HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Prof. Romuald
Mosqueron

Olivier Auberson

Laboratoire d'architecture des ordinateurs
semestre printemps 2020 - 2021

Quiz : Laboratoire PIPELINE

Nom / Prénom :
Groupe :

Total : **/ 20pts**

Informations générales

— Vous avez 40 minutes pour répondre aux questions.
— Vous avez le droit à vos notes sur le labo, votre PC privé, ainsi qu'à l'utilisation de Logisim pendant le quiz. Vous n'avez pas le droit d'utiliser Internet, ni aucun autre moyen de communication pendant le quiz.
— Le quiz est individuel.
— Vous devez rendre la donnée à la fin du quiz. Veuillez bien noter votre **nom** et **prénom** sur **toutes** les feuilles.
— Vous pouvez répondre aux questions directement sur la donnée.

Question 1 - Aléas de Contrôle

1. **(2pts)** Parmi ces 4 instructions, encerclez celle(s) qui génère(nt) un **aléa de contrôle** :

```
-> 1) B label1

2) LDRH r0, [r0, r2]

3) ADD r1, #0

4) STRB r2, [r6, #0]
```

Question 2 - Aléas de Donnée et Forwarding

1. **(3pts)** Parmi ces codes assembleur, encerclez le(s) code(s) qui génère(nt) des aléas de donnée. Les 3 codes sont indépendants les uns des autres.

```
1)  ADD r0, #4
    MOV r0, #3

-> 2)  MOV r1, #6
    BL label1

-> 3)  MOV r2, #5
    STRH r2, [r3, #0]
```

2. **(4pts)** Expliquez pourquoi l'instruction BL génère un aléa de donnée **et** un aléa de contrôle.

   L'instuction BL est séparée en deux instruction bl_msb et bl_lsb. bl_msb calcule une partie de l'adresse du saut et l'écrit dans le registre LR. L'instruction bl_lsb lit le registre LR. Il y a donc un aléa de donnée pour lire car il faut que bl_msb ait fini d'écrire la valeur de LR. Comme bl_lsb est un saut il y a aussi un aléa de contrôle pour résoudre l'adresse du saut.

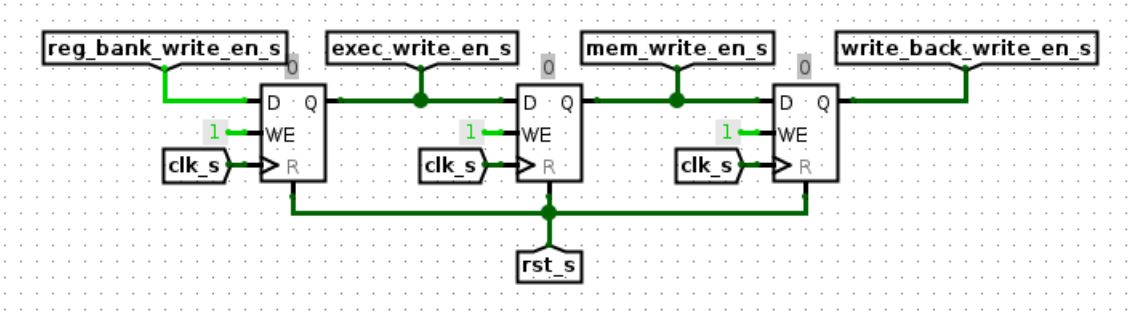

FIGURE 1 – Question 2.3 : Utilisation de reg_bank_write_en_s dans le bloc data_hazard

3. **(3pts)** Expliquez la Figure 1 du bloc data_hazard. En particulier, expliquez pourquoi le signal reg_bank_write_en_s passe à travers une série de registre et comment ces signaux sont utilisés dans la détection d'aléa de donnée.

   Le signal reg_bank_write_en_s permet de savoir si une instruction va faire une écriture dans un registre. C'est information permet de savoir si le registre adr_reg_d est valide. Ce qui nous intéresse est de savoir si l'instruction qui est dans le bloc EXECUTE, MEM_ACCESS ou WRITE_BACK va faire une écriture. Donc le signal doit être sauvegardé dans une série de registres. Si adr_reg_d d'une instruction dans un de ces blocs corresponds à adr_reg_n/m/mem d'une instruction dans le bloc DECODE mais que son reg_bank_write_en correspondant vaut 0, alors il n'y a pas d'aléa de donnée.

Question 3 - Chronogramme

Analysez le chronogramme du **processeur avec forwarding** qui vous a été fourni à la Figure 3 et répondez aux questions suivantes.Vous pouvez directement annoter le chronogramme si vous le désirez. N'oubliez pas de mettre votre nom et prénom sur la feuille.
Le code désassemblé du programme executé vous est fourni à la Figure 2.

