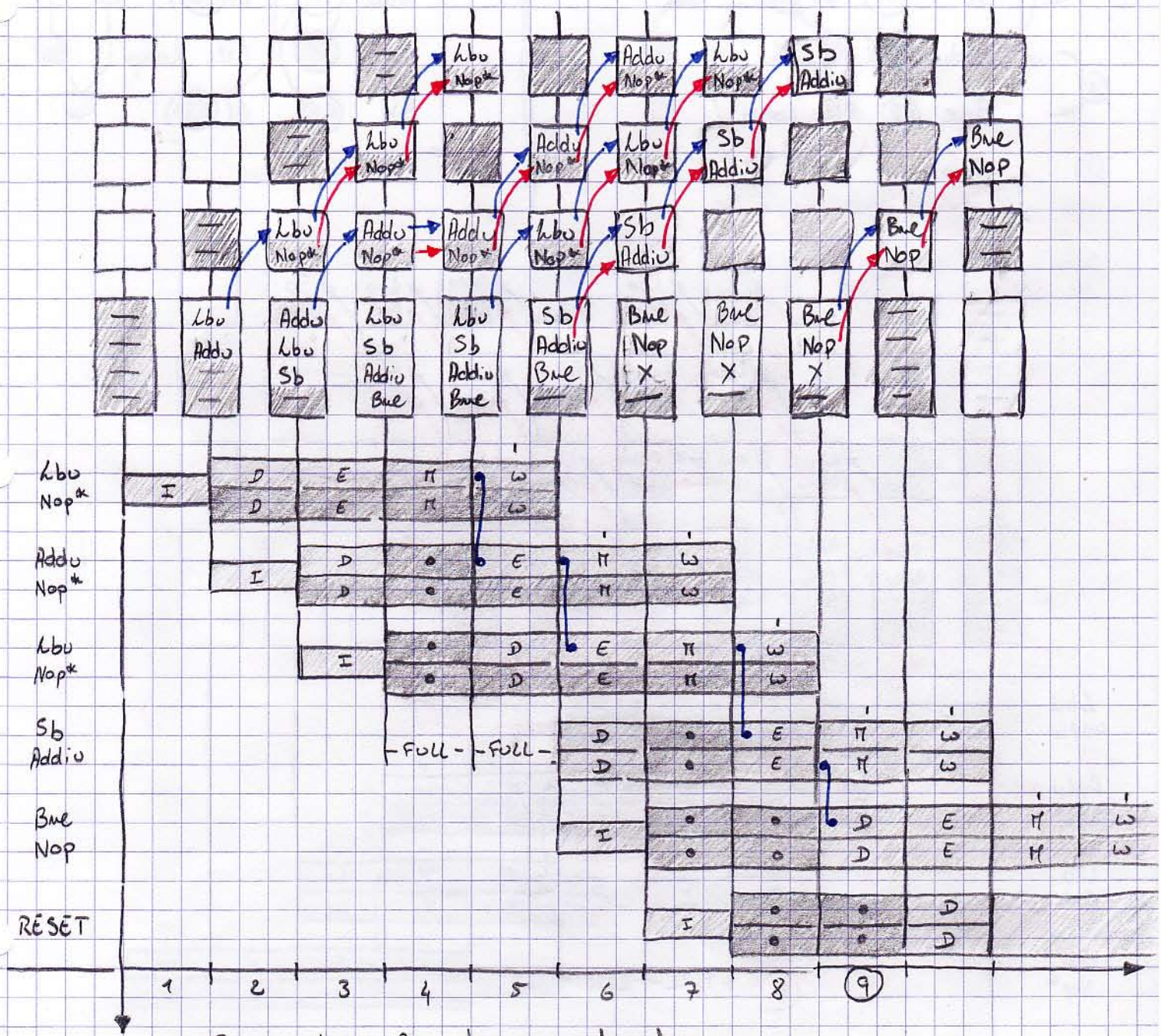
```
Loop: Lbu  R10, 0(R8)
      Addu R10, R10, R9
      Lbu  R11, 0(R10)
      Sb   R11, 0(R8)

      Addiu R8, R8, 1
      Bne  R8, R12, Loop
```

Q1/ Modifiez le programme pour qu'il soit executable sur le SS2.

