Exercices computer architecture

Chapter IV

Exo 7 :

- si on a une instruction ALU qui finit au stage E, il peut forward sa valeur (si besoin) depuis le prochain path forward même si c'est un stage M1
- si on a pas de forwarding path au M1, on doit stall jusqu'au prochain forwarding path
- OOO c'est out of order
- on ne doit jamais avoir deux unités qui s'exécutent en même temps, on doit sall (structural hazards)

Exo 12 :

- ne pas oublier les dépendances dans les boucles!
- ici on a deux dépendances, on prend tjrs le dernier `write` comme dépendance

```
addi t0, ---
addi t1, t0, t2
addi t2, t0, t0
```

- on fait une flèche du dépendent vers la dépendance (3 → 1)
- ne pas oublier qu'à la dernière itération on doit aussi stall le temps que la branche soit connue
- ne pas faire dessiner de "forwarding" depuis E quand la branche est connue
- on doit prioriser le **M** sur le **F**.
- pendant le addi, le stage M ne fait rien

Exo 15

- pour le ideal **CPI** ils prennent la moyenne (1/3) sans tenir compte des derniers cycles dûs aux dernières instructions
- "valid entries in the re-order buffer" → les entrées dans le ROB fetched mais pas encore commit
- "counted as modified in the re-order buffer" → les entrées commit, dont la valeur est arrivée, etc.
- bien comprendre le code pour faire du loop unrolling! pour qu'il soit vraiment efficace

Exo 16

- **toujours** prendre le temps d'entourer les dépendances à gauche

- ne pas oublier que le **IPC** c'est 1 quand il n'y a aucune dépendance (ou **2** si on lance deux instructions à la fois -- pas la peine de faire des calculs plus compliqués)

Exo 18

- on ne met pas les instructions déjà exécutées dans la RS (quand une instruction est terminée, on l'enlève de la RS, on efface le champ "tag" du ROB et on met la valeur en dur dans le ROB)
- une fois qu'on a choisi une place dans la RS, on doit garder ce même choix pour la suite (une instruction ne va pas se déplacer toute seule d'un tag vers un autre : https://edstem.org/eu/courses/1568/discussion/164267)

Chapter V

Exo 1

- "holds 32 bytes of data" $\rightarrow$ on est en byte-addressable (donc chaque adresse comme 3000 fait un byte). Avec 32 bytes on peut stocker 32 adresses donc 32/4 = 8 words. 3000, 3004, 3008, 30
- on considère une ligne invalide même si elle n'est pas présente dans le cache (invalide c'est non caché **ou** caché et invalidé).
- on peut faire un **W** et un **D** en même temps (contrairement aux autres path forwarding)
- pour **sw** une ligne non cachée on utilise **Mx** (**BusReadX**)

Exo 2

- on doit attendre que toutes les opérations de mémoire soient terminées avant de commencer la prochaine (pas de pipeline)
- considérer plus de 2 processeurs dans le cas de la liste des états question d.