1. **(2pts)** Quelle(s) instruction(s) n'est (ne sont) pas executée(s) correctement ?
   L'instruction 0x24 n'est pas executée correctement. On peut le voir sur le chronogramme car le résultat du calcul devrait être 0x4 + 0x5 = 0x9 mais le résultat est 0xC
2. **(4pts)** Expliquer d'où vient le problème.
   le problème est le forwarding du registre R0. Il devrait venir du bloc execute mais il vient du bloc MEM_ACCESS. On peut le voir car sel_op1_forward = 2 au lieu de 1.

```
Disassembly of section .text:

00000000 <.text>:
   0:        2205              movs        r2, #5
   2:        e00d              b.n        20 <SAUT1> ; saut inconditionnel
   4:        3004              adds        r0, #4
        ...

00000020 <SAUT1>:
  20:        1c90              adds        r0, r2, #2
  22:        2004              movs        r0, #4
  24:        1881              adds        r1, r0, r2
  26:        2202              movs        r2, #2
  28:        46c0              nop
  2a:        46c0              nop
```

FIGURE 2 – Question 3 : Code désassemblé

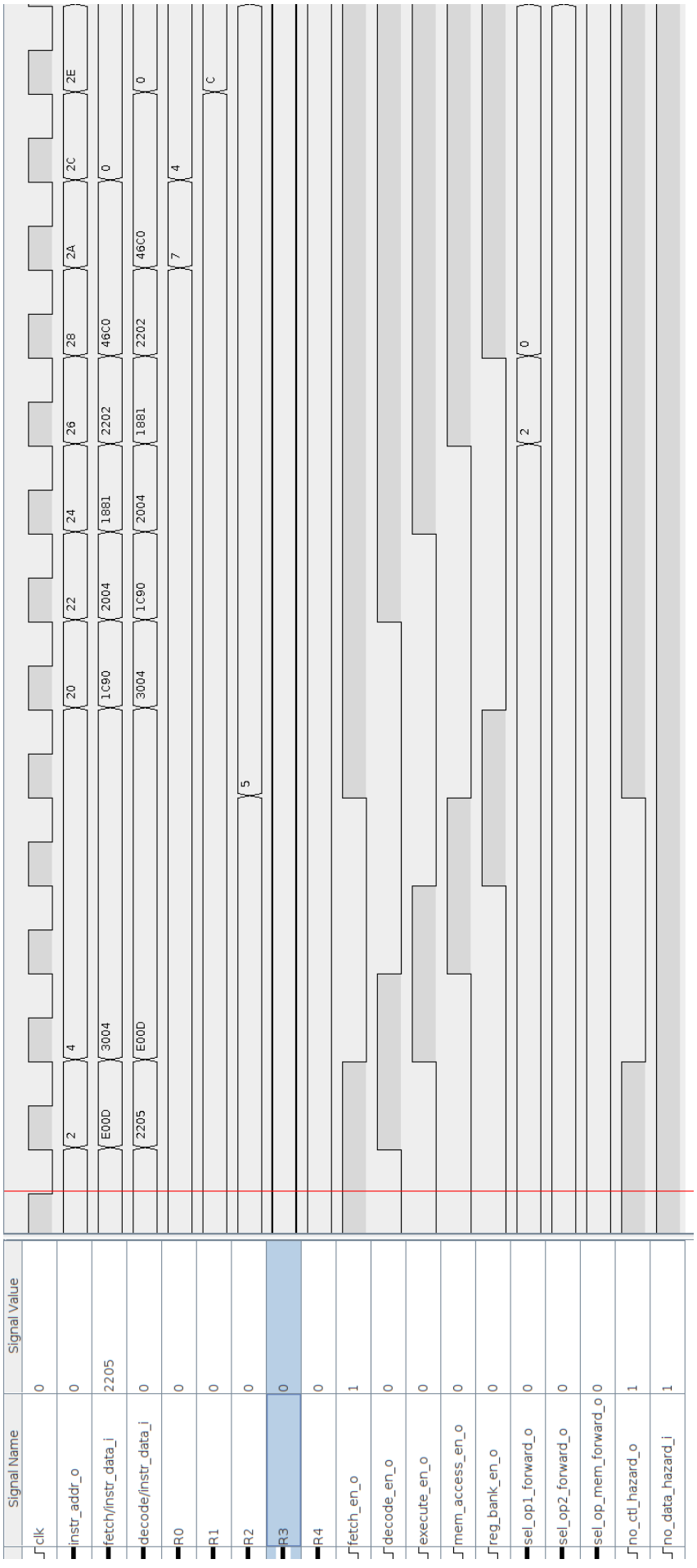

FIGURE 3 – Question 3 : Chronogramme