Il suffit de rajouter le delayed slot du Bne.

Q2/ Analysez, à l'aide d'un schéma simplifié, l'execution dans le SS2. On suppose qu'au debut de l'execution de la boucle, le buffer d'instruction est vide et que l'étiquette Loop se trouve à une adresse alignée. Calculez le nombre de cycles pour une itération.

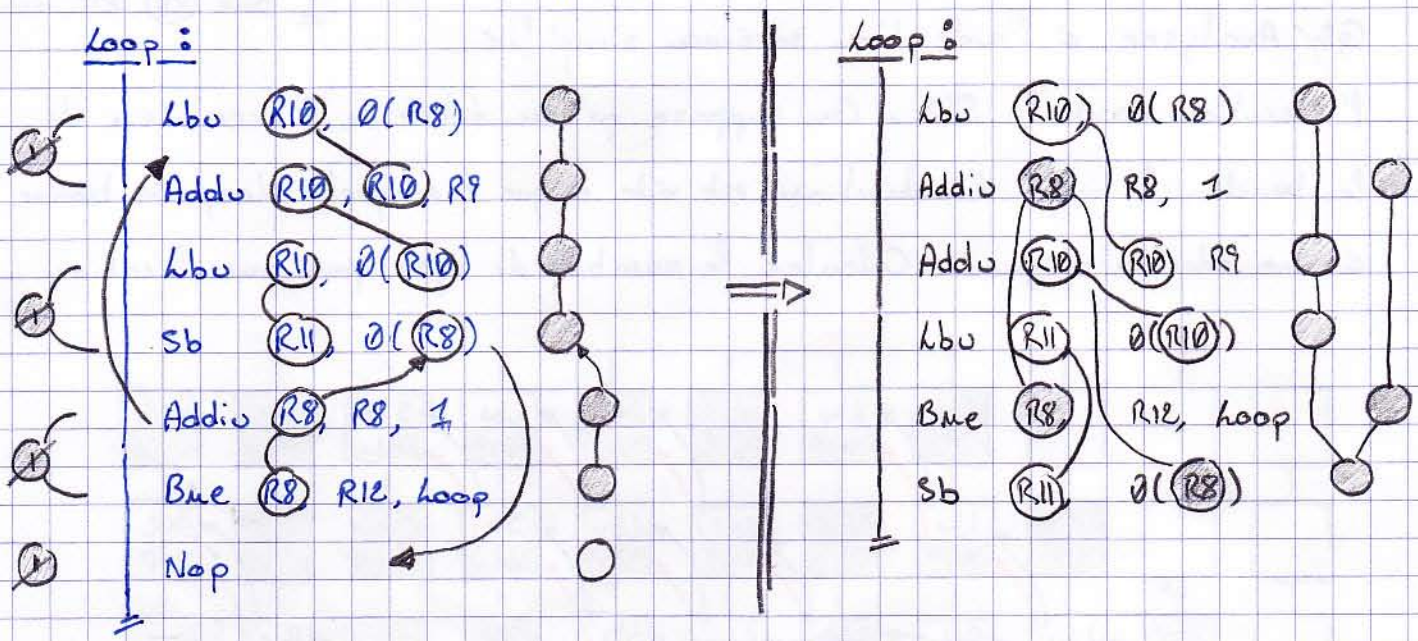


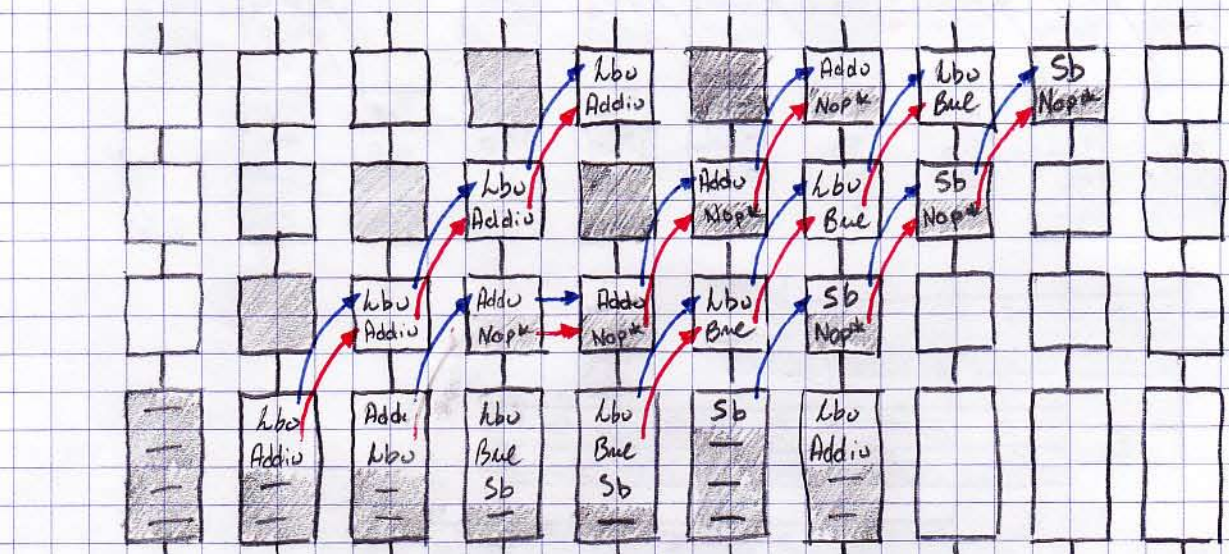
On a donc 9 cycles par iterations

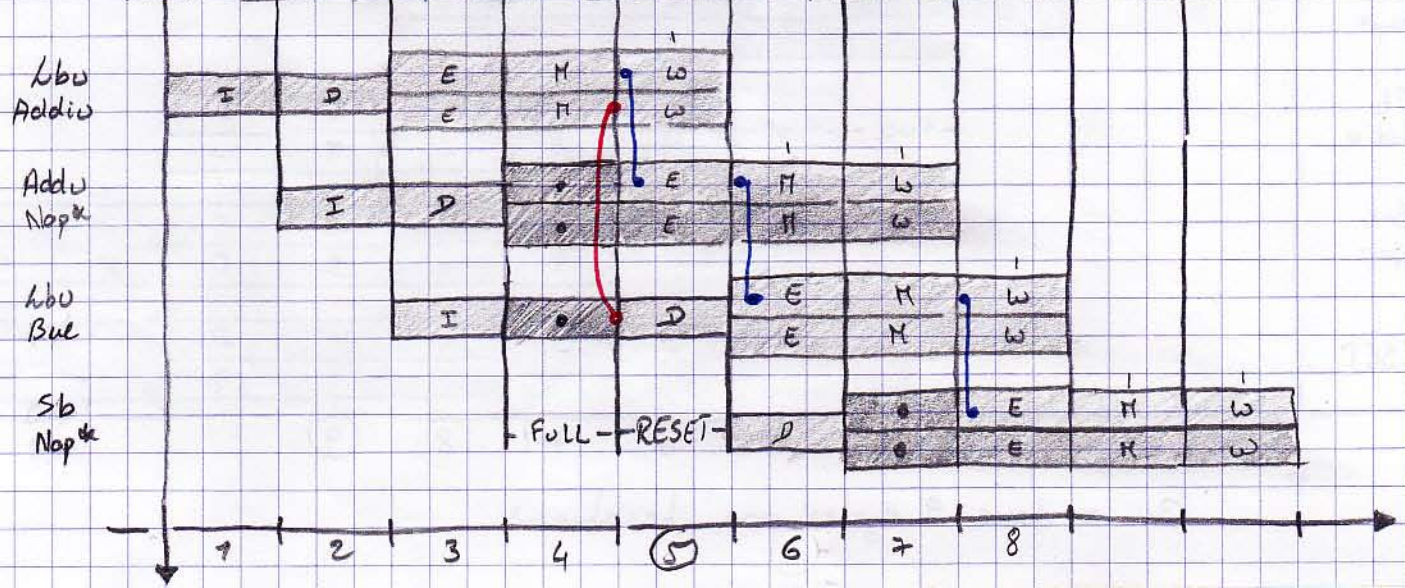
Q3/ Comparez le nombre de cycles nécessaires à une itération avec la réalisation Mips.

Sur Mips, nous avons 10 cycles par itération. Avec le SSE, nous avons donc un gain de 10%.

Q4/ Optimisez le code en changeant l'ordre des instructions.
Calculez le nombre de cycles par itération.





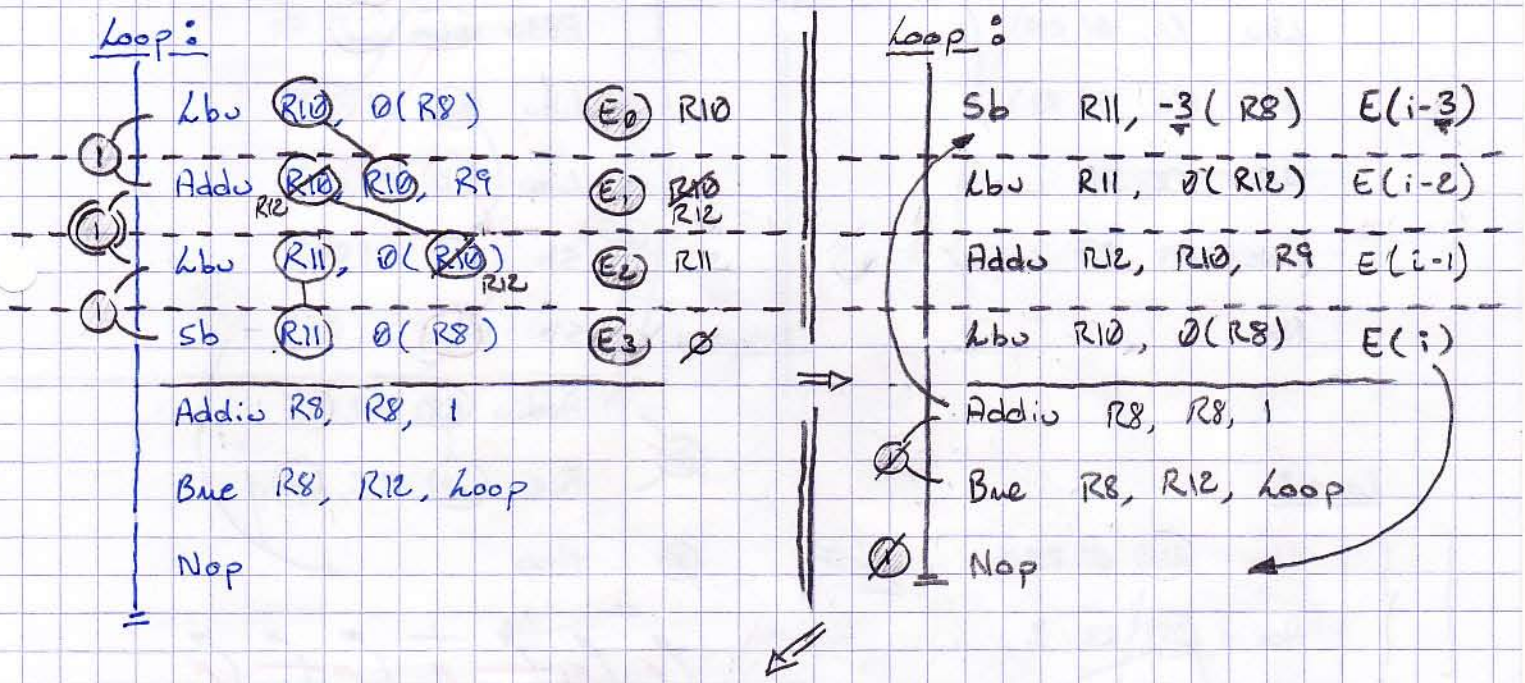


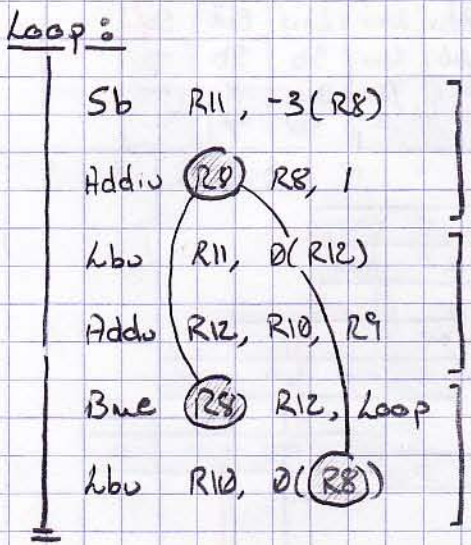
Exe 5 : Suite

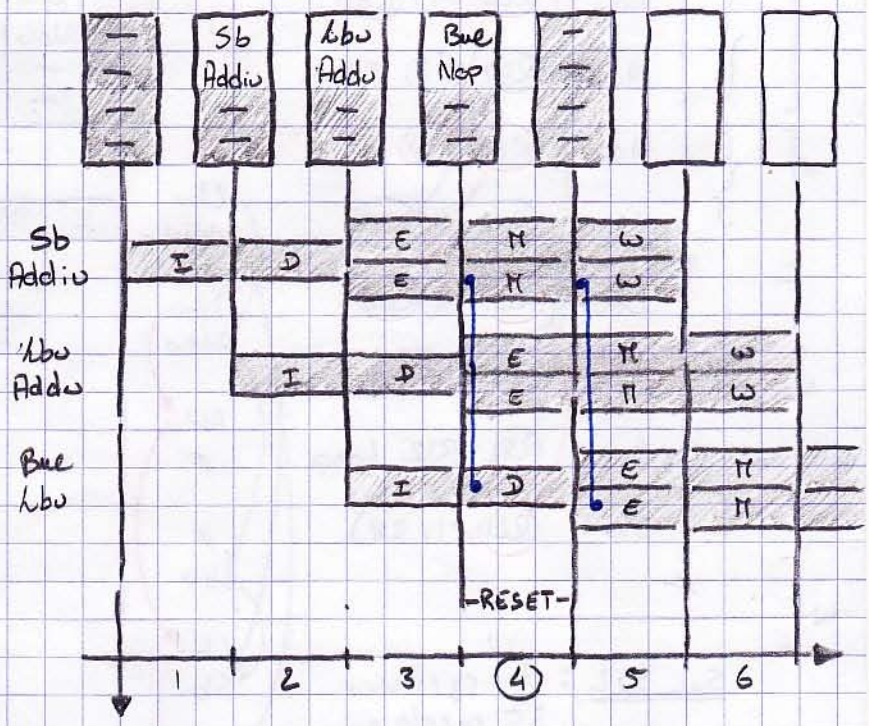
Q5/ Comparez le nombre de cycles nécessaires à une itération avec le Mips.

Sur une réalisation Mips, il nous faut 6 cycles également.

Q6&7/ Optimisez le code initial en utilisant le "pipeline logiciel". Calculez le nombre de cycles par itération.







Nous avons donc un total de 4 cycles par itération pour une exécution sur SS2 par le pipeline logiciel.

Avec pipeline logiciel, sur le Mips on a 6 cycles par itération.

(cf. Pipeline, Superpipeline et Optimisation : Exe 2)

Q8 & 9/ Déroulez le corps de la boucle 1 fois (traitement de deux éléments par itération) et optimisez. Calculez le nombre de cycles par itération et par élément pour SS2.

Comparez avec le Mips.

```
Loop:
    Lbu   R10, 0(R8)
    Addu  R10, R10, R9
    Lbu   R11, 0(R10)
    Sb    R11, 0(R8)
    Addiu R8, R8, 1
    Bne   R8, R12, loop
    Nop
```

T (Lbu, Addu, Lbu, Sb) ; 8 (Addiu, Bne, Nop)

```
Loop:
    Lbu   R10, 0(R8)
    Addiu R8, R8, 2
    Lbu   R20, -1(R8)
    Addu  R10, R10, R9
    Addu  R20, R20, R9
    Lbu   R11, 0(R10)
    Lbu   R21, 0(R20)
    Sb    R11, -2(R8)
    Bne   R8, R12, loop
    Sb    R21, -1(R8)
```

Sur SS2 : 7 cy/iter. 3,5 cy/elt.

Sur Mips : 10 cy/iter. 5 cy/elt.

=> Gain ≈ 30